Yari Marchetti

ARCHITETTURA DISTRIBUITA PER IL MONITORAGGIO DELLA RETE DI COMUNICAZIONE IN UN SISTEMA GRID

Tesi di Laurea Specialistica

Università di Pisa

Dicembre 2007

UNIVERSITA' DEGLI STUDI DI PISA

Facoltà di Scienze, Matematiche, Fisiche e Naturali

Corso di Laurea in Tecnologie Informatiche

ARCHITETTURA DISTRIBUITA PER IL MONITORAGGIO DELLA RETE DI COMUNICAZIONE IN UN SISTEMA GRID

Tesi di

Yari Marchetti

Relatore:

Prof. A.Ciuffoletti

Chi non ha mai commesso un errore

non ha mai tentato qualcosa di nuovo.

Albert Einstein

Sommario

La rete di comunicazione è una risorsa fondamentale per le applicazioni che utilizzano sistemi Grid, risorsa che deve essere tenuta sempre sotto stretto controllo sia per poterla sfruttare al massimo delle sue capacità che per individuare possibili problemi. La tesi esplora i requisiti di una attività di monitoraggio di rete, rivolgendo l'attenzione al controllo delle prestazioni durante l'esecuzione di una applicazione di Grid. Viene presa in esame l'architettura astratta proposta nell'ambito del progetto CoreGRID Network Monitoring, e se ne esaminano problematiche e fattibilità, affrontando e risolvendo questioni riguardanti il recapito delle richieste di monitoraggio e delle misurazioni. Attraverso l'implementazione di un prototipo nel linguaggio Java se ne dimostra la realizzabilità e la possibilità di integrazione all'interno di progetti esistenti. Il prototipo viene infine messo alla prova in un testbed virtuale, costituito da una rete realizzata con la tecnologia User Mode Linux su un singolo computer. I risultati si inquadrano nel Progetto Europeo CoreGRID, e contribuiscono direttamente ai risultati dell'istituto virtuale Grid Information, Resource and Workflow Monitoring".

Dedico questa tesi alla mia famiglia senza la quale non sarei sicuramente riuscito ad arrivare fino a qui.

Indice

Capitolo 1

Introduzione

L'introduzione di nuovi modelli di calcolo consente di superare limiti che parevano inavvicinabili, permettendo di utilizzare i computer nella risoluzione di problemi molto complessi in ambiti quali la medicina o la biologia per esempio.

Per oltrepassare questi limiti, un modo che sta ricevendo sempre più attenzione, in quanto consente di abbinare bassi costi ad ottimi risultati, è quello della *computazione distribuita*. Invece di effettuare tutti i calcoli su un solo computer, e quindi avere dei limiti fisici imposti dall'architettura del computer, i calcoli sono suddivisi fra molti computer diversi, ognuno che lavora in maniera coordinata su parte del problema. Questo tipo di computazione prende anche il nome di *Grid Computing*.

In [2] una *grid* viene definita come: Una infrastruttura distribuita, di-

namicamente riconfigurabile, scalabile ed autonoma che fornisce un accesso indipendente dalla posizione, pervasivo, affidabile, sicuro ed efficiente ad un insieme di servizi che incapsulano e virtualizzano risorse per generare conoscenza. Questa tesi è stata sviluppata nel quadro del progetto Coregrid [2], da cui è tratto il riferimento precedente. Coregrid è una iniziativa della comunità europea che mira a rafforzare e sviluppare le conoscenze scientifiche e tecnologiche nell'ambito del *grid computing*. All'interno di Coregrid trovano posto ambiti di ricerca diversi che studiano le diverse problematiche e soluzioni necessarie alla creazione di una struttura condivisa per il *Grid Computing*: questa tesi è inquadrata all'interno del progetto Network Monitoring sviluppata in collaborazione con il FORTH [3] i cui risultati contribuiscono direttamente all'istituto virtuale Integrated Resource and Workflow Monitoring. In [6] si possono trovare riferimenti specifici all'architettura del progetto Coregrid ai componenti ed alle modalità di integrazione.

Molte entità di ricerca ed aziende si sono mosse allo studio delle applicazioni di tipo grid, ognuna in maniera indipendente, facendo sorgere l'esigenza di integrare lavori con un comune denominatore. OGF [4] è una comunità di utenti, sviluppatori, e vendors che si occupa di coordinare tutti gli sforzi di standardizzazione svolti nell'ambito del grid computing. Nato dalla fusione di due organizzazioni diverse, il Global Grid Forum e l'Enterprise Grid Alliance, integra al suo interno una serie di gruppi di lavoro provenienti

dal mondo della ricerca e dell'industria. Il lavoro svolto da questi gruppi è mirato allo sviluppo di standard e specifiche comuni al fine di accelerare l'adozione e sviluppare standard aperti per l'interoperabilità fra software di grid. In [15, 14, 8] si può trovare una descrizione generale delle problematiche, dell'architettura e dei meccanismi che si ritrovano nelle applicazioni di tipo grid.

Il concetto che sta alla base del funzionamento di una grid è la connettività comune fra i computer che eseguono i calcoli. Ogni computer deve essere in grado di ricevere i dati sui quali effettuare la computazione e poi poter spedire i risultati alla destinazione finale. Per poter effettuare questi trasferimenti è necessario quindi che ogni computer abbia accesso ad una struttura di rete comune che sia in grado di interconnettere in maniera efficiente i partecipanti alla grid.

In una computazione di questo tipo molteplici sono i computer che partecipano, ognuno mettendo a disposizione risorse di tipo diverso (capacità di calcolo, spazio disco, etc.). L'uso di queste risorse deve essere coordinato in maniera efficiente e sicura, cercando di ottenerne la massima resa, e permettendone l'impiego solo a soggetti autorizzati (per esempio, coloro che partecipano al progetto). L'ottimizzazione sull'uso delle risorse di un grid passa anche attraverso un uso consapevole della rete di interconnessione ed il fattore decisivo per il raggiungimento di questo obbiettivo è la conoscenza

dello stato in cui si trova. Per avere tale informazione è necessario effettuare delle misurazioni che consentano di definire il valore di alcuni parametri caratteristici, per esempio la banda usata su uno specifico tratto di rete in un preciso istante. Il monitoraggio di rete, come è inteso comunemente, è eseguito ad intervalli regolari di tempo, con strumenti fissi, in alcuni punti ben definiti della infrastruttura di rete, registrando le misurazioni effettuate in depositi dati, noti a priori. I dati raccolti mediante queste misurazioni vengono poi analizzati per stabilire se l'uso che viene fatto dell'infrastruttura è in linea con quanto atteso o meno, permettendo quindi di individuare eventuali problemi.

Durante lo sviluppo di questa tesi si mostrerà perché l'uso di un tipo di monitoraggio di rete come quello appena descritto (definito *monitoraggio programmato*) non risulti adatto a contesti molto dinamici come quello di grid. In un contesto di grid è necessario avere un controllo approfondito dei parametri di monitoraggio (è necessario poter specificare come, dove, e per quanto si debba eseguire il monitoraggio) ed avere a disposizione le misurazioni in maniera celere, in quanto necessarie come dato per l'ottimizzazione dei processi. Verranno mostrati quali sono i problemi derivanti dall'uso di un *monitoraggio programmato*, e si definirà un nuovo tipo di monitoraggio, chiamato *monitoraggio on demand*, che consente di risolvere i problemi esposti.

Si procederà allo studio delle problematiche legate al *monitoraggio on demand*, mostrando quali queste siano e come possano essere risolte. Si passerà quindi a definire quali sono le componenti necessarie alla realizzazione di un'infrastruttura per il *monitoraggio on demand*, infrastruttura complessa, composta da parti diverse, ognuna complementare all'altra. Si presenterà infine l'architettura interna di un agente che realizza il trasferimento delle richieste e delle misurazioni, ne verranno spiegate le parti che lo compongono e le tecnologie usate per sviluppo.

Nel capitolo due si analizzeranno le problematiche legate al monitoraggio *on demand* ed alla sua implementazione, nel capitolo tre si definiranno le componenti necessarie alla realizzazione di una infrastruttura per il monitoraggio *on demand* e nel capitolo quattro si mostrerà l'implementazione di un agente e dell'ambiente di test. A seguire una introduzione approfondita sulle caratteristiche del monitoraggio programmato e di quello *on demand*.

1.1 Monitoraggio di rete: concetti e motivazioni

Il termine *monitoraggio di rete* descrive l'uso di sistemi e metodologie atti a valutare in maniera più oggettiva possibile l'uso e lo stato di apparecchia-

ture ed interconnessioni che formano la struttura portante di una rete di computer. Il monitoraggio di rete ha una importanza fondamentale per l'insieme più ampio di funzioni che è quello della *gestione di rete*, all'interno del quale ricadono tutte quelle operazioni che servono a controllare, pianificare, allocare, coordinare e monitorare le risorse di una rete di computer.

L'importanza strategica del monitoraggio di rete è data dal valore delle misurazioni che da questo si possono ricavare. Questi dati possono essere usati in diversi ambiti della gestione di rete, e non: servono per migliorare la qualità dell'infrastruttura nel suo insieme, consentono di individuare problemi ad interconnessioni o apparecchiature di trasporto, sono indispensabili per determinare punti dell'architettura che necessitano di miglioramenti strutturali, consentono di determinare la presenza di traffico anomalo o non preventivato, possono essere usati per l'accounting di risorse utilizzate, ed in molti altri casi. L'uso di una struttura efficiente e l'impiego di raffinate metodologie di monitoraggio sono una delle basi fondamentali dalle quali non si può prescindere se si vuole ottenere scalabilità ed efficienza da strutture di rete non banali. Tanto più cresce la complessità della struttura monitorata, tanto più sorge la necessità di trovare strutture, strumenti e metodologie che consentano di ridurre la complessità del monitoraggio.

Le metodologie di monitoraggio di rete possono essere divise principalmente in due categorie: quella passiva e quella attiva: nella prima le misura-

zioni sono effettuate in modo da sollevare totalmente l'infrastruttura di rete da carico aggiuntivo recuperando le misurazioni dalle apparecchiature di interconnessione, nella seconda si usano dei messaggi aggiuntivi che transitano sulla rete per poter effettuare le misurazioni. L'uso di misurazioni passive è quindi in generale da preferirsi rispetto a quelle attive in quanto non si appesantisce ulteriormente l'infrastruttura con traffico aggiuntivo, e quindi non si sprecano risorse che potrebbero essere utilizzate per altri scopi. Inoltre visto che il monitoraggio attivo fa affidamento su messaggi che usano la rete stessa come mezzo di trasporto, questi potrebbero essere a loro volta soggetti a rallentamenti, scheduling o addirittura essere scartati durante il tragitto, dando quindi una rappresentazione inesatta dello stato reale dell'infrastruttura. Il problema delle misurazioni passive è che per alcuni tipi di misurazioni questa metodologia non è applicabile, cosa vera anche per le misurazioni attive, ma in un numero minore di casi. Il problema dell'uso di misurazioni passive/attive è oggetto di studio da ormai moltissimi anni [27, 29, 16, 20, 22] ed alcune soluzioni, come [29], sono state sviluppate per combinare insieme l'uso di misurazioni sia attive che passive. Queste soluzioni mirano a massimizzare l'attendibilità dei risultati, minimizzando però l'intrusività delle misurazioni. Come descritto in [7] quindi, un uso combinato di misurazioni passive ed attive risulta essere la scelta migliore in moltissimi scenari; usando il monitoraggio passivo quando possibile, e lasciando al monitoraggio attivo

tutti gli altri casi.

Gli strumenti per effettuare le misurazioni possono essere divisi a seconda del tipo di monitoraggio, attivo o passivo, che si intende effettuare. In caso di monitoraggio attivo si avranno strumenti che generano dei messaggi che vengono spediti sulla infrastruttura di rete ad host specifici, dai quali si attende una risposta per effettuare la misurazione. Il più famoso di questa tipologia di strumenti è sicuramente Ping, che genera un pacchetto ICMP echo request che viene spedito attraverso la rete ad un host, il quale rispedisce indietro un pacchetto echo reply, consentendoci quindi di valutare la raggiungibilità di un host remoto, ed avere una stima del tempo di andata/ritorno (round-trip time). Uno dei problemi lasciato aperto da questo tipo di strumenti è però, per esempio, l'impossibilità di fare misurazioni nel caso in cui ci siano delle unità che bloccano il passaggio di alcuni tipi di traffico (come un firewall per esempio), problema che trova parziale soluzione nell'uso di strumenti più avanzati. Nel caso di monitoraggio passivo, le misurazioni vengono effettuate su host specifici (di solito apparecchiature intermedie) che vedono passare il traffico generato da altri e si occupano di salvare in strutture dati interne i valori misurati. Un tipico esempio di questo sono i contatori che si possono trovare nei router di Internet che tengono traccia dei pacchetti spediti su una certa interfaccia di rete, ed a cui si può accedere da remoto mediante SNMP [10] per leggerne le misurazioni.

Allo stato attuale delle cose, il monitoraggio di rete è eseguito con una ampia gamma di strumenti, anche molto diversi fra loro, ognuno con sue peculiarità, che permettono di monitorare aspetti diversi della rete. Questa disomogeneità ha portato alla necessità di avere decine e decine di strumenti sempre disponibili ovunque si volesse effettuare una misurazione, a forte discapito della dinamicità del monitoraggio. Essendo stato realizzato in maniera indipendente dagli altri, ogni strumento ha una propria interfaccia per ricevere i parametri e per passare le risposte; da questo fatto ha preso spunto la ricerca negli ultimi anni per sviluppare delle architetture all'interno delle quali integrare tutti gli strumenti di monitoraggio. Avere una struttura unica, con interfacce ben definite, attraverso la quale si può accedere a moltissimi tipi di monitoraggio diversi, consente di facilitare l'accesso alle misurazioni, favorendone quindi l'utilizzo. Due esempi che si muovono nella direzione citata sono il Network Weather Service [27] e NPM [1], in entrambi i casi le misurazioni sono eseguite mediante l'uso di una architettura ben definita che si occupa anche di trasferirle e renderle disponibili ad altre entità.

Con l'introduzione di architetture integrate sta cambiando il ruolo che assumono i dati ricavati dal monitoraggio; questi non vengono più soltanto usati per fault detection e accounting, un ruolo prettamente *passivo*, ma anche per fornire informazioni fondamentali alle decisioni, portandoli quindi ad assumere un ruoto di primaria importanza ma soprattutto un ruolo *attivo*.

L'uso di informazioni di monitoraggio può risultare fondamentale per l'ottimizzazione di trasferimenti, per l'attivazione di flussi con requisiti specifici (tempo reale per esempio), ed in molti altri casi in cui avere informazioni sullo stato della rete può evitare uno spreco di risorse e garantire una migliore qualità dei risultati.

1.2 Monitoraggio programmato

Al concetto di monitoraggio di rete vi è, fino ad ora, accostato praticamente sempre il concetto di programmazione. Le misurazioni sono effettuate ad intervalli di tempo regolari, da punti di monitoraggio ben definiti, verso altre destinazioni anche queste ben definite. Le possibilità di interazione, e quindi di personalizzazione, con questo tipo di monitoraggio sono ridotte al minimo, ed in molti casi sono del tutto assenti.

Questo modo *programmato* di effettuare le misurazioni è il riflesso naturale dell'uso che viene fatto dei dati di monitoraggio nella maggior parte degli casi: i risultati vengono raccolti come un'istantanea dello stato dell'infrastruttura di rete in un determinato momento. Gli usi dei dati raccolti vanno dall'accounting delle risorse utilizzate per fini commerciali, al riconoscimento di tipologie di traffico anomalo, piuttosto che valutazioni sull'impatto di nuovi servizi che utilizzano l'infrastruttura di rete.

Tutte queste tipologie di utilizzo hanno un fattore comune: è necessario avere a disposizione le misurazioni effettuate in passato, tenendo traccia di ogni singola misurazione o con l'uso di medie per intervallo, per esempio; i risultati del monitoraggio vengono poi raccolti in apposite strutture dati preparate in precedenza. Questo fatto introduce un altro problema nell'infrastruttura di monitoraggio: l'accessibilità dei dati di monitoraggio da parte di soggetti interessati. Essendo le misurazioni disponibili solo nei supporti preparati da chi ha predisposto il monitoraggio, questi potrebbero non essere disponibili per tutti gli interessati (basti pensare al caso di un contatore SNMP di un router) ma solo per coloro che ne hanno richiesto l'accesso all'amministratore, facendo di fatto insorgere la necessità di concedere le autorizzazioni ad ogni soggetto interessato.

Un altro fattore da considerare nel monitoraggio programmato è la totale discrezione nella scelta delle modalità con cui le misurazioni vengono eseguite da parte del controllore dell'host. Le misurazioni, nella maggior parte dei casi, vengono effettuate senza renderne pubbliche le modalità ne tanto meno gli strumenti, lasciando quindi i fruitori di tali misurazioni nell'impossibilità di valutarne l'attendibilità. Una misurazione effettuata con metodologie sbagliate o comunque diverse da quelle attese può non essere significativa o può addirittura essere dannosa. Questo problema diventa tanto più sensibile tanto più le misurazioni effettuate hanno requisiti stringenti di accuratezza

ed affidabilità.

1.2.1 Strutture necessarie al monitoraggio programmato

Le strutture necessarie per effettuare monitoraggio programmato si possono sintetizzare in due entità, una che effettua le misurazioni e l'altra che salva i risultati.

Per effettuare le misurazioni è necessario un agente equipaggiato con gli strumenti necessari, come per esempio Ping, ed che questo sia inoltre in grado di comunicare (connessione di rete, messaggi fra processi, ecc.) con l'entità che li salva. Questo agente deve poter effettuare il monitoraggio ad istanti ben definiti e deve essere in grado di portare a termine richieste cicliche.

L'agente che salva le misurazioni deve essere pronto a ricevere, e salvare, i dati che gli vengono spediti da chi effettua le misurazioni (più agenti possono confluire nella solita struttura di salvataggio). Alcune operazioni possono essere eseguite sui dati ricevuti prima di salvarli al fine di ridurre lo spazio di immagazzinamento richiesto o per calcolare altri valori significativi da quelli già presenti. Un esempio di questo è il sistema di previsione del traffico che si può trovare in Network Weather Service [26], in cui sui dati raccolti vengono applicate delle euristiche per cercare di prevedere il traffico futuro partendo

da misurazioni disponibili.

In generale è molto semplice definire dei canali sicuri per il trasferimento dei dati fra chi effettua le misurazioni e chi li deve immagazzinare: questo è possibile in quanto si conosce a priori identità e posizione di chi effettua le misurazioni, orario di inizio e fine, tipologia e quantità dei dati che si andranno a trasferire.

1.3 Monitoraggio *on demand*

Una nuova attenzione da parte della ricerca si sta spostando verso questa parte ancora poco esplorata del monitoraggio. Come nel caso *programmato* quello che si vuole fare è ricavare delle misure di alcune metriche delle performance di rete; cambiano però le modalità ed in parte le motivazioni di come questo si realizza.

Le differenze sostanziali del monitoraggio programmato da quello *on demand* si hanno nella possibilità di non conoscere a priori la destinazione della richiesta, del tipo di monitoraggio, l'identità del partner e l'indirizzo dove andranno spedite le misurazioni effettuate. Tutte queste alternative aprono nuovi scenari di utilizzo che non erano prima possibili con il monitoraggio programmato.

Lasciare libertà a chi fa richiesta di monitoraggio di poter stabilire ti-

pologia e modalità con le quali verrà poi effettivamente eseguito, consente sempre di avere misurazioni delle quali è possibile garantire l'utilità, cosa che non accadeva nel monitoraggio programmato in cui era chi effettuava il monitoraggio che faceva queste scelte. Inoltre la possibilità di scegliere *on demand* lo strumento e le modalità dà la possibilità di avere un dinamismo nelle richieste che è irraggiungibile nel monitoraggio programmato.

Nel monitoraggio *on demand* si slega completamente la conoscenza da parte di chi vuole ricevere i dati di monitoraggio dalla posizione di chi effettivamente lo porterà a termine. Questo consente da un lato, di delegare alla sottostante infrastruttura di monitoraggio il compito di trasferire le richieste agli agenti in grado di soddisfarle, e dall'altro consente anche a chi genera i dati non aver bisogno di conoscere l'indirizzo di dove questi andranno a finire. Questo permette di esplicitare un effettivo disaccoppiamento fra produttori e consumatori di dati.

La scissione che c'è fra i produttori e consumatori di dati ha notevoli ripercussioni sia sui primi che sui secondi. Lasciando alla struttura sottostante il compito di trasferire le richieste ci si trova nella condizione di dover verificare l'autenticità dei messaggi ricevuti e l'identità del mittente per evitare usi impropri. Questi accorgimenti non erano necessari nel monitoraggio programmato in quanto le comunicazioni fra produttori e consumatori erano dirette. Dall'altro lato, questo disaccoppiamento consente l'introduzione di

meccanismi interni all'infrastruttura per il caching di eventuali misurazioni già effettuate, diminuendo così l'impatto sull'infrastruttura di rete.

Un altro punto di forza del monitoraggio *on demand* è quello di permettere il monitoraggio non su base fissa, come nel monitoraggio programmato, ma in maniera totalmente personalizzata. La scelta di tutti i parametri caratteristici del monitoraggio avviene nel momento in cui si effettua la richiesta, e non è fissata a priori, di fatto lasciando al richiedente pieno controllo sul monitoraggio.

Con il monitoraggio *on demand* si cerca di utilizzare i dati di monitoraggio, che normalmente richiedono un tempo lungo per poter essere raccolti e trasferiti, come dati locali, ai quali si può accedere quando, come e da dove si vuole. Da notare è anche il fatto che una struttura per il monitoraggio *on demand* può facilmente implementarne una per il monitoraggio programmato con un uso accorto delle risorse; non è altresì vero il contrario.

Alcune ricerche nel campo della computazione distribuita hanno mostrato caratteristiche simili al monitoraggio *on demand* senza però avere tutte le peculiarità necessarie ad implementare una infrastruttura di questo tipo.

In [18] viene mostrato l'uso di una architettura publish/subscribe per la distribuzione di CRL (Certificate Revocation Lists). Questo tipo di soluzione è sicuramente interessante ed ha notevoli pregi, per esempio la rapidità con la quale i client interessati ricevono gli aggiornamenti e la possibilità di evitare

misurazioni multiple nel solito istante, ma risulta difficilmente applicabile al caso specifico del monitoraggio *on demand*. L'uso di una architettura di tipo publish/subscribe è molto semplice in contesti in cui siano statici e ben definiti i servizi che si vogliono fornire (come nel caso del monitoraggio programmato). In casi come quello del monitoraggio *on demand*, in cui ogni richiesta è in generale diversa dalle altre (si pensi per esempio alla richiesta di una misurazione che riguarda l'host A piuttosto che il B) l'introduzione di questo tipo di architettura è impossibile o comunque problematica. Lo spreco di risorse e la limitata libertà di azione che verrebbero introdotti con l'uso di questo tipo di soluzione eliminerebbero tutti i benefici che questo tipo di architettura porta con se, rendendola di fatto non adatta allo scopo.

In [21] viene definita un'infrastruttura per il monitoraggio passivo distribuito, molto interessante in quanto si ritrova un disaccoppiamento fra produttori e consumatori. I Measurement Point (MP) effettuano le misurazioni, i Measurement Area (MAr) fungono da controllori per i MP, ed i consumer accettano i pacchetti derivanti dalle misurazioni. Il problema principale che non ne consente l'applicazione in un contesto di monitoraggio *on demand* deriva dal fatto che, essendo studiato specificatamente per misurazioni passive, da effettuare su interfacce di rete, non si prevedono modalità di monitoraggio specifiche ma ci si limita sempre alla cattura del traffico, imponendo di fatto un solo tipo di monitoraggio ai consumer. Inoltre l'infrastruttura proposta

è studiata per essere applicata a contesti locali, quindi con un numero di client limitato di fatto ponendo un limite superiore all'espandibilità dell'infrastruttura. Per l'integrazione di più MAr fra loro è previsto l'inserimento di un nuovo soggetto, definito Super Measurement Areas (SMAr), che consente di raggruppare più MAr sotto un unico SMAr ed emettere le richieste verso questo, piuttosto che verso i singoli MAr, ma questo risolve solo parzialmente il problema della scalabilità. Nello studio non vengono definiti molti aspetti necessari, come per esempio la coordinazione fra più SMAr (l'uso di uno SMAr solo non è una soluzione applicabile), aspetti che, in contesti fortemente distribuiti e con un numero alto di produttori e consumatori, devono essere studiati attentamente, pena il fallimento dell'infrastruttura.

Nello studio della bibliografia sulla computazione distribuita i Mobile Agents [17] sono risultati una delle tecnologie emergenti più interessanti. Un mobile agents è un componente software, comprendente sia dati che codice, che è in grado di essere spedito ad un server remoto, ed eseguire su questo computazioni usando risorse locali (dati, memoria, etc.); vengono poi ritrasferiti indietro i soli i risultati elaborati. La possibilità di impiego all'interno di una struttura per il monitoraggio *on demand* è stata considerata, e sotto alcuni aspetti avrebbe semplificato l'architettura, ma alcune problematiche intrinseche della tecnologia non ne hanno consentito l'utilizzo. Usando i Mobile Agents risulta difficile definire delle policy sul tipo di operazioni che un

agente può o non può eseguire, soprattutto all'interno di un contesto come quello del monitoraggio *on demand*. Questo avrebbe insinuato dubbi e perplessità negli amministratori delle macchine sulla correttezza e validità delle operazioni che potrebbero essere eseguite con software proveniente da remoto. Inoltre per un utilizzo sicuro dei Mobile Agents è necessario disporre di un ambiente di esecuzione specifico, ambiente che ha un peso, sia di calcolo che di memoria, non trascurabile, e che quindi ne limita l'applicabilità alle sole macchine con delle risorse ben precise (molte apparecchiature di rete non rientrano in questo lista).

Per l'utilizzo all'interno dell'architettura di monitoraggio *on demand* si è preso in considerazione l'uso del *Session Initiaton Protocol* (SIP) [23]. SIP è un protocollo di signaling, sviluppato originariamente da H.Schulzrinne e M.Handley, per la gestione dello stato di sessioni con più partecipanti. SIP funziona a livello applicazione, in maniera indipendente dal protocollo di trasporto, lasciando quindi piena libertà di scelta sul tipo di infrastruttura su cui può essere impiegato. Una serie di messaggi di testo viene scambiato fra i partner della comunicazione ogni volta che è necessario richiedere l'inizio di una nuova sessione, terminarne una, o scambiarsi informazioni sullo stato di una già attiva. SIP permette di specificare le modalità con le quali si andrà ad effettuare lo scambio di dati fra i partecipanti, ma senza realizzare effettivamente nessun tipo di trasferimento, compito lasciato a protocolli

specifici. L'uso di SIP all'interno di una infrastruttura di monitoraggio *on demand* non è risultato possibile, sebbene le caratteristiche che vi si ritrovano, per la creazione e gestione di sessioni, siano state la base su cui poi si è sviluppato. Il problema fondamentale è la separazione netta che c'è fra SIP ed il protocollo usato per trasportare i dati, separazione che non consente quindi di avere una struttura unica per il routing sia delle richieste che dei risultati. Questa limitazione è risultata impossibile da superare usando SIP come protocollo per la gestione delle sessioni, e questo ne ha poi impedito l'utilizzo nello sviluppo di una architettura per il monitoraggio *on demand*.

Dopo essersi confrontati con le soluzioni già presenti, e non avendone trovata nessuna adatta, è risultato quindi necessario definire una nuova infrastruttura che permettesse di effettuare monitoraggio di tipo *on demand*.

1.3.1 Strutture necessarie al monitoraggio *on demand*

Rispetto al caso programmato, ritroviamo anche qui un agente che si occupa di fare le misurazioni ed anche un agente che si occupa di riceverle (sebbene lo scopo principale non sia solo per lo stoccaggio). Quello che c'è in più rispetto al caso programmato è una infrastruttura interposta fra chi produce le misurazioni e chi le riceve che permette il trasferimento delle informazioni in maniera autonoma fra le due entità.

Attenzione particolare deve essere posta sul fatto che, essendo le misurazioni effettuate *on demand*, i parametri con i quali il monitoraggio viene eseguito non sono noti a priori. Identità del mittente, tipologia, orario di inizio e durata, sono tutti parametri che possono essere scelti a piacimento dell'agente che richiede il monitoraggio, imponendo quindi una svolta radicale.

Capitolo 2

Analisi delle problematiche legate al monitoraggio *on demand*

2.1 Le problematiche

Nel passaggio da un tipo di monitoraggio programmato ad uno *on demand* una serie di problematiche legate alle nuove strutture ed alle interazioni necessarie fra gli attori che partecipano al monitoraggio devono essere risolte.

Il primo problema da risolvere nasce dalla presenza di una nuova entità che partecipa al monitoraggio, ossia chi effettua la richiesta. Nel monitoraggio programmato, tutto è definito staticamente a priori, non sono necessarie

comunicazioni o richieste per far iniziare il monitoraggio; le azioni necessarie sono svolte sulla base dello scorrere del tempo ad istanti ben definiti. Inserendo all'interno di questa architettura una nuova entità che effettua richieste e si aspetta dei risultati, si determina la necessità di sviluppare una metodologia che permetta di far comunicare chi effettua il monitoraggio con la controparte.

Un altro problema a cui si deve trovare soluzione è quello del trasferimento delle richieste e dei risultati di monitoraggio fra chi le genera e chi ne è invece destinatario. Nel monitoraggio programmato non c'è bisogno di effettuare nessun tipo di richiesta, inoltre conoscendo già a priori la destinazione dei risultati del monitoraggio dei canali per il trasferimento vengono preventivamente predisposti. Nel monitoraggio *on demand* questo non è invece possibile dato che le richieste possono arrivare da qualsiasi agente, anche da alcuni di cui non si conosce la posizione; i risultati parimenti devono essere rispediti indietro, a destinazioni potenzialmente sconosciute. La possibilità di slegare la conoscenza della posizione di un generico agente dal modo in cui i dati vengono recapitati, è sì un vantaggio ma anche un problema da risolvere: avere a disposizione la globalità delle informazioni di indirizzamento degli agenti non è una cosa possibile, se non in casi banali.

Un problema che si ritrova in tutti gli ambienti distribuiti, quello dell'identità e dell'autenticazione di agenti remoti, non fa certo eccezione all'in-

terno della struttura di monitoraggio *on demand*. L'importanza di conoscere l'identità dell'agente che fa richiesta di monitoraggio cresce al crescere della sensibilità delle misurazioni: dare la possibilità a soggetti non autorizzati di accedere ad informazioni riservate può avere effetti catastrofici. Un tipico esempio di questo problema si ritrova nel protocollo SNMP v1 e v2, nel quale la mancanza di meccanismi di autenticazione (o con meccanismi molto blandi) portava a rivelare dati sensibili anche a soggetti male intenzionati; problema al quale si è trovato soluzione solo con i nuovi meccanismi presenti nella revisione 3 del protocollo (vedere [28]). Oltre al problema appena esposto la conoscenza delle identità risulta fondamentale per evitare usi impropri di risorse di monitoraggio o, nei casi peggiori, per evitare attacchi all'infrastruttura di rete stessa o ad host che si appoggiano a questa.

Rispetto al caso programmato, in cui ogni agente che effettuava il monitoraggio sa quali strumenti usare ed il modo in cui effettuare le rilevazioni, nel monitoraggio *on demand* questo non è in generale possibile a priori: la conoscenza della tipologia e delle modalità di misurazione dipende esclusivamente dalla richiesta di monitoraggio. Quando un agente di monitoraggio riceve una richiesta deve essere in grado di interagire con un gran numero di strumenti diversi, ognuno in grado di effettuare tipologie di misurazioni diverse; situazione diametralmente opposta a quella che si ritrova nel monitoraggio programmato. In quest'ultimo c'è un'infrastruttura specifica intorno

ad ogni singolo strumento, in quello *on demand* l'infrastruttura è unica, e quindi questa si deve adattare a tutti i possibili strumenti. Ne risulta quindi un problema di interazione fra chi riceve le richieste e gli strumenti che effettuano le misurazioni; questo problema deve trovare soluzione tenendo conto del fatto che esistono strumenti per misurazioni molto eterogenei fra loro. L'interazione con strumenti già esistenti, quindi non predisposti al monitoraggio *on demand*, deve essere garantita visto che la riscrittura di codice già esistente, solo per essere integrato all'interno dell'infrastruttura *on demand*, non sarebbe una soluzione accettabile (nonché possibile in molti casi).

La struttura di monitoraggio *on demand* soffre di un problema che quella di monitoraggio programmato non aveva, ossia la scalabilità all'aumentare del numero di partecipanti al monitoraggio. Se consideriamo che ogni agente tenga traccia direttamente di tutte le informazioni necessarie all'indirizzamento e all'autenticazione, all'aumentare del numero di agenti aumenta anche l'impatto sullo stato di ogni singolo agente, limitando di fatto la scalabilità. In [27] si ritrova un problema simile dato che le misurazioni vengono eseguite fra ogni coppia di sensori, richiedendo quindi l'esecuzione di numero che è il quadrato rispetto al numero di sensori. La soluzione proposta dagli autori è stata quella di organizzare in *clique*, ossia in gruppi, i sensori; ogni sensore effettua misurazioni solo verso gli altri sensori appartenenti al clique e può partecipare a più di uno di questi. Organizzando i clique gerarchica-

mente in livelli, e deputando all'interno di ogni clique un solo rappresentante alla comunicazione verso l'alto e verso il basso della gerarchia, il problema della scalabilità ha un impatto limitato ai solo partecipanti al clique.

2.2 Le possibili soluzioni

Definiamo ora un insieme di possibili soluzioni ai problemi sopra esposti che consentiranno poi di elaborare una infrastruttura per il monitoraggio di rete *on demand*.

2.2.1 Le richieste di monitoraggio

Con l'introduzione di una nuova entità all'interno dello schema già esistente del monitoraggio programmato, si delinea chiaramente la necessità di sviluppare una sintassi ed una semantica che consenta di esprimere la volontà da parte di un agente di avviare e ricevere le misurazioni su uno specifico monitoraggio. Una soluzione che risulta essere molto semplice ed efficace è quella di esprimere questa volontà mediante una richiesta, che chiameremo *richiesta di monitoraggio*, la quale contiene tutti i dati necessari ad iniziare un monitoraggio *on demand*.

Ragionando sui problemi sopra esposti si delinea la necessità di avere una parte comune a tutte le richieste di monitoraggio ed una parte che invece

può variare da richiesta a richiesta. La parte comune è necessaria in quanto una serie di parametri, per esempio la sorgente del monitoraggio e la durata, sono presenti in qualsiasi tipo di richiesta in quanto fondamentali per l'avvio del monitoraggio. Osservando però il problema sopra esposto dell'interazione con molti strumenti diversi si delinea anche la necessità di poter specificare dei parametri che non sono universali, ma che invece dipendono dal tipo specifico di monitoraggio che si vuole effettuare. Questi parametri sono necessari alla stregua di quelli comuni in quanto il corretto uso degli strumenti di monitoraggio dipende da loro. Questi parametri devono essere specificati tenendo presente lo strumento ed il tipo di monitoraggio che si vuole effettuare: per esempio per effettuare la misura del Round-Trip Time da un punto ad un altro della rete saranno necessari parametri totalmente diversi rispetto alla misura del packet loss su un tratto di rete.

In definitiva si delinea all'interno di una *richiesta di monitoraggio* la presenza di due parti ben distinte, ma complementari fra loro, una che definisce parametri comuni e l'altra che definisce parametri dipendenti dallo strumento che si vuole utilizzare.

2.2.2 L'interazione con gli strumenti di monitoraggio

Mediante l'uso delle *richieste di monitoraggio* un agente interessato può definire in maniera precisa il tipo, lo strumento, ed i parametri necessari per iniziare un monitoraggio *on demand*. Rimane aperto però il problema sull'interazione da parte dell'agente che deve effettuare il monitoraggio con gli strumenti. Molti strumenti sono stati sviluppati non con l'obbiettivo di essere integrati all'interno dell'infrastruttura di monitoraggio *on demand*, ma semplicemente come entità software a se stanti.

Una soluzione che appare ragionevole, e consente di riusare anche software già esistente, risulta quella di sviluppare dei piccoli componenti specifici che si occuperanno dell'interazione fra agente di monitoraggio e strumenti designati alle misurazioni. L'agente definirà un'interfaccia, alla quale ogni componente dovrà aderire, ed attraverso la potrà effettuerà il passaggio delle richieste agli strumenti. Si realizza così il passaggio in maniera uniforme ed indipendente dalle modalità, delegando la conoscenza delle problematiche di comunicazione con gli strumenti ai componenti; all'agente rimane soltanto il compito di scegliere il componente corretto per recapitare la richiesta.

Si osservi inoltre che uno sviluppo dei componenti basato sul polimorfismo semplifica enormemente la fase di definizione, sviluppo, permettendo inoltre un caricamento *on demand*.

2.2.3 Il database condiviso

Nelle problematiche sopra esposte si nota come ci sia necessità da parte degli agenti di avere informazioni sugli altri partecipanti al monitoraggio. Queste informazioni, necessarie per autenticazione e comunicazione, se da un lato devono essere sempre disponibili presso un agente, dall'altro non possono essere memorizzate tutte localmente in quanto, come spiegato sopra, la scalabilità risulterebbe compromessa.

Si pensa quindi all'uso di un database condiviso fra tutti gli agenti che partecipano al monitoraggio: ogni agente che vuole partecipare pubblica all'interno del database le informazioni che lo riguardano e tutti gli altri agenti possono leggerle quando ne hanno necessità.

L'accesso al database condiviso risulta essere di fondamentale importanza per lo svolgimento di tutte le attività legate al monitoraggio *on demand*; nella sua implementazione si deve tenere conto di possibili limiti alla scalabilità ed a noti problemi di *single point of failure*. Per risolvere questi problemi si può ipotizzare l'uso di database distribuiti e replicati, in modo da aumentare scalabilità ed affidabilità dell'intero sistema, punto fondamentale per avere una solida architettura di monitoraggio.

2.2.4 Le identità

All'interno dell'infrastruttura di monitoraggio *on demand* c'è la necessità che ogni agente abbia delle credenziali che ne permettano una identificazione univoca. Questa necessità nasce dal fatto che la sicurezza può essere garantita solo se l'accesso a risorse ed informazioni è limitato ad agenti autorizzati: agenti di cui si conosce l'identità.

Per poter garantire identità di un'agente è necessario che la generazione sia effettuata da una sola entità preposta allo scopo; questa entità deve creare le identità garantendone l'unicità all'interno della struttura di monitoraggio. Per evitare problemi di omonimia o furto di credenziali si può quindi ipotizzare l'uso di una Certification Authority che si fa carico di questo compito. Un agente che vuole creare una nuova identità valida, fa richiesta alla Certification Authority, la quale genera un nome agente univoco ed una coppia (chiave pubblica, chiave privata). La Certification Authority pubblica il (nome agente, chiave pubblica) all'interno del database condiviso, così da rendere pubbliche ed accessibili a tutti queste informazioni, e spedisce (nome agente,chiave privata, chiave pubblica), che è l'identità univoca, all'agente che ne ha fatto richiesta.

Quando un agente riceve la sua identità, questo diventa in grado di firmare i messaggi che spedisce; non appena pubblicate le informazioni nel database

condiviso, a loro volta, gli altri agenti diventano in grado di decidere se l'identità del mittente di un messaggio è corretta o meno.

Capitolo 3

Infrastruttura per il trasferimento di misurazioni di rete

Avendo determinato in maniera precisa quali sono i problemi da risolvere nel monitoraggio *on demand*, ed avendo proposto delle possibili soluzioni, cerchiamo ora di definire quali siano gli elementi necessari allo sviluppo di una architettura per il trasferimento di misurazioni di rete.

Punto fondamentale per l'infrastruttura risultano essere gli agenti che partecipano al monitoraggio. Questi sono gli attori principali, nonché quelli che ne determinano la buona riuscita, quindi definire le capacità e le in-

terazioni fra loro risulta essere di fondamentale importanza. Altro fattore che richiede sicuramente attenzione sono le richieste di monitoraggio: una corretta definizione risulta fondamentale per effettuare un monitoraggio *on demand*. Una organizzazione rigorosa degli agenti è un altro fattore da considerare in quanto necessario ai fini dello del routing ed indirizzamento delle richieste. Va inoltre definito come viene risolto il problema delle identità che, come spiegato sopra, richiede una generazione univoca. Tutti questi aspetti devono essere esplicitati e studiati chiaramente prima di poter procedere allo sviluppo vero e proprio.

Come già accennato questa tesi si inquadra all'interno del progetto Coregrid [2], e che quindi alcune parti sono state sviluppate tenendo conto di necessità da esso derivanti. Questo non toglie generalità alla soluzione proposta, ma anzi ne dimostra l'applicabilità a casi concreti di ricerca.

Definiamo ora in maniera più precisa gli aspetti sopra citati, spiegandone la struttura, il comportamento e le motivazioni.

3.1 Gli agenti di monitoraggio

A differenza del monitoraggio programmato in cui le misurazioni vengono eseguite ad intervalli regolari, in quello *on demand* c'è bisogno di un qualche tipo di entità che effettui una richiesta per iniziare il monitoraggio.

Inoltre considerando le difficoltà da parte di un agente di interfacciarsi con tanti strumenti di monitoraggio diversi, una soluzione ragionevole risulta essere quella di scindere in due entità separate chi effettua le misurazioni da chi invece si occupa solo di ricezione delle richieste. Questo consente di assegnare compiti distinti ad ognuna delle due, rendendo la struttura più definita ed espandibile, oltre ad avere ripercussioni positive sia sulla manutenibilità che sullo sviluppo dei componenti.

Da questa suddivisione risultano quindi chiari tre ruoli all'interno della struttura di monitoraggio:

- Un agente che si occupa di effettuare le richieste di monitoraggio, che chiameremo *Client del Monitoraggio* o CdM

- Un agente che si occupa di effettuare le misurazioni, che chiameremo *Network Monitoring Element* o NME

- Un agente che si occupa di trasferire le misurazioni fra chi effettua la richiesta e chi invece effettuerà le misurazioni, ed ovviamente per trasferire indietro i risultati, che chiameremo *Network Monitoring Agent* o NMA

3.1.1 Network Monitoring Agent

Il ruolo centrale all'interno dell'infrastruttura di monitoraggio è quello del *Network Monitoring Agent*. Gli agenti di questo tipo sono incaricati di ricevere le richieste di monitoraggio dai *Client del monitoraggio*, farle pervenire ai *Network Monitoring Element* in grado di eseguire il monitoraggio, e trasportare indietro i risultati.

Tutta l'architettura di monitoraggio *on demand* si basa sulla possibilità da parte di questi agenti di smistare richieste e misurazioni in maniera autonoma. Gli NMA devono quindi essere in grado di consegnare richieste e risposte senza dover chiedere ai CdM o ai NME alcuna informazione, ma anzi fornendo loro un sistema di recapito affidabile e sicuro.

Un generico NMA riceve le *richieste di monitoraggio* direttamente dai CdM e si occupa di smistarle ai successivi NMA sulla base delle informazioni incluse nella richiesta. Quando un NMA determina che una richiesta è di sua competenza, passa la richiesta di monitoraggio all'NME che è in grado di portarla a termine. I risultati derivanti dal monitoraggio sono quindi rispediti indietro, seguendo il percorso inverso rispetto alla richiesta, fino ad arrivare al CdM che ne aveva fatto domanda. La capacità da parte di un NMA di instradare richieste e risultati verso le destinazioni corrette dipende da due fattori principali: *la richiesta di monitoraggio* e l'accesso al database condivi-

so. Nella richiesta devono essere specificate abbastanza informazioni da rendere identificabile la sorgente del monitoraggio, e nel database condiviso deve essere possibile ritrovare tutte le informazioni necessarie all'indirizzamento.

3.1.2 Client del Monitoraggio

Gli agenti di questo tipo sono quelli che richiedono il monitoraggio di rete, e che quindi si aspettano di ricevere le misurazioni. Ogni *Client di monitoraggio* conosce il modo di contattare uno o più NMA ai quali passa le *richieste di monitoraggio*, e dai quali rimane in attesa dei risultati. Si può pensare che ogni client conosca staticamente gli NMA ai quali può passare le richieste; questo non intacca la struttura di recapito, ne tanto meno la validità del modello generale. Qualsiasi sia il primo NMA a cui viene passata una richiesta, questo è in grado di determinare un instradamento per farla giungere a destinazione.

La generazione della *richiesta di monitoraggio* è compito di questo tipo di agenti, e deve essere fatta in accordo al tipo di monitoraggio che si vuole eseguire. Avendo le richieste una parte comune, ed una parte specifica rispetto al tipo di monitoraggio da eseguire, è necessario che il client abbia la conoscenza necessaria dei parametri specifici dello strumento da usare e dei valori che questi possono assumere. Si è preferito, per il momento non

specificare come questa conoscenza sia ottenuta, lasciando libertà ai singoli sviluppatori di client di decidere le modalità che risultano migliori per loro.

Analogamente alle richieste anche per i risultati è stata lasciata massima libertà su come questi possono essere specificati nei messaggi di risposta. E' quindi lasciato anche a chi sviluppa i client di monitoraggio il compito di comprendere, e quindi utilizzare, i risultati.

3.1.3 Network Monitoring Element

Gli agenti di questo tipo realizzano quello che è il monitoraggio vero e proprio, ricevendo le richieste dagli NMA, interpretando i parametri, eseguendo il monitoraggio, e spedendo indietro i risultati verso i *client di monitoraggio*.

Ogni *Network Monitoring Element* riceve le richieste di monitoraggio dall'NMA attraverso un componente specifico; per ogni tipo di NME si avrà un componente integrato nell'NMA che è in grado di gestirlo. Al momento del caricamento, il componente deve dichiarare all'NMA quale tipo di NME è in grado di gestire; questo permette all'NMA di decidere quale, fra tutti i componenti, è quello in grado di occuparsi di una richiesta.

L'NME è una entità logica che identifica quella parte dell'infrastruttura che esegue le misurazioni e spedisce i risultati. Questo implica che l'implementazione può risultare anche molto diversa a seconda dei casi: si potranno

avere degli NME direttamente integrati nei componenti caricati nell'NMA, ma anche dei processi separati (locali o remoti) che ricevono le richieste dai componenti integrati nell'NMA.

Per ogni misurazione eseguita i risultati vengono incapsulati in un pacchetto, che può essere compreso dagli NMA ai fini del routing, e spediti. L'invio delle misurazioni effettuate è eseguito in tempo reale, senza quindi attendere che si sia completata l'intera fase di monitoraggio. Mano a mano che le misurazioni vengono effettuate, si spediscono, generando di fatto un *flusso di misurazioni*. I dati interni al pacchetto possono essere organizzati in maniera totalmente arbitraria, dipendente solo dallo strumento che li genera, quindi scaricando l'interpretazione sui Client di Monitoraggio.

Presso il FORTH è in sviluppo DiMAPI [25], una soluzione che permette di eseguire un monitoraggio passivo mediante l'uso di sensori distribuiti fra più host. Questa interessante soluzione consente, mediante l'analisi in tempo reale del traffico, di raggruppare pacchetti diversi in entità di livello superiore denominate *flusso*: ogni flusso è una sequenza di pacchetti che soddisfa un dato insieme di condizioni. Utilizzando DiMAPI si possono definire monitoraggi complessi e distribuiti fra molti sensori in maniera semplice e coordinata, lasciando all'infrastruttura il compito di portarla a termine. Al momento attuale l'integrazione dell'architettura qui presentata con quella di DiMAPI è in via di sviluppo: non appena completata sarà possibile utiliz-

zare DiMAPI per effettuare monitoraggi complessi (quindi come una serie di NME) e l'infrastruttura presentata in questo lavoro per trasportare le richieste ed i risultati. L'integrazione fra i due progetti di ricerca è da ricercasi all'interno del progetto CoreGRID, che ne coordina le attività, al fine di avere una infrastruttura comune che copra tutti gli aspetti del monitoraggio di rete all'interno di un grid, dalla specifica delle richieste e dei risultati, fino al monitoraggio vero e proprio.

3.2 Network Monitoring Session Description

Per la definizione delle richieste di monitoraggio la scelta è caduta su *Network Monitoring Session Description* [11], sviluppato all'interno del progetto Coregrid, proprio per i suoi legami stretti e la perfetta aderenza alle necessità di specifica del monitoraggio *on demand*.

Lo schema proposto, in breve NMSD, ci consente di dare una rappresentazione univoca e ben definita di quei parametri necessari al monitoraggio *on demand*. Ritroviamo, per esempio, i parametri per definire ora e data di inizio, la durata, la sorgente e la destinazione del monitoraggio, e l'identità di chi ha effettuato la richiesta.

Da notare la presenza di un identificatore di richiesta, il SessionId, che deve essere generato in maniera *univoca* da chi crea l'NMSD. Trovandosi in un

ambiente distribuito, l'unicità del SessionId viene garantita sulla base dell'unicità dei nomi degli agenti, a loro volta unici perché generati da una entità centralizzata. La necessità di un identificatore univoco per ogni richiesta nasce da problemi legati ad accounting e routing dei flussi di risultati.

Come richiesto per la soluzione delle problematiche relative alle tipologie di monitoraggio è possibile specificare il nome dello strumento da usare ed una serie di parametri dipendenti da questo. Questa peculiarità consente di avere un unico tipo di richiesta per infinite tipologie di monitoraggio, senza per questo dover ogni volta modificare la sintassi o dover prevedere tutti i possibili parametri.

3.3 L'organizzazione degli agenti

Dividendo tutti gli agenti che partecipano al monitoraggio in base alla comune connettività, come definito in [11], si può realizzare un partizionamento molto utile ai fini di monitoraggio: ogni partizione definita in questo modo viene chiamata *dominio*. All'interno di ogni *dominio* può essere presente un numero arbitrario di agenti; non ci sono limiti imposti oltre i quali si deve effettuare una scissione.

L'organizzazione degli agenti in domini ha ripercussioni positive sulla scalabilità. Buona parte di informazioni, che altrimenti sarebbero state inserite

all'interno del database globale, possono ora essere rese condivise solo all'interno del dominio, nascondendo di fatto la loro presenza ad agenti esterni. Se questo a prima vista può sembrare una limitazione, incide invece in maniera fondamentale sulla scalabilità dell'architettura. Le richieste di monitoraggio devono ora seguire un tragitto più lungo, non potendo essere spedite direttamente a destinazione, ma il database globale è scaricato della necessità di avere informazioni anche su NME e CdM. Nel complesso quindi, i benefici generati dalla migliore scalabilità sorpassano di gran lunga la maggiore lunghezza dei cammini.

La località comune fra NMA e NME di una solita partizione può essere sfruttata per effettuare decisioni di ottimizzazione delle risorse o di bilanciamento del carico. Per esempio si può ritardare l'invio di alcune richieste nel caso in cui ci siano risorse sovraccariche, o passarle a diversi NME meno carichi. Un altro vantaggio è la possibilità, da parte degli NMA responsabili del dominio, di effettuare caching dei risultati di monitoraggio; questi potranno essere successivamente riusati in caso si presentino richieste uguali a quelle appena servite. L'introduzione del caching consente di alleggerire l'impatto sugli NME e sull'infrastruttura di rete, eliminando la necessità di ripetere nuovamente le stesse misurazioni.

Risulta chiaro come nel database globale vengano quindi inserite soltanto le informazioni riguardanti gli NMA, ossia di quelle entità che hanno il

compito esclusivo del recapito delle richieste *fra* domini diversi. Viene quindi lasciata all'organizzazione interna del dominio il compito di effettuare lo smistamento finale verso client o NME. L'uso di un database interno al dominio permette di rivelare solo le informazioni necessarie (ossia le informazioni sugli NMA) a tutta la rete di monitoraggio, a vantaggio di sicurezza ed manutenibilità.

3.4 IL Database globale

Il database globale, condiviso fra tutti gli NMA che partecipano al monitoraggio, rappresenta il collante necessario a tenere insieme tutta l'infrastruttura. Senza una struttura condivisa che metta a disposizione di tutti gli agenti le informazioni necessarie per contattare ed identificare gli altri partecipanti, il monitoraggio *on demand* non risulterebbe possibile. Essendo l'elemento più sensibile di tutta l'architettura, una cura particolare deve essere prestata nella sua realizzazione, sia nella scelte progettuali che implementative.

Varie sono le possibilità di implementazione; una che riteniamo si adatti bene alle necessità dell'infrastruttura risulta essere quella di un database distribuito fra più macchine all'interno della rete di monitoraggio: a seconda del numero di macchine fra cui verrà ripartito si avranno impatti maggiori o minori sulle latenze di comunicazione e sul carico che ognuna dovrà sostene-

re. Allo stato attuale, maggiori indagini sono necessarie per determinare le modalità di partizionamento del database e quali siano gli aspetti più importanti da considerare nella sua implementazione. Un aspetto che sicuramente ha una valenza fondamentale è l'affidabilità; questa deve essere garantita nel maggior numero di casi possibili, pena il blocco parziale (o anche totale) dell'intera struttura di monitoraggio.

Come accennato sopra, all'interno del database globale si trovano le informazioni necessarie all'identificazione ed indirizzamento di tutti gli NMA, ossia quelli che compongono l'ossatura della struttura di monitoraggio. Si può ipotizzare quindi che un generico record contenga il nome dell'agente (utilizzabile come chiave primaria in quanto unico all'interno di tutta la struttura), il nome del dominio di cui è responsabile, il suo indirizzo di rete per potere essere contattato, e la chiave pubblica che ne consente l'identificazione.

Il database globale viene interrogato da un generico NMA quando questo si accorge di non avere informazioni sufficienti riguardo altri NMA, per esempio quando necessita la chiave pubblica o l'indirizzo di rete. Si può ipotizzare che gli NMA salvino le informazioni ricevute dal database in una cache locale da usare in caso di successiva necessità, questo consente di limitare al minimo l'accesso alla struttura condivisa, struttura molto sollecitata. L'aggiunta di una cache locale in ogni NMA comporta però una nuova serie di problemi relativi alla validità dei dati locali che richiede un ulteriore studio.

In prima approssimazione si può considerare che i dati prelevati dal database globale abbiano una validità locale fissa, dopo la quale necessitano di un aggiornamento.

Il caricamento dei dati riguardanti NMA nel database globale avviene manualmente da parte di soggetti autorizzati (gli amministratori del database). Inserimenti e rimozioni di dati nel database condiviso da parte degli stessi agenti sarebbero possibili, ma questo introdurrebbe una serie di problematiche che allo stato attuale non si ritiene opportuno trattare.

In situazioni simili a quelle sopra esposte, normalmente problemi analoghi sono risolti mediante l'uso di server DNS; si è pensato invece di usare server LDAP per una prima implementazione del database distribuito. Questa scelta è stata fatta per avere maggiore possibilità di espansione in futuro in quanto, l'introduzione di nuove tipologie di informazioni nel database condiviso sarebbe risultata molto difficile usando dei server DNS, mentre è molto facile con quelli di tipo LDAP. Visto che lo studio su questa architettura è ancora ad uno stadio iniziale, si è preferito lasciare un più ampio margine per sviluppi futuri, invece di fare delle assunzioni che poi si sarebbero potute rivelare stringenti in uno stadio avanzato.

3.4.1 Il database locale

Nel database locale trovano posto i dati riguardanti non solo NMA del *dominio*, ma anche di NME e Client di Monitoraggio. Questo deriva dal fatto che anche nei casi di comunicazioni fra CdM/NME con gli NMA c'è l'esigenza di avere a disposizione le informazioni di identificazione ed indirizzamento.

L'unica differenza che si riscontra fra il database globale e quello locale, è la presenza fra i dati mantenuti di quello relativo alla tipologia di agente che ha effettuato una registrazione (NMA, NME e CdM). Vista la presenza di tipi diversi di agenti che afferiscono al database locale, è necessario sapere che tipo di agente riguarda una specifica registrazione, fattore necessario nello smistamento delle richieste e dei risultati.

3.5 Il routing delle richieste e dei risultati

Il routing delle richieste e dei risultati verso le destinazioni finali avviene con l'accesso alle informazioni contenute nei database locali e globali. Essendo il modello di infrastruttura fondamentalmente piatto, dove cioè tutti gli agenti sono sullo stesso livello, il routing risulta essere molto semplice. Tutte le informazioni necessarie allo smistamento di richieste e flussi di misurazioni possono essere ricavate dal database condiviso.

Un generico NMA per decidere la prossima destinazione di una richiesta

di monitoraggio, per esempio, ha bisogno soltanto di leggere le informazioni contenuta nella richiesta: qui è presente il nome del dominio destinazione, ed interrogando il database globale può trovare l'indirizzo dell'agente a cui questa dovrà essere spedita.

Questo modello di routing è valido perché siamo in presenza di una organizzazione molto semplice degli agenti: nel caso in cui si passasse ad una organizzazione gerarchica, o comunque più complessa, sarebbero sicuramente necessarie delle variazioni significative, fino ad arrivare anche a routing con una complessità tipica di Internet nei casi più articolati.

3.6 Il *soft state* dell'agente

Ogni NMA presente nella rete di monitoraggio tiene traccia, in un *soft state* interno, di tutte le richieste che vede passare. Quando una *richiesta di monitoraggio* viene spedita da un CdM, questa attraversa una serie di NMA per arrivare infine ad un NME che si occuperà di effettuare le rilevazioni. Da qui partirà poi un flusso di messaggi, contenenti le misurazioni, verso l'NMA che gli ha passato la richiesta, e da qui indietro fino al CdM.

L'uso di un *soft state* è necessario in quanto, le misurazioni sono spedite in messaggi molto piccoli, per evitare un impatto eccessivo sulla rete, dove il solo identificatore univoco della richiesta è usabile ai fini di routing. Questo

comporta che l'unico modo da parte degli NMA di smistare un pacchetto partendo dall'identificatore unico di richiesta, è tenendo traccia della richiesta che ha visto passare in precedenza. Quando l'NMA riceve un pacchetto contenente misurazioni interroga il *soft state*, ricavando quindi le informazioni necessarie al routing.

L'inclusione all'interno di ogni pacchetto di risposta di tutta la *richiesta di monitoraggio* o anche solo del percorso da seguire, oltre ad essere uno spreco di risorse, vanificherebbe tutti quegli sforzi volti a minimizzare l'impatto del trasferimento dei risultati sulla rete, di fatto limitando la scalabilità dell'infrastruttura.

Si può presupporre che nel *soft state* dell'agente vengano salvati soltanto quei dati necessari ai fini del successivo routing, ossia l'identificatore unico della richiesta, il nome dell'agente a cui è stata spedita la richiesta, e il nome dell'agente al quale spedire eventuali risultati.

3.7 L'architettura di un NMA

In figura 3.1 si può vedere l'architettura di un Network Monitoring Agent.

Spieghiamo ora più nel dettaglio le parti principali che compongono il Network Monitoring Agent, e le interazioni che intercorrono fra loro.

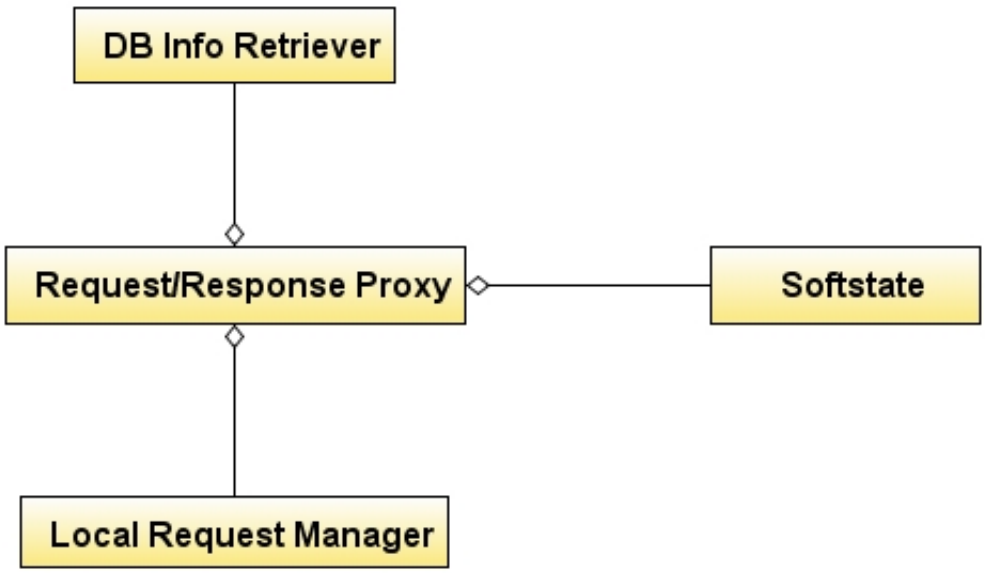

Figura 3.1: L'architettura interna di un agente

3.7.1 Request/Response Proxy

Il compito di questo componente è quello di ricevere le richieste di monitoraggio, da CdM o da altri NMA, e smistarli verso le loro destinazioni finali. Quando una richiesta di monitoraggio viene ricevuta, viene controllato se risulta valida e se è possibile stabilire l'identità dell'agente che l'ha spedita. In caso non risulti possibile, o la richiesta non risulti valida, questa viene scartata.

Se la richiesta passa il controllo, si cerca di determinare se debba essere spedita ad un NMA remoto, e si determina quale. Si procede quindi a firmare la richiesta con la chiave privata dell'agente e la si spedisce. Nel caso in cui la richiesta debba essere spedita ad un NME del dominio, in quanto locale, la si passa al *Local Request Manager* che si occuperà di passarla all'NME responsabile. Per tutte le richieste sulle quali viene effettuato routing o che vengono passate al *NMEManager*, se ne aggiunge traccia all'interno del *ConnectionState*.

Ulteriore compito di questo componente è quello di effettuare routing sui risultati del monitoraggio provenienti dagli NME del dominio e dagli altri NMA. Per portare a termine questo compito, come spiegato sopra, è necessario accedere al *ConnectionState*, con il quale si dialoga per ricevere tutte le informazioni che precedentemente avevamo inserito.

Sia in questo caso che nel caso del routing delle richieste, le informazioni riguardanti autenticazione ed indirizzamento sono raccolte attraverso il *NMA_DB*. Mediante questo si può comunicare, in maniera trasparente, con i database (globale o locale) per ottenere le informazioni necessarie.

3.7.2 ConnectionState

Come ampiamente spiegato in precedenza questo componente implementa il soft state di un NMA. Sia l'aggiunta che la richiesta di informazioni è effettuata dal *Request/Response Proxy*, l'inserzione avviene quando questo riceve una nuova richiesta di monitoraggio; la richiesta di informazioni invece quando arrivano dei flussi di dati e si deve effettuare il routing.

Il *ConnectionState* è in grado di interpretare la parte riguardante orario di inizio e durata delle richieste di monitoraggio e quindi si occupa di effettuarne la rimozione in maniera automatica quando queste diventano obsolete.

3.7.3 NMA_DB

Il *NMA_DB* permette all'NMA di interfacciarsi con il database globale e locale, per ritrovare quelle informazioni necessarie all'indirizzamento ed al routing. L'accesso avviene in modo trasparente così da scaricare la conoscenza delle modalità di accesso. Modalità che possono essere anche diverse fra lo-

ro, e di cui tutte le altre parti dell'agente non devono essere a conoscenza, limitandosi all'uso di una interfaccia unica.

3.7.4 NMEManager

L'*NMEManager* si occupa di trasferire le richieste di monitoraggio agli NME che devono portare a termine il monitoraggio. Il lavoro effettivamente svolto da questo componente è quello di ricevere le richieste dal *Request/Response Proxy*, che determina quali siano locali e quali no, e farle pervenire all'NME che può effettuare il monitoraggio.

L'*NMEManager* ha a disposizione una serie di componenti, ognuno con la capacità di trasferire la richiesta ad un tipo di NME diverso. Questi componenti sono integrati in maniera dinamica all'interno dell'*NMEManager*, ed al momento del caricamento lo informano della tipologia di strumenti che sono in grado di contattare. Nella richiesta viene specificato quale tipo di monitoraggio si chiede di effettuare: incrociando l'informazione reperita dai componenti con quella presente nella richiesta si riesce ad individuare il componente adatto da utilizzare.

L'effettivo trasferimento della richiesta all'NME avviene poi attraverso l'uso del componente che si è individuato: le modalità sono lasciate a completa libertà dello sviluppatore.

Capitolo 4

Implementazione di un agente per il trasferimento di misurazioni di rete

Dopo aver definito il progetto di una applicazione per il trasferimento di misurazioni di rete, si è pensato di realizzare un prototipo funzionante di un Network Monitoring Agent. Lo sviluppo di questo agente avrebbe permesso sia di dimostrare l'applicabilità dell'infrastruttura di monitoraggio *on demand* a casi reali di utilizzo che di essere usato come base per successive ricerche.

Insieme all'agente è stata realizzata una struttura di test che consente

di riprodurre il funzionamento di una rete di monitoraggio *on demand*. Si è implementato inoltre un piccolo NME per poter effettuare misurazioni di tipo Ping, insieme con un client di monitoraggio per la spedizione delle richieste.

Per la realizzazione di tutte le entità software, ci si è avvalsi del linguaggio di programmazione Java. La scelta è ricaduta su questo linguaggio in quanto la portabilità degli agenti era fattore di primaria importanza in un ambiente intrinsecamente distribuito come quello del monitoraggio. Inoltre la capacità di Java di caricare codice on demand, permettendo così una semplice realizzazione di plugin, è stato un fattore che ha influito sulla scelta.

Si va ora a spiegare le parti che necessitano una maggiore definizione sulle modalità realizzative e sulle tecnologie impiegate per l'implementazione.

4.1 I protocolli di comunicazione

All'interno dell'infrastruttura hanno luogo diversi tipi di comunicazione con partner remoti ed ognuna richiede l'uso di un protocollo comune. Si possono individuare tre tipi diversi di comunicazioni all'interno dell'infrastruttura:

- Il trasferimento delle richieste di monitoraggio.
- Il trasferimento delle misurazioni risultanti dal monitoraggio.
- Il trasferimento delle richieste da un NMA ad un NME

Per ognuno di questi tipi si è quindi dovuto studiare il problema e definire un protocollo di comunicazione che fosse il più adeguato possibile alle esigenze dell'infrastruttura. Un fattore che ha influito in maniera determinante sulle scelte dei protocolli è stato l'overhead che questo comportava per ogni singolo trasferimento.

4.1.1 Protocollo per il trasferimento delle richieste

Si è visto che il protocollo per il trasferimento delle richieste sarebbe stato impiegato per il passaggio delle richieste di monitoraggio fra Client di Monitoraggio ed NMA, e fra NMA e NMA. Nella scelta dello specifico protocollo si è tenuto presente che in questo si doveva trovar spazio per poter definire l'identità del mittente: la scelta è caduta sul protocollo SOAP.

SOAP (Simple Object Access Protocol) è un protocollo per lo scambio di messaggi XML su reti di computer, sviluppato da Dave Winer, Don Box, Bob Atkinson, and Mohsen Al-Ghosein nel 1998, ed ora standard del W3C Consortium. Il protocollo SOAP fa uso dell'internet application layer (HTTP,HTTPS, SMTP, ecc) per trasferire richieste a metodi remoti come se fossero locali mediante l'uso di una sintassi XML per definire nome della funzione da richiamare, tipo dei parametri e dei risultati.

La richiesta di monitoraggio. prima di essere spedita, è avvolta all'interno

di un aggiuntivo frame XML per il trasferimento: questo frame aggiuntivo è necessario per poter includere la firma dell'agente che effettua la spedizione, e quindi poter determinare l'identità del mittente. La firma è generata, utilizzando la chiave privata del mittente a partire dalla richiesta di monitoraggio stessa e poi codificata mediante l'uso della notazione posizionale Base64.

La scelta è ricaduta su SOAP in quanto si adatta perfettamente allo scenario di utilizzo all'interno dell'infrastruttura, ossia invocare operazioni remote in maniera affidabile. Altri protocolli sarebbero stati adatti allo scopo, fra cui la scrittura di uno ad hoc, ma si è ritenuto di usare SOAP in quanto consentiva l'utilizzo di una ampia base di librerie già testate e funzionanti, gestendo in maniera autonoma tutta la fase di trasporto ed identificazione del metodo da invocare.

4.1.2 Protocollo per il trasferimento delle misurazioni

Per il protocollo di trasferimento delle misurazioni le caratteristiche che sono state considerate necessarie sono state il minimo overhead possibile e la possibilità di verificare l'identità del mittente, come nel caso del protocollo di trasferimento delle richieste, ma direttamente a livello di trasporto.

L'uso di un protocollo con basso overhead per pacchetto è necessario in quanto le misurazioni sono spedite non appena effettuate, senza accorpare

più misurazioni in un pacchetto. Se si fosse usato un protocollo che non teneva conto di questo fattore, l'impatto che il trasferimento delle misurazioni avrebbe avuto sull'infrastruttura di rete sarebbe stato notevole, limitando fortemente la scalabilità dell'infrastruttura. Si pensi per esempio all'uso del protocollo SOAP, che richiede un frame XML, solo per trasferire una misurazione, si sarebbe potuti arrivare ad un rapporto 20 a 1, fra peso del protocollo e carico utile, rendendo quindi il protocollo la parte più pesante del trasferimento.

Il secondo requisito, quello di dare verificabilità del mittente a livello di trasporto, deriva dal fatto che il flusso di misurazioni è generato dagli NME in piena libertà. I dati inseriti all'interno di un messaggio possono essere scritti con qualsiasi sintassi, di fatto impedendo una autenticazione a livello applicazione. Da questo è seguito lo spostamento del problema dell'identificazione del mittente a livello trasporto. La mancata consegna di un pacchetto o l'arrivo fuori ordine non risultano un problema di forte impatto sul tipo di dati trasportato, e quindi non sono stati considerati nella scelta del protocollo.

Visto che un protocollo del tipo richiesto non esisteva, si è pensato di crearne uno molto semplice che avesse tutte le caratteristiche richieste. Il protocollo è stato implementato su UDP, cercando quindi di diminuire il più possibile l'impatto degli header, e di eventuali messaggi di controllo spediti

sulla rete. UDP consente di spedire messaggi dove soltanto 28 byte non sono carico utile (20 byte per l'header ip e 8 per l'header UDP), raggiungendo quindi un'alta efficienza.

Un pacchetto per il trasferimento di misurazioni è composto da 4 campi:

- Session Name Length [2 byte]: Indica la lunghezza (in byte) del campo *Session Name*.
- Session Name [lunghezza variabile]: Il nome della sessione a cui appartiene il pacchetto, permette di definire pacchetti provenienti dalla stessa sessione di monitoraggio ai fini di identificazione e routing.
- Payload [lunghezza variabile]: I dati che vogliono essere trasferiti fra chi esegue le misurazioni e chi le vuole ricevere.
- Packet Signature [lunghezza variabile]: La firma del pacchetto eseguita usando la chiave privata dell'host che spedisce i pacchetti (quindi ricalcolata ad ogni host prima della spedizione) calcolata sul solo campo *Payload*. La lunghezza di questo campo è variabile in quanto dipende dalla lunghezza delle chiavi utilizzate per generare la firma, ad una maggiore lunghezza corrisponde una maggiore sicurezza, ma anche un maggiore uso di risorse.

Nel prototipo che è stato realizzato, per esempio, nel campo *Session Name*

è inserito il valore dell'attributo SessionId ripreso dalla richiesta di monitoraggio, ed il campo *Signature* può essere definito nella configurazione degli agenti, ed ha un valore di 64 byte per chiavi a 512 bit.

4.1.3 Protocollo per passaggio richieste agli NME

Come spiegato nella parte del progetto dell'architettura, il passaggio delle richiesta fra NMA ed NME, avviene attraverso plugin caricati in maniera dinamica. Questo consente la massima espandibilità e lascia totale libertà di decisione sulle modalità da usare per il trasferimento della richiesta all'NME designato. Si può prevedere l'uso di file condivisi, messaggi inter processo, protocolli di rete, o qualsiasi altra tipologia risulti adatta a comunicare con l'NME.

4.2 Il Database condiviso

La scelta per la realizzazione di questo componente fondamentale è ricaduta su un server LDAP accessibile da tutti gli agenti.

LDAP (Lightweight Directory Access Protocol) è un protocollo per interrogare e modificare un directory service, dove per directory si intende un insieme di oggetti con attributi comuni organizzati in maniera gerarchica. Sviluppato sulla base del vecchio protocollo X500, è stato pensato per po-

ter essere usato su reti TCP, e non solo OSI, come invece prevedeva X500. Sfruttando l'organizzazione gerarchica con la quale le informazioni sono organizzate all'interno dei server LDAP, risulta molto semplice implementare un database distribuito su più host diversi per favorirne scalabilità ed affidabilità.

L'ampia affidabilità che questo tipo di soluzioni ha ormai maturato nel corso di anni di sviluppo unita alla capacità di organizzare dati di tipologie diverse ne ha fatta la soluzione ideale per essere usato nella infrastruttura di test. Maggiori indagini sono comunque necessarie per l'utilizzo di questa soluzione in scenari reali di utilizzo, dove la gestione del database distribuito potrebbe essere molto complessa.

Una differenza da segnalarsi rispetto al progetto dell'architettura riguarda la presenza di un solo database per implementare sia quello globale che tutti quelli interni ad ogni dominio. Questa scelta è stata effettuata per favorire la fase di deployment, dove l'attivazione di un server diverso per ogni dominio avrebbe comportato diversi problemi; essendo solo in uno stadio di test dell'infrastruttura, l'aver accesso ad informazioni sugli agenti di altri domini non comporta problemi di sicurezza.

4.3 La tecnologia usata

Oltre al già citato Java (in versione 1.6) usato per la realizzazione dell'NMA, si è utilizzato anche per lo sviluppo di un plugin per l'architettura (spiegato in dettagli più avanti), e per la realizzazione di un client per spedire richieste di monitoraggio.

Internamente, per la realizzazione della parte di trasferimento di richieste di monitoraggio fra agenti, si è usato la libreria Axis2 [24], sviluppata dalla Apache Foundation: questa implementa lo stack SOAP, ed un piccolo web server per ricevere le richieste, utilizzando il linguaggio Java. Come mezzo per il trasferimento dei messaggi SOAP si era da prima scelto TCP, in quanto introduceva un bassissimo overhead di comunicazione, ma a causa di problemi legati alla versione attuale di Axis si è dovuti ripiegare su HTTP come mezzo di trasporto. L'uso di HTTP non incide comunque in maniera determinante sul consumo di risorse per il trasferimento di una richiesta: quest'ultima risulta essere ancora la parte principale. In caso di sviluppo di una architettura per usi non di ricerca, l'utilizzo di un'altra libreria o l'integrazione del supporto TCP sono possibilità da tenere in considerazione.

Per il server LDAP, si è scelto di usare OpenLDAP, una implementazione open source dello standard LDAP. Oltre al server, ed alle utility da linea di comandi per effettuare le inizializzazione, si sono usate anche una serie

di classi per l'accesso al database direttamente in Java. Il server usato è la versione 2.3.30, ed il protocollo usato per le comunicazioni fra agenti ed il server è la versione 3.

4.4 Le classi

In figura 4.1 si possono vedere le classi principali (sono state tralasciate le classi di utilità) che implementano l'NMA.

ConnectionState implementa l'accesso al soft state dell'agente, l'*NMEManager* si occupa del trasferimento delle richieste agli NME, così come *NMA_DB* implementa l'accesso al database condiviso, in maniera del tutto conforme a come definito in precedenza. L'unica differenza risulta esserci nell'implementazione di quello che nel progetto era il Request/Response Proxy, componente scisso in tre parti distinte: la prima si occupa di ricevere le richieste (SOAPRequestDispatcher), la seconda di farne routing (RequestManager), e l'ultima di effettuare routing dei risultati (StreamProxy). Questa divisione è stata effettuata solo per motivi di separazione di compiti a livello realizzativo, e non cambia quello che sarebbe, nel suo insieme, il *Local Request Manager*.

Rispetto al progetto, l'unico componente che necessità di qualche spiegazione aggiuntiva è l'*NMEManager*: questo è stato realizzato utilizzando una caratteristica particolare del linguaggio Java, ossia il caricamento dina-

mico delle classi. Nel momento in cui l'*NMEManager* riceve una richiesta di monitoraggio da trasferire ad un NME, controlla una cache interna per il costruttore di un componente in grado di gestire il tipo di passaggio specificato nella richiesta. In caso positivo, crea un'istanza della classe ed, utilizzando un'interfaccia ben definita, il trasferimento all'NME si realizza. Nel caso in cui l'*NMEManager* non trovi un costruttore adatto, controlla una directory locale in cui può trovare tutti i componenti a sua disposizione, alla ricerca di uno in grado di realizzare il trasferimento. Un componente, per poter essere riconosciuto come tale, deve avere due requisiti fondamentali: deve implementare l'interfaccia *NMEPlugin* e deve avere dei costruttori con una signature adeguata. Opzionalmente è possibile inserire sempre nella stessa directory un file da caricare contestualmente al componente per specificare dei parametri necessari alla configurazione.

La possibilità di caricare le classi a runtime, e di poterne controllare l'aderenza alle interfacce necessarie ad espletare determinati compiti è risultata essere una caratteristica fondamentale per lo sviluppo di plugin per questa architettura, confermando così la scelta di Java, come linguaggio per la realizzazione del prototipo.

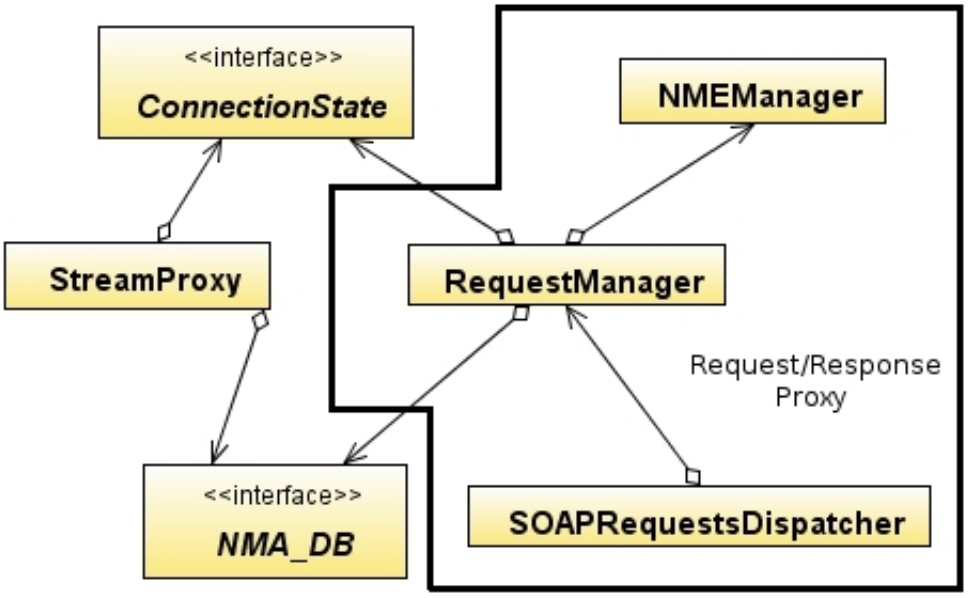

Figura 4.1: Le classi dell'NMA.

4.5 Il test dell'infrastruttura

Per effettuare il test del prototipo, è stato necessario sviluppare un'ambiente specifico, e dei componenti che permettessero di effettuare richieste e misurazioni.

Essendo l'NMA la parte centrale di una infrastruttura di monitoraggio, è stato necessario sviluppare un client, un piccolo NME, ed una infrastruttura di monitoraggio.

4.5.1 Il client

Quello che è stato realizzato è un client, con una doppia interfaccia, sia grafica che su console, che permette l'invio di richieste di monitoraggio, create in maniera personalizzata.

L'interfaccia grafica , vedi figura 4.2, permette di specificare tutte le opzioni necessarie ad iniziare un monitoraggio. Quelle comuni a tutti i tipi hanno un campo distinto per ognuna, mentre quelle specifiche, dipendenti dallo strumento da utilizzare, hanno un campo unico. In questo campo i nomi dei parametri dovranno essere specificati in accordo alla sintassi attesa dallo strumento. L'interfaccia mediante console, accetta invece in input un file contenente una richiesta che viene considerata valida, su cui quindi non viene effettuato nessun tipo test, limitandosi a firmarla e spedirla.

Il client dopo aver spedito la richiesta si mette in attesa di ricevere i dati relativi al monitoraggio, mostrandoli in console o sull'interfaccia grafica non appena arrivano.

4.5.2 Il plugin del ping

Per poter effettuare dei test era necessario sviluppare almeno un NME con relativo plugin necessario al trasferimento delle richieste. Si sarebbe così potuta osservare la fase di routing delle richieste e dei risultati: si è pensato

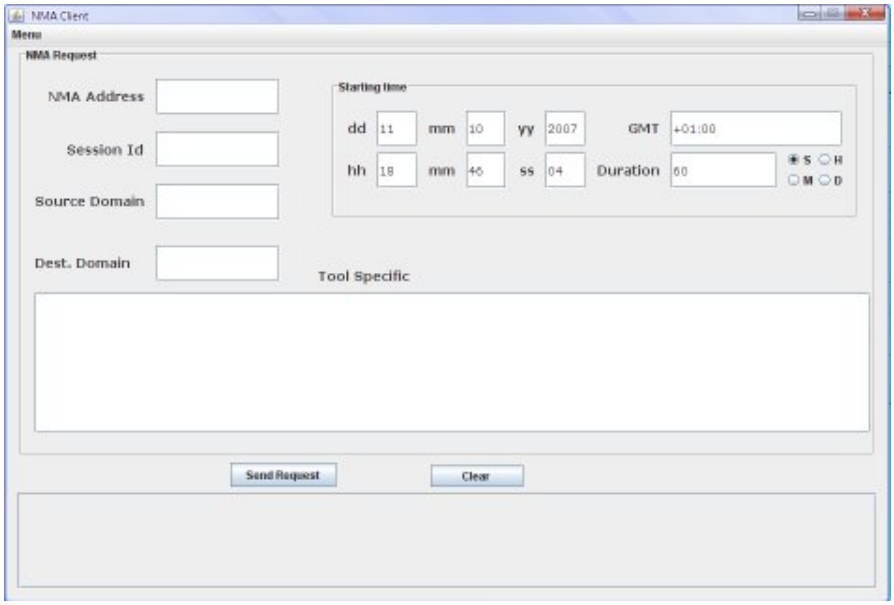

Figura 4.2: L'interfaccia grafica del client

quindi di realizzare un semplice NME che effettuasse ping su una macchina remota.

Vista l'estrema semplicità, e visto il basso uso di risorse necessarie in fase operativa, si è pensato di integrare sia il componente per il trasferimento delle richieste che l'NME all'interno della stessa classe; questo ha consentito un deployment unico. In casi reali, dove è necessario un uso di risorse maggiore, oppure quando l'NME deve risiedere su host remoti (per esempio effettuare misurazioni su un'interfaccia), lo sviluppo di due entità distinte è una necessità e non solo un'opzione. Essendo componente ed NME integrati all'interno della solita classe viene escluso dal test il passaggio della richie-

sta fra i due: questo non risulta essere un punto interessante da studiare in quanto ogni plugin può implementarlo in maniera arbitraria (e quindi anche implementare l'NME all'interno del plugin).

Il plugin è sviluppato per far uso del comando ping presente su tutti i sistemi operativi. Non appena arriva una richiesta si leggono i parametri significativi, si controlla il tipo di sistema operativo sul quale si è ospiti, e si richiama l'uso del ping specifico del sistema. Sui risultati ritornati dal programma viene effettuato il parsing (dipendente dal ping usato) per individuare i valori caratteristici, questi sono poi inseriti in una struttura XML ben definita, e spediti. I parametri specifici che possono essere inseriti nella richiesta sono l'indirizzo dell'host verso cui effettuare ping e la frequenza di campionamento.

4.5.3 L'ambiente di test

Per lo sviluppo dell'ambiente di test si è dovuto tener conto del fatto che essendo questo prototipo, sviluppato all'interno del progetto Coregrid, c'era la necessità di renderlo facilmente distribuibile; inoltre la fase di configurazione manuale richiesta doveva essere minima in quanto rivolto principalmente ad altri ricercatori per effettuare test sull'architettura.

L'uso di macchine reali aperte al pubblico non era una soluzione attuabile,

quindi la scelta è ricaduta sullo sviluppo di un ambiente di test mediante l'uso di macchine virtuali. Le macchine virtuali hanno come vantaggio la facilità con le quali possono essere replicate e distribuite; inoltre la totale sicurezza, condizione irraggiungibile nel caso in cui si rendendo disponibili macchine fisiche, ne hanno fatto la scelta ideale.

Netkit

Visto che si doveva realizzare una rete non banale per i test, la scelta è ricaduta su Netkit [9]. Direttamente dal sito del progetto Netkit: Netkit è stato concepito come un ambiente per preparare ed effettuare esperimenti di rete a basso costo e con pochi sforzi. Permette di creare molteplici dispositivi virtuali di rete che possono essere interconnessi in modo da formare una rete su un singolo PC.

Come si può intuire dalla presentazione del progetto, Netkit ha tutte le caratteristiche che servono per sviluppare un ambiente di test adatto. Permette di virtualizzare una rete (anche complessa) all'interno di una macchina, è facilmente pacchettizzabile, e non ha problemi di licenza essendo open source (elemento molto utile in fase di distribuzione).

Come spiegato in [19], Netkit consente l'attivazione di una macchina virtuale per ogni apparecchiatura di rete che si vuole virtualizzare. Dà la possibilità di specificare, mediante parametri, in che modo queste apparecchiature

siano interconnesse fra loro, creando di fatto una rete di computer all'interno di uno solo.

Netkit basa il suo funzionamento sull'uso di UML (User Mode Linux) [5, 13, 12] un port del kernel di Linux progettato per funzionare come un processo in user-space. Essendo UML un kernel, completo di tutti i sottosistemi (gestione della memoria, file system, scheduler, ecc), che gira in user space è possibile attivare un numero indefinito (dipendente soltanto dalle risorse e dal sistema operativo host) di macchine virtuali, ognuna con le proprie caratteristiche.

Netkit usa questa questa capacità per attivare una macchina virtuale (e quindi un processo) per ogni apparecchiatura da implementare (sia essa un pc o un router o uno switch) nella rete virtuale. Le interconnessioni sono implementate, anch'esse, come un processo in user space che si occupa di passare i pacchetti che arrivano da un'interfaccia di una macchina virtuale a tutte le altre macchine che risultano connesse su quella specifica sottorete virtuale.

Testbed

Il testbed è un pacchetto unico, utilizzabile in un sistema operativo Linux Debian-Like. E' possibile creare in maniera automatica un ambiente pronto per effettuare prove sull'infrastruttura di monitoraggio o per lo sviluppo di

plugin per NMA.

Un insieme di script permette la creazione di una rete di macchine virtuali, il deployment degli NMA all'interno di ogni host virtuale, insieme con client e plugin dove necessario. Viene eseguita inoltre l'installazione di un server OpenLDAP, e viene preparato l'ambiente per l'esecuzione dell'infrastruttura di monitoraggio.

Con una struttura pacchettizzata, quindi di facile distribuzione, e con una serie di script che consentono il deployment (procedimento che eseguito manualmente è abbastanza complesso), risulta molto semplice usare il testbed per eseguire test su l'architettura. In caso si volesse cambiare l'architettura o il deployment di agenti all'interno della rete virtuale, per renderla più adatta ad esigenze specifiche, le modifiche da effettuare sono molto semplici, consentendo quindi una elevata personalizzabilità del testbed.

La rete

In figura 4.3 si può vedere la rete che viene virtualizzata all'interno del testbed. Sono presenti tre pc(Ares, Apollo,Zeus), e 2 router (R1, R2) che interconnettono i pc. Sui ognuno dei tre pc viene installato un NMA, con annesso NME per effettuare il ping, sull'host Zeus inoltre è installato un client che permette di eseguire richieste e ricevere i risultati.

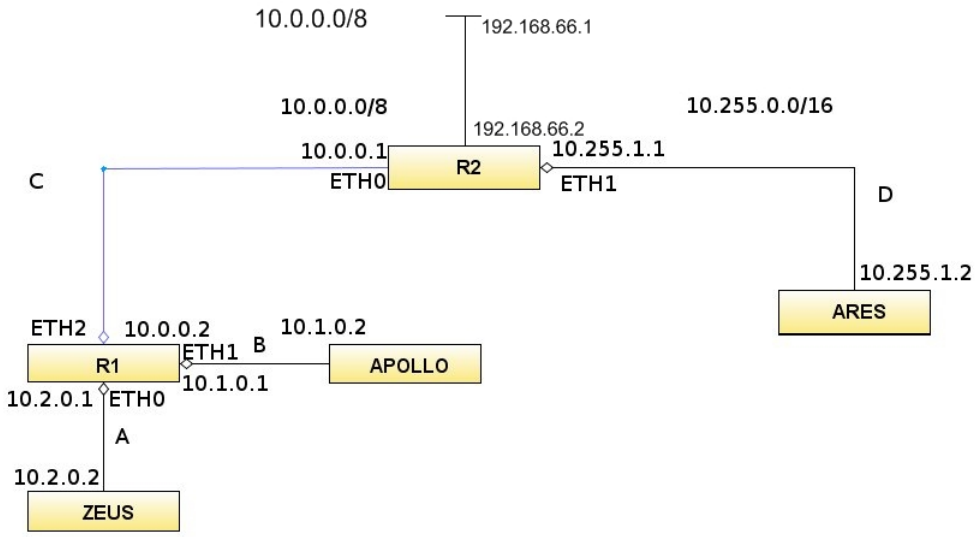

Figura 4.3: La rete virtuale implementata nel testbed

4.5.4 I risultati

La struttura di test ha mostrato una buona usabilità, sia in fase di installazione che in fase di utilizzo. Si è riusciti ad eseguire test approfonditi su macchine virtuali diverse che hanno consentito l'individuazione di bug nel prototipo e nei plugin. L'uso di un ambiente uniforme per i test ha facilitato enormemente il processo di individuazione e risoluzione dei bug, in quanto facilmente replicabili, anche su macchine remote.

Per quanto riguarda l'NMA non sono stati effettuati test di performance, in quanto allo stato attuale, l'obbiettivo principale era arrivare ad uno sviluppo che ne consentisse l'utilizzo in ambienti di ricerca; obbiettivo raggiunto in quanto l'infrastruttura è risultata stabile nel suo estensivo test.

In un secondo stadio si prevedono dei test specifici per valutare le performance di un NMA (quanti pacchetti viene fatto routing al secondo, impatto del numero di sessioni contemporanee sulle performance, ecc). Test che risultano necessari per valutare la scalabilità dell'architettura su reti con migliaia di domini, con la presenza di decine e decine di flussi di monitoraggio attivi in contemporanea, dove quindi le performance sono una necessità.

Capitolo 5

Conclusioni e sviluppi futuri

In ambienti distribuiti, dove le interconnessioni assumono un ruolo fondamentale, sopra il quale si costruisce l'intera architettura del sistema, il monitoraggio di rete risulta uno strumento da cui non si può prescindere, sia nell'individuazione delle problematiche che nel supporto alle decisioni.

La tesi ha argomentato i limiti del monitoraggio programmato, monitoraggio che non si adatta ad ambienti dinamici. In ambienti, dove c'è una forte esigenza di personalizzazione nei parametri di monitoraggio, così come necessità di tempo reale nella ricezione dei risultati, un monitoraggio di tipo *on demand* è sicuramente più adatto. Il monitoraggio *on demand* risolve i problemi ascritti a quello programmato, consentendo quindi una migliore integrazione sia in ambienti statici che in quelli dinamici.

Sono state sviluppate le problematiche legate al monitoraggio di tipo *on*

demand, mostrando quali sono le possibili soluzioni, e come queste vadano integrate fra loro. Si è quindi studiato nel dettaglio la creazione di una infrastruttura per il monitoraggio *on demand* individuando quali sono gli attori, le modalità e gli strumenti attraverso i quali questo si può realizzare. Si è passati infine a mostrare l'architettura interna di un *Network Monitoring Agent*, uno degli agenti che sta alla base dell'infrastruttura per il monitoraggio *on demand*, mostrando inoltre le tecnologie usate per lo sviluppo.

Nella tesi si possono individuare alcuni punti che necessiterebbero di un più approfondito studio, per esempio la scalabilità dell'infrastruttura in presenza di un elevato numero di agenti e l'impatto che le singole componenti hanno su questa. Altro fattore che richiede uno studio ulteriore è il database condiviso, nodo fondamentale di cui si deve garantire affidabilità ed efficienza; allo stato attuale è stato realizzato con una soluzione conservativa di cui quindi non si può garantire l'ottimalità. Necessario sarebbe inoltre uno studio per identificare le modalità di generazione e di distribuzione dei nomi di dominio, degli agenti, e delle coppie di chiavi pubblica/privata, modalità che al momento sono state soltanto accennate.

La tesi è quindi riuscita ad individuare i componenti necessari alla realizzazione di una infrastruttura necessaria al monitoraggio *on demand* ed a darne una prima implementazione mediante prototipo, base necessaria ad un successiva integrazione in una architettura complessa come quella di un grid.

Bibliografia

[1] Egee network performance monitoring. http://www.egee-npm.org/.

[2] European research network on foundations, software infrastructures and applications for large scale distributed, grid and peer-to-peer technologies. http://www.coregrid.net.

[3] Forth: Foundation for research and technology-hellas. http://www.forth.gr.

[4] Open grid forum (ogf). http://www.ogf.org.

[5] User-mode linux kernel. http://user-mode-linux.sourceforge.net/.

[6] A.Ciuffoletti, A.Congiusta, G.Jankowski, M.Jankowski, and N.Meyer. Grid infrastructure architecture. a modular approach from coregrid. Technical report, Coregrid, Institute on Grid Information, Resource and Workflow Management Services, august 2007.

[7] S. Andreozzi, D. Antoniades, A. Ciuffoletti, A. Ghiselli, E.P. Markatos, M. Polychronakis, and P. Trimintzios. Issues about the integration of passive and active monitoring for grid networks.

[8] Ruth Aydt, Dan Gunter, Warren Smith, Martin Swany, Valerie Taylor, Brian Tierney, and Rich Wolski. A grid monitoring architecture, July 2001.

[9] G.Di Battista, M.Patrignani, S.Pettini, M.Pizzonia, F.Ricci, M.Rimondini, A.Cecchetti, L.Colitti, F.Mariani, and F.Picard. Netkit, the poor man's system to experiment computer networking. http://www.netkit.org.

[10] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple network management protocol (snmp) rfc 1157. May 1990.

[11] Augusto Ciuffoletti, Antonis Papadogiannakis, and Michalis Polychronakis. Network monitoring session description. Technical report, INFN-CNAF, FORTH, May 2007.

[12] Jeff Dike. A user-mode port of the linux kernel. In *In Proc. 4th Annual Linux Showcase and Conference*, October 2000.

[13] Jeff Dike. *User Mode Linux*. Prentice Hall, May 2006.

[14] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.

[15] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–??, 2001.

[16] Y. Fu, L. Cherkasova, W. Tang, and A. Vahdat. Ete: Passive end-to-end internet service performance monitoring, 2002.

[17] Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile agents: Are they a good idea? Technical report, IBM Research Division, March 1995.

[18] Daniel Kouril, Ludek Matyska, and Michal Prochazka. A robust and efficient mechanism to distribute certificate revocation information using the grid monitoring architecture. In *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, pages 614–619, Washington, DC, USA, 2007. IEEE Computer Society.

[19] M.Rimondini. Emulation of computer networks with netkit. Technical report, Dipartimento di Informatica ed Automazione, Università di

Roma Tre, Gennaio 2007.

[20] K. Obraczka and G.Gheorghiu. The performance of a service for network-aware application. In *In Proc. of 2nd SIGMETRICS Conference on Parallel and Distributed Tools*, August 1998.

[21] P.Arlos, M.Fiedler, and A.Nilsson. A distributed passive measurement infrastructure, 2005.

[22] R.Carter and M.Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical report, Boston University, 1996.

[23] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip: Session initiation protocol.rfc 3261. http://faqs.org/rfcs/rfc3261.html, June 2002.

[24] Axis2 team. Apache axis2 is a core engine for web services. http://ws.apache.org/axis2/.

[25] Panos Trimintzios, Michalis Polychronakis, Antonis Papadogiannakis, Michalis Foukarakis, Evangelos P. Markatos, and Arne Øslebø. DiMAPI: An application programming interface for distributed network monitoring. In *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2006.

[26] R. Wolski. Dynamically forecasting network performance to support dynamic scheduling using the network weather service, 1997.

[27] Rich Wolski, Neil T.Spring, and Jim Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing.

[28] W.Stallings. Snmpv3: A security enhancement for snmp, 1998.

[29] Marcia Zangrilli and Bruce B. Lowekamp. Comparing passive network monitoring of grid application traffic with active probes. Technical report, College of William and Mary, 2003.