Sinergia tra tecnologie Multi-Agente ed ambienti di Grid Computing

Giovanni Novelli

DOTTORATO IN INFORMATICA - XX Ciclo

Dipartimento di Matematica e Informatica
Università degli Studi di Catania, ITALIA

Sommario

Sinergia tra tecnologie Multi-Agente ed ambienti di Grid Computing

Giovanni Novelli

In questa tesi viene descritta una sinergia tra tecnologie multi-agente e di Grid Computing in uno scenario di fornitura di contenuti multimediali con supporto a tasks CPU intensive di online transcoding e attività con requisiti di QoS (Quality of Service) di multimedia streaming. Inoltre è presentata la libreria GAP (Grid Agents Platform) per la simulazione di sinergie tra tecnologie multi-agente e Grid Computing in contesti meno specifici ed i risultati ottenuti facendo uso di tale libreria per la simulazione della soluzione proposta nell'ambito del contesto applicativo specificamente considerato.

Le motivazioni della sinergia tra tali tecnologie sono varie. I middleware di Grid Computing sono generalmente caratterizzati da bassa reattività non supportando in modo nativo applicazioni che invece richiedono un'alta reattività. Le tecnologie multi-agente possono fornire uno strumento per la riduzione delle latenze di rete nell'attività di sottomissione di jobs alla Grid, ponendosi come delegati degli utenti in tale attività. Il deployment di applicazioni multi-utente in un ambiente Grid non è immediato in quanto i sistemi di autenticazione dei middlewares di Grid Computing sono basati su meccanismi che generalmente impongono al singolo utente l'ottenimento di un certificato personale e l'utilizzo di questo per qualsivoglia accesso alle risorse della Grid; per un provider sarebbe auspicabile l'outsourcing su risorse Grid di propri servizi applicativi, con un accoppiamento lasco tra tali applicazioni e il middleware di Grid Computing per il tramite di tecnologie multi-agente, per fornire agli utenti un accesso trasparente a tali ingenti risorse di calcolo e memorizzazione senza che questi siano coinvolti in complessi aspetti amministrativi e tecnici propri dell'accesso ad un ambiente di Grid Computing; il ricorso a tecnologie multi-agente fornisce un paradigma per sfruttare il potenziale di risorse offerte dal Grid Computing delegando la sfida di complessità propria di tali ambienti alle capacità degli agenti di interagire con l'ambiente esterno in modo intelligente e collaborativo al fine di raggiungere i propri obiettivi.

Nello specifico viene affrontato uno scenario di fornitura di contenuti multimediali dove le risorse di un ambiente di Grid Computing, tramite una sinergia a runtime con tecnologie multi-agente, vengono utilizzate per emulare una CDN (Content Delivery Network) e permettere, tramite un'euristica di prossimità, a ciascun utente di ottenere il contenuto richiesto da un Computing Element vicino e ai Computing Element di reperire velocemente i contenuti da Storage Element vicini, realizzando in tal modo uno streaming fluido di tali contenuti, opportunamente transcodificati al volo quando il formato richiesto dall'utente o le condizioni di rete non permettano di servire in modo soddisfacente la richiesta dell'utente.

Lo streaming di contenuti multimediali implica requisiti impliciti di Quality of Service, in particolare requisiti temporali connessi al degrado della qualità percepita dall'utente all'aumentare del ritardo nella trasmissione dei singoli frame. La soluzione proposta sfruttando tale sinergia supporta meccanismi di soft QoS per la selezione di risorse appropriate e per l'adattamento della QoS alle condizioni di carico delle risorse di calcolo, di memorizzazione e di rete. L'adattamento della QoS è ottenuto variando la qualità dello stream tramite attività di online transcoding governate da scelte intelligenti effettuate dagli agenti tramite collaborazione ed individualmente tramite meccanismi di controllo fuzzy.

La soluzione proposta è stata sottoposta a verifiche sperimentali, principalmente basate sulla simulazione tramite la libreria GAP sviluppata nell'ambito della ricerca, mostrandone l'applicabilità allo scenario considerato e la bontà del ricorso alla sinergia proposta rispetto ad un approccio tradizionale di utilizzo della Grid considerando il medesimo scenario.

Inoltre la libreria GAP è servita a simulare i risultati conseguibili dalle tecnologie multi-agente per sfruttare un ambiente di Grid Computing general purpose per emulare soluzioni proprietarie e dedicate quali sono le CDN, grazie alle capacità degli agenti di approcciare la complessità di un ambiente a loro estraneo per ottenere risultati globali, orchestrando in modo collaborativo l'utilizzo delle risorse Grid nell'intento di fornire prestazioni tipiche di una generica CDN aumentata di un'attività CPU intensive quale l'online transcoding.

Sommario

Elenco delle figure

Elenco delle tabelle

Parte I

Introduzione

In questa tesi viene proposta una soluzione architetturale, basata sulla sinergia tra tecnologie multi-agente e di Grid Computing, che fa riferimento a funzionalità proprie di una CDN (Content Delivery Network) per la distribuzione di contenuti multimediali secondo requisiti di Quality of Service e funzionalità di adattamento dei contenuti tramite online transcoding. Inoltre è presentata la libreria GAP (Grid Agents Platform) per la simulazione di sinergie tra tecnologie multi-agente e Grid Computing in contesti meno specifici ed i risultati ottenuti facendo uso di tale libreria per la simulazione della soluzione proposta nell'ambito del contesto applicativo considerato.

La Parte I, organizzata in tre capitoli, fornisce una breve introduzione alle tecnologie considerate nel resto della tesi; nel capitolo 1 vengono sinteticamente presentate le tecnologie di Grid Computing; nel capitolo 2 si introducono le tecnologie multi-agente e il middleware multi-agente JADE; nel capitolo 3 è fornita una breve panoramica sul transcoding.

La Parte II presenta la soluzione proposta ed è organizzata in quattro capitoli; dopo una breve introduzione alla problematica, nel capitolo 4 vengono descritte le motivazioni della sinergia tra tecnologie multi-agente e ambienti di Grid Computing; nel capitolo 5 vengono descritti i servizi della soluzione proposta, l'euristica di prossimità adottata e i meccanismi di adattamento della Quality of Service utilizzati; nel successivo capitolo 6 è trattato il deployment della soluzione ad agenti su Grid.

La Parte III si compone di tre capitoli ed è dedicata alla simulazione con particolare riferimento alla libreria GAP; nel capitolo 8 sono presentati alcuni tools esistenti adatti alla simulazione di ambienti di Grid Computing; nei capitoli 9 e 10 è presentata la libreria GAP e il suo ricorso nella simulazione della soluzione proposta.

La Parte IV si compone di due capitoli; il capitolo 11 presenta un estratto degli esperimenti di transcoding effettuati; il capitolo 12 presenta i risultati delle simulazioni condotte con la libreria GAP.

Capitolo 1

Grid Computing

In relazione ad una serie di fattori concomitanti che si sono manifestati all'inizio degli anni '90, quali la proliferazione su larga scala di Internet dopo l'introduzione del WWW, la riduzione dei costi e l'aumento di prestazioni delle tecnologie di rete, di calcolo e di memorizzazione che hanno permesso di interconnettere geograficamente sistemi distribuiti di calcolo per fronteggiare la crescita pervasiva della domanda di potenza di calcolo in particolare da parte della comunità scientifica, sono emerse le tecnologie di Grid Computing.

Con il termine *Grid Computing* ci si riferisce ad un ampio spettro di soluzioni tecnologiche mirate alla fornitura di capacità computazionale agli utenti indipendentemente dai loro dispositivi di calcolo, dalla loro posizione geografica e dai loro requisiti applicativi. Svariate definizioni sono state coniate per delimitare il significato di tale termine ed identificarne le caratteristiche salienti. Una prima definizione risale al 1998 [1]: *"A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities."*.

Il termine Grid Computing nasce come metafora delle reti energetiche (Power Grid) in una visione in cui la potenza di calcolo sarebbe stata resa accessibile come l'energia elettrica con modalità analoghe di tipo *plug and play* [1, 2]. Il padre unanimemente riconosciuto di questo paradigma è Ian Foster che, considerata l'evoluzione di Internet come strumento di disseminazione dell'informazione su scala globale, per primo ha riconosciuto il potenziale offerto da Internet per interconnettere non solo l'informazione ma la potenza di calcolo lavorando in modo sistematico allo sviluppo di middleware, atti ad

astrarre risorse distribuite geograficamente ed amministrativamente e ad interconnetterle tra loro in modo interoperabile attraverso un approccio basato su standard aperti [3].

Il Grid Computing storicamente nasce in seguito al coinvolgimento di Ian Foster nello sviluppo di protocolli per il progetto I-WAY [4, 5, 6], prima Grid de facto, sviluppata alla fine del 1994 all'Argonne National Laboratory. A seguito di tale esperienza la DARPA avviò il progetto Globus [1], che nel 1997 portò al rilascio della prima versione di GT (Globus Toolkit) [7], primo middleware di Grid Computing, e alla sua immediata diffusione su scala mondiale [8]. La diffusione di Globus portò ad un effetto domino coinvolgendo altre importanti istituzioni a contribuire allo sviluppo del Grid Computing e relativi middleware.

Nel corso degli ultimi anni il Grid Computing è entrato a far parte del vocabolario corrente di svariate istituzioni accademiche, scientifiche ed imprese. Da un lato ciò è ascrivibile ad un certo grado di maturità raggiunto dal Grid Computing con installazioni di produzione su scala globale che forniscono capacità computazionali fino a qualche anno fa pensabili solo avendo accesso a supercomputer e pertanto fino ad allora fruibili solo da pochi eletti; dall'altro lato il Grid Computing è tuttora un settore di ricerca vitale e passibile di soluzioni originali e innovative, considerato che di fatto la metafora iniziale in analogia alle Power Grid risulta tuttora parzialmente realizzata. Le capacità di calcolo disponibili a livello mondiale non sono accessibili in modo così immediato come collegare un dispositivo alla rete elettrica. Inoltre la complessità connessa al fornire l'accesso, in modo trasparente ed efficiente, a risorse etereogenee da dispositivi disparati da qualsiasi luogo del pianeta attraversando domini amministrativi differenti, la molteplicità di tipologie applicative e relativi requisiti, le possibilità offerte dalla convergenza di altre tecnologie quali P2P e sistemi multi-agente, fanno sì che ad oggi, se da un lato ci sono soluzioni mature di Grid Computing e sebbene gli ambienti Grid si moltiplichino e diffondano sempre più, tuttora non esiste una Grid ma tante Grid tra loro non interoperabili. Esistono difatti svariate tipologie di Grid Computing e relativi middleware che forniscono i meccanismi fondanti per il deployment di ambienti Grid, dedicati e non, su scala mondiale [9].

Per una breve panoramica architetturale degli ambienti Grid si può fare riferimento alle figure 1.2 e 1.3.

In figura 1.2, tratta da [10], un ambiente di Grid Computing è schematizzato

[1] http://www.globus.org

Figura 1.1: Globus Toolkit 4

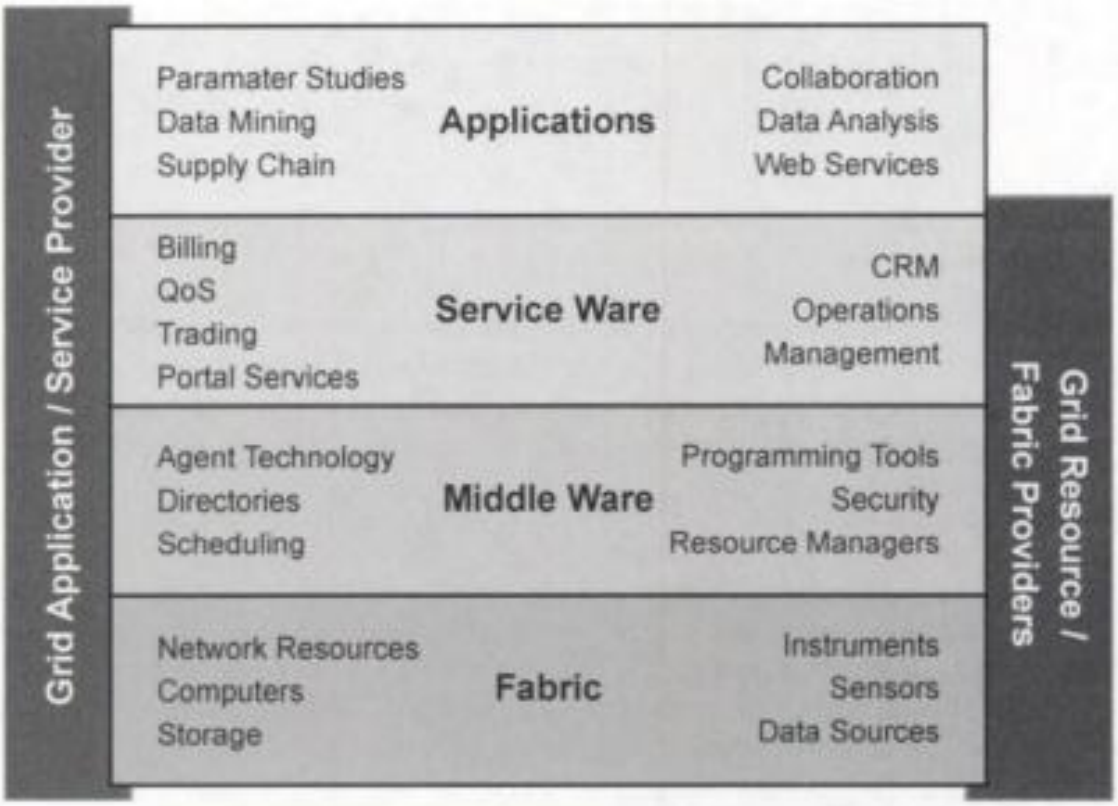

Figura 1.2: Grid Computing Market Taxonomy

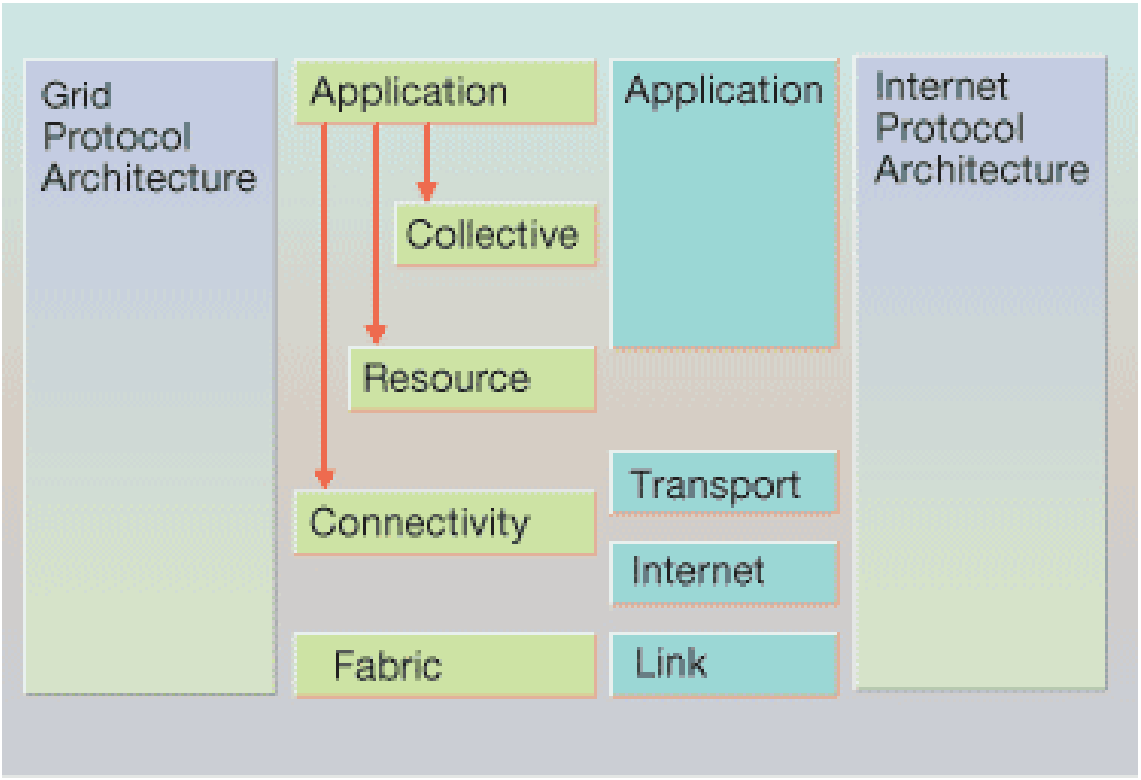

Figura 1.3: Grid Protocols Architecture

come costituito da quattro livelli primari ovvero la *Grid fabric*, il *middleware*, il *serviceware* e le applicazioni. La Grid fabric consiste di svariate risorse quali clusters di computers, reti, sistemi di memorizzazione di massa, strumentazioni scientifiche per la raccolta di dati e i relativi sistemi di gestione delle risorse. Il middleware, di cui Globus è un esempio, fornisce i servizi essenziali ad accedere in modo sicuro, uniforme e trasparente a risorse distribuite geograficamente fornendo la visione di singolo sistema cui l'utente accede al fine di eseguire applicazioni native o appositamente sottomesse alla Grid; inoltre il middleware fornisce svariate funzionalità in particolare per la gestione di dati, l'allocazione di risorse alle applicazioni ed il monitoraggio delle risorse (vedasi figura 1.1).

L'eterogeneità delle risorse e la loro distribuzione geografica ed amministrativa, proprie di un ambiente di Grid Computing, hanno imposto un approccio basato su standard aperti e mirato all'interoperabilità. Infatti sono molteplici le problematiche, tecniche e non, con cui si confrontano i modelli di Grid Computing ed in particolare la sicurezza in terminità di identità, autenticità e autorizzazione e la gestione delle risorse in termini di localizzazione, caratterizzazione, allocazione e monitoraggio. Per affrontare queste ed altre problematiche sono stati sviluppati protocolli standard secondo un approccio a livelli analogo a quello adottato per il modello ISO/OSI dei protocolli di Internet (vedasi figura 1.3[2]). Secondo tale stack, il livello *fabric* include protocolli e interfacce per l'accesso alle risorse rese disponibili nell'ambiente di Grid Computing, fornendo una visione logica di tali risorse. Al livello *connectivity* sono demandate le funzionalità relative all'interoperabilità tra risorse distinte e ad un approccio globale alla sicurezza tramite la GSI (*Grid Security Infrastructure*). A livello *resource* sono definiti protocolli relativi all'allocazione, monitoraggio e controllo remoto delle risorse (GRAM - *Grid Resource Allocation Management*), al trasferimento di dati tramite ad esempio GridFTP, a servizi di informazione sulle risorse disponibili (GRIS - *Grid Resource Information Services*). Al livello *collective* è demandato il compito di aggregare i servizi informativi relativi alle risorse su scala globale tramite indici distribuiti. Infine al livello *application* è demandata la definizione di protocolli e servizi per specifiche applicazioni o classi di applicazioni, secondo un approccio SOA (Service-Oriented Architecture) [11] e le direttive OGSA[3] [12] valide anche per l'implementazione di *Grid Web Services* applicativi, secondo il *Web Services Resource*

[2] http://www.research.ibm.com/journal/sj/434/joseph.html
[3] http://forge.gridforum.org/projects/ogsa-wg

Framework[4], per i livelli connectivity, resource e collective.

Tra i middleware di Grid Computing più diffusi, oltre il primo ad essere stato definito come tale, vi è per l'appunto Globus che è stato preso a riferimento durante la sperimentazione nel ricorso a testbed esistenti, in particolare Gilda[5], TriGrid[6] e Cometa[7], basati su gLite[8] [13], un'estensione al middleware Globus.

[4] http://www.globus.org/wsrf/
[5] https://gilda.ct.infn.it/
[6] http://www.trigrid.it/
[7] http://www.consorzio-cometa.it/
[8] http://glite.web.cern.ch/

Capitolo 2

Sistemi multi-agente

2.1 Introduzione

I sistemi multi-agente sono costituiti da agenti software caratterizzati da un certo grado di autonomia, intelligenza e reattività che collaborativamente si interfacciano con l'ambiente esterno per raggiungere i loro obiettivi [14]. La Foundation for Intelligent Physical Agents (FIPA) [1], nata come associazione non-profit ed attualmente undicesima commissione per gli standards della IEEE Computer Society [2] nota come FIPA Standards Committee (FIPA SC), ha come scopo primario la promozione di tecnologie Multi-Agent Systems (MAS) attraverso la definizione di linee guida e standards per l'interoperabilità tra implementazioni differenti di sistemi multi-agente.

Le specifiche FIPA (vedasi figura 2.1) sono pubblicamente disponibili e forniscono indicazioni su cui basare le implementazioni di tecnologie adatte alla progettazione di sistemi complessi caratterizzati da elevata interoperabilità ed al contempo eterogeneità in relazione a scopi e contesti di utilizzo. Di particolare interesse è il nucleo di specifiche FIPA (Agent Management [3] e Agent Communication Language [4]) che forniscono un framework concettuale, ampiamente accettato anche in ambito industriale, per la progettazione di piattaforme multi-agente per la gestione del ciclo di vita degli agenti, della loro operatività e collaborazione (vedasi figura 2.2).

[1] http://www.fipa.org/
[2] http://www.computer.org/portal/site/ieeecs/
[3] http://www.fipa.org/repository/managementspecs..html
[4] http://www.fipa.org/repository/aclspecs.html

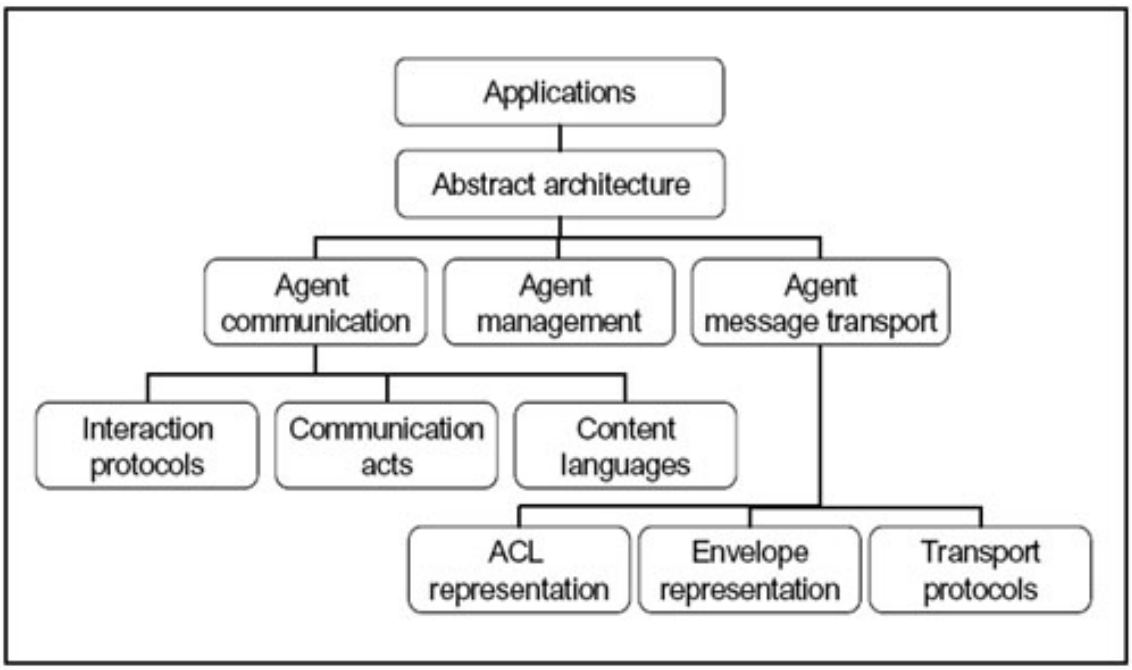

Figura 2.1: Specifiche FIPA

Esistono svariati middleware per sistemi multi-agente. Nel seguito è presentata una panoramica sul middleware JADE (Java Agent Development Framework), cui si è fatto ricorso nel valutare la fattibilità della sinergia e nello sperimentare alcune delle funzionalità della soluzione presentata nella Parte II.

2.2 JADE

JADE (*Java Agent Development Framework*) [15, 16] è un development framework open source scritto in Java per lo sviluppo di sistemi multi-agente conformi agli standard FIPA; per utilizzare JADE è necessario essere familiari con tali standard ed in particolare con le *Agent Management Specifications* e con l'*Agent Communication Language*.

JADE include un *Runtime Environment* all'interno del quale vengono creati ed eseguiti gli agenti, una libreria di classi per lo sviluppo degli agenti e un insieme di tool visuali per l'amministrazione, il monitoraggio ed il debugging degli agenti. Ciascuna

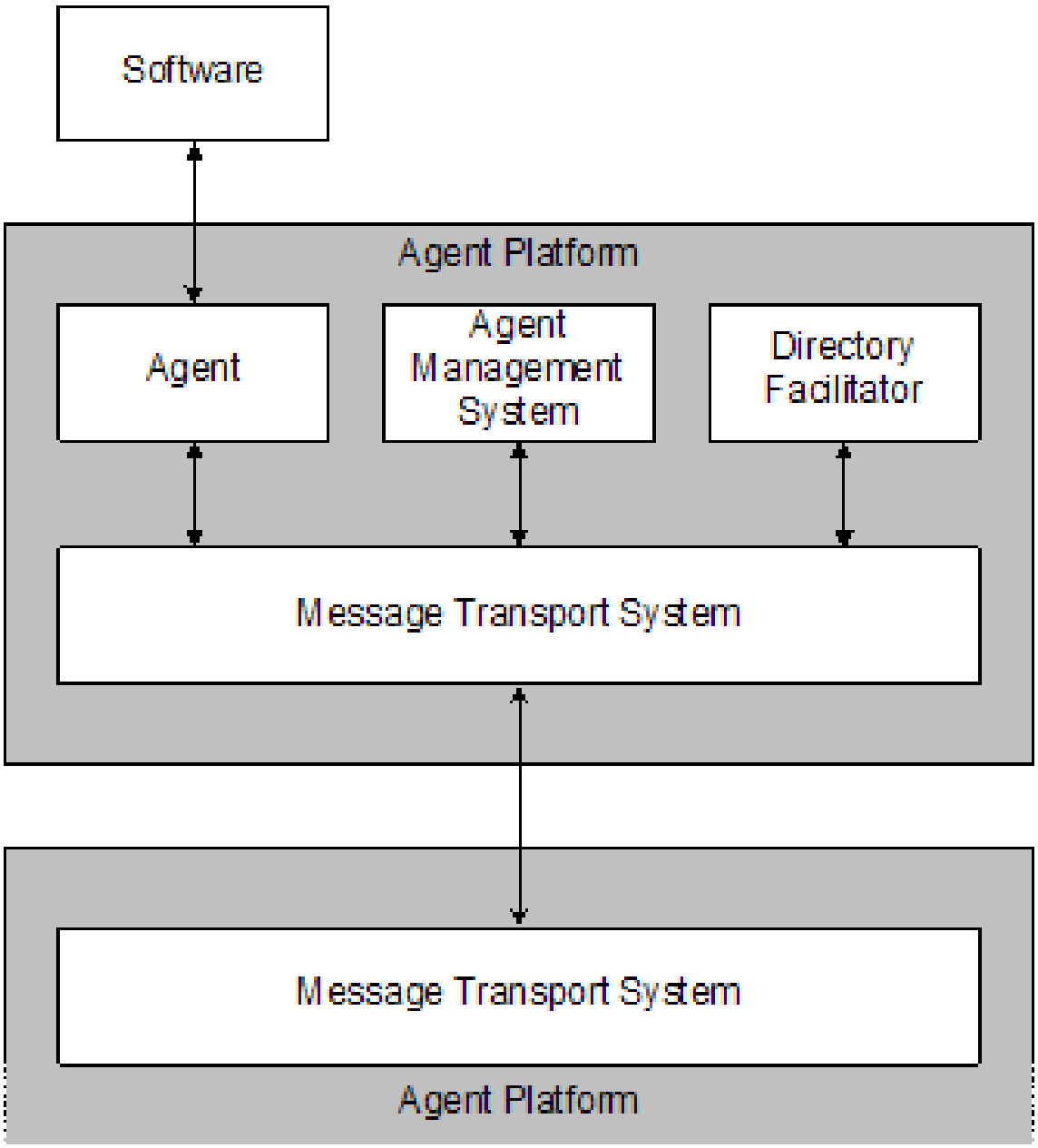

Figura 2.2: Agent Management Reference Model

istanza del JADE Runtime Environment è chiamata *Container* ed è instanziata all'interno di una *Platform*, insieme dei containers attivi.

In ogni Platform vi è almeno un container speciale detto *Main Container* che deve essere sempre attivo e che serve a registrare gli agenti contenuti in altri containers. Il Main Container contiene sempre i due servizi base previsti da FIPA per l'Agent Platform ovvero l'AMS (*Agent Management System*) che fornisce servizi di naming e di autorizzazione all'interno della Platform e il DF (*Directory Facilitator*) che agisce come servizio pagine gialle cui ciascun agente fa riferimento per conoscere quali agenti interrogare per raggiungere i propri obiettivi.

La comunicazione tra servizi e agenti avviene per il tramite del *Message Transport System* noto anche come ACC (*Agent Communication Channel*) che fornisce indipendenza dal protocollo di trasporto purché le comunicazioni siano conformi all'ACL (*Agent Communication Language*) definito dallo standard FIPA per l'interoperabilità degli agenti.

Gli agenti in JADE sono implementati come classi, derivate dalla classe base *Agent*, che si specializzano in funzione dei loro comportamenti. A ciascun agente è associato un identificatore nome come AID (*Agent ID*) nella forma <agent-name>@<platform-name>. Ciascuna funzionalità di un agente è realizzata tramite la gerarchia di classi *Behaviours* e sue classi derivate definite dal programmatore.

Conformemente alle specifiche FIPA sull'*Agent Life Cycle* (vedasi figura 2.3) la vita degli agenti è regolata da un diagramma di transizione tra una serie di stati:

1. INITIATED, agente appena creato
2. ACTIVE, agente attivo
3. SUSPENDED, agente sospeso, nessun behaviour in esecuzione
4. WAITING, agente bloccato in attesa di un evento
5. DELETED, agente distrutto
6. TRANSIT, stato in cui entra un agente durante la migrazione verso una nuova locazione

Il modello computazionale degli agenti è di tipo multi tasking non preemptive;

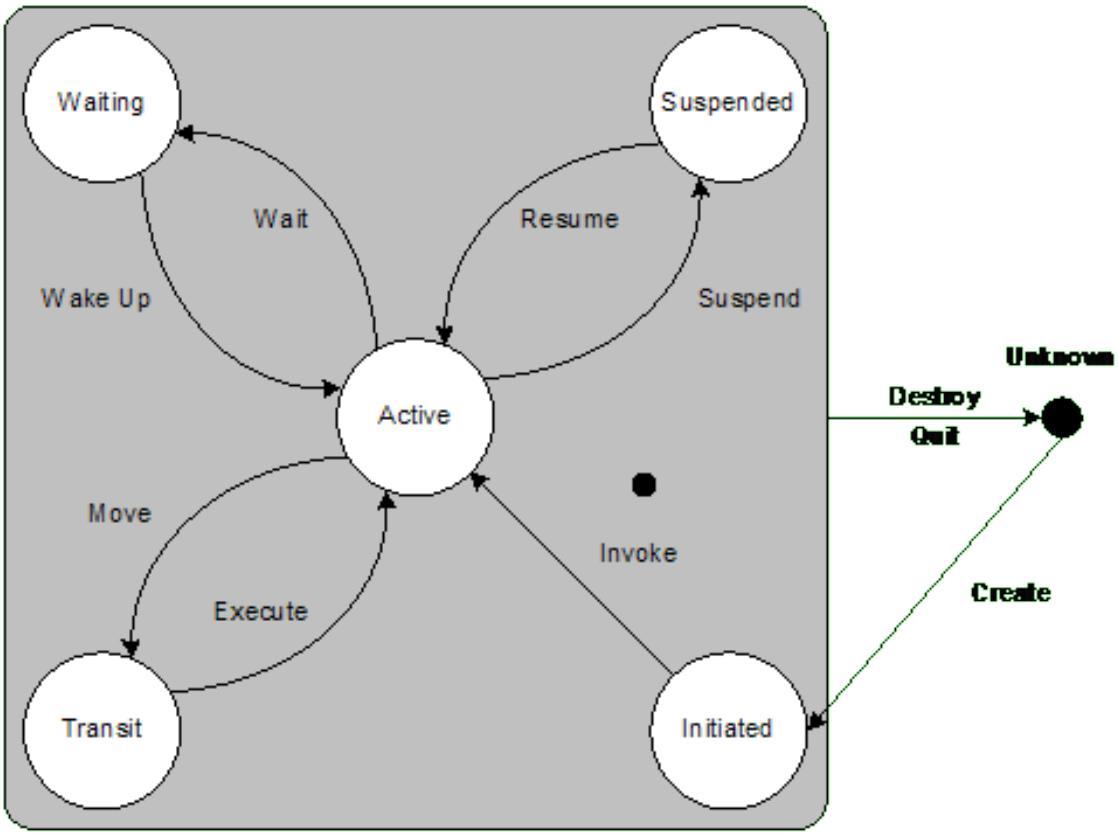

Figura 2.3: Agent Life Cycle

uno scheduler interno alla classe base Agent si occupa di schedulare automaticamente i behaviours.

JADE fornisce tre principali forme di behaviours estendibili:

1. *OneShotBehaviour*, eseguiti una sola volta
2. *CyclicBehaviour*, eseguiti periodicamente
3. *CompositeBehaviour*, che permettono di creare behaviours composti da più children behaviours; è possibile specificare una policy di scheduling dei figli per ogni CompositeBehaviour:
   - *SequentialBehaviour*, esecuzione sequenziale
   - *ParallelBehaviour*, esecuzione parallela
   - *FSMBehaviour*, esecuzione basata su una *Finite State Machine* specificata dal programmatore

Per ciascun agente è allocata una coda privata di messaggi a cui l'agente accede in modo sincrono o asincrono. Sebbene l'ACL fornisca quanto necessario a realizzare tutte le forme di comunicazioni desiderate in un sistema multi agente, JADE implementa secondo le specifiche FIPA una serie di protocolli per la gestione dei pattern conversazionali più comuni quali *Achieve Ractional Effect*, *FIPA-Contract-Net* e *FIPA-Subscribe*.

L'esecuzione di un agente può avvenire fondamentalmente per tre vie:

1. specifica degli agenti da eseguire da linea di comando
2. esecuzione tramite il tool amministrativo RMA (*Remote Monitoring Agent*)
3. lancio tramite l'interfaccia Java *InProcess* da un'applicazione Java esterna tramite il metodo statico jade.core.Runtime.instance()

Nell'insieme queste funzionalità di base forniscono un potente middleware per la comunicazione trasparente di agenti distribuiti e per l'adattamento degli agenti a compiti ed applicazioni disparate.

Capitolo 3

Transcoding

Il termine *transcoding* ha un significato ampio per la varietà delle tipologie di contenuti cui si riferisce e di metodologie con cui si può implementare. La scelta di studiare uno scenario di distribuzione di contenuti multimediali comprensivo di adattamento tramite online transcoding è nata in funzione del crescente interesse verso tale tecnica per finalità sempre più sentite che vanno sotto la sigla di *UMA* (*Universal Media Access*) [17] e degli elevati costi computazionali connessi a tale attività che ben si confà ad essere eseguita sfruttando le ingenti risorse fornite da ambienti di Grid Computing. L'UMA nasce come risposta alla molteplicità di formati, dispositivi di accesso e tipologie di rete[18].

La molteplicità di approcci al transcoding è tale da costituire una branca di ricerca a sè stante volta tra l'altro ad ottenere un compromesso ottimale tra qualità e costi attraverso tecniche che mirano a fornire alti livelli di qualità, riducendo gli effetti indesiderati e la complessità computazionale.

Nella soluzione presentata in questa tesi si fa riferimento, in relazione alle sperimentazioni effettuate, ad un approccio di transcoding intenzionalmente incentrato su elevati costi computazionali per dare risalto alle proprietà offerte dal ricorso al Grid Computing e riferirsi ad uno scenario *worst-case* rispetto alle potenzialità di ottimizzazione offerte dal ricorso ad algoritmi di adattamento dei contenuti più raffinati usufruibili grazie alla flessibilità del ricorso ad agenti (vedasi sezione 4.1 a pagina 27).

Il transcoding nasce inizialmente in ambito Web [19, 20, 21] fino ad arrivare all'ambito della distribuzione di contenuti multimediali tenendo conto di aspetti di Quality of Service [18, 22, 23].

Lo scenario considerato in questa tesi si riferisce a contenuti multimediali, in particolare video, in virtù degli elevati costi di memorizzazione, reperimento e adattamento di tale classe di contenuti che giustifica il ricorso alle tecnologie di Grid Computing.

Il video transcoding [24] ingloba più dimensioni, in particolare la riduzione del bitrate, la riduzione spaziale (ad esempio la risoluzione e il numero di colori), la riduzione temporale (ad esempio il frame rate), la resilienza all'errore.

Il video transcoding include la trasformazione di un video da un formato di codifica ad un altro. Partendo da un formato di alta qualità si può generare un contenuto multimediale adatto per un'ampia gamma di piattaforme. Il transcoding è basato fondamentalmente sulla compressione dei dati che, mentre in passato aveva come scopo quello di ottimizzare l'utilizzo dello spazio di storage a causa di costi elevati, oggi è più orientata all'ottimizzazione del trasporto di dati nelle reti ed in particolare su Internet. Una distinzione di base della vasta gamma di algoritmi di compressione [25] è basata sulla distinzione tra tecniche *lossy* e tecniche *lossless*.

La compressione lossless in genere si basa su tecniche di entropy encoding e algoritmi quali quello di Huffman e tecniche di dictionary encoding e algoritmi quali quello di Lempel-Ziv-Welch. Considerato che in questa tesi si fa riferimento a contenuti multimediali costituiti da video, le tecniche di transcoding applicabili utilizzano algoritmi di compressione video. Nell'ambito della compressione video la suddivisione tra tecniche lossy e tecniche lossless resta valida. Esistono infatti algoritmi di compressione video lossless [1] fondamentalmente basati su principi analoghi a quelli citati sopra.

La compressione lossy solitamente si basa su tecniche di DCT (Discrete Cosine Transform) [26] e di MC (Motion Compensation) [27]. I coefficienti di DCT sono sottoposti a quantizzazione per ottenere il bitrate desiderato. I frame sono codificati in gruppi detti GOP (Group of Pictures) [28] e in formato compresso ciascun GOP inizia con un frame speciale noto come I-Frame (Intra-Frame) o keyframe; agli I-Frame si affiancano i P-Frame e i B-Frame che sono essenzialmente basati sulla codifica delle differenze rispetto agli I-Frame (all'indietro per i P-Frame e bi-direzionalmente per i B-Frame). La rappresentazione di P-Frame e B-Frame è basata su *motion vector* che rappresentano la differenza introdotta dall'evoluzione temporale del video rispetto all'I-Frame. Le imprecisioni intrinseche alla rappresentazione delle differenze tra frame tramite motion vector

[1] http://www.compression.ru/video/

sono la fonte principale di perdita di informazione e conseguentemente di qualità rispetto al video originale dando luogo in particolare a fenomeni di distorsione noti sotto il nome di drift. Per ridurre il drift si fa solitamente ricorso alla MC (Motion Compensation) tramite l'introduzione di un prediction vector finalizzato alla compensazione degli errori introdotti dal motion vector.

L'approccio più immediato al video transcoding è noto come CPDT (Cascaded Pixel-Domain Transcoding) [24, 29] che, partendo dai singoli frame completi ottenuti dalla decodifica e decompressione del video originale, è in grado di fornire la migliore qualità ma al prezzo più alto sia in termini di risorse di calcolo che di risorse di I/O. Questo per l'appunto è l'approccio di transcoding adottato nella sperimentazione e simulazione della soluzione presentata in questa tesi.

Un approccio alternativo al transcoding per l'adattamento dei livelli di QoS è l'SVC (Scalable Video Coding); in tale approccio il video viene codificato una sola volta in più strati (layers) fornendo differenti livelli di qualità in relazione a parametri quali bitrate, risoluzione spaziale e temporale; in particolare per tale approccio esiste un'estensione di MPEG4 nota come MPEG4-Visual che supporta una forma di SVC nota come FGS (Fine Granularity Scalability) [30]. SVC permette la memorizzazione con più livelli di qualità di un contenuto su un singolo file e non è direttamente comparabile con il transcoding; inoltre fornisce una distribuzione efficiente di contenuti multimediali tenendo conto di aspetti di QoS e permette di semplificare la distribuzione dei file memorizzati in formato scalabile, in quanto un singolo file permette di fruire di più livelli di qualità ed è più semplice ed economico replicarlo.

Tuttavia il grado di adozione di SVC è limitato [31] mentre la maggioranza dei contenuti multimediali sono memorizzati in formati di codifica non scalabili e le tecnologie di transcoding sono adatte a fornire un adattamento dei contenuti esistenti che tenga conto di aspetti di QoS.

Parte II

Una architettura QoS-aware per l'online transcoding e streaming di contenuti multimediali in un ambiente di Grid Computing

In questa tesi si propone una sinergia tra tecnologie Multi-Agent e Grid Computing con l'abilità di emulare alcune caratteristiche proprie di una CDN (Content Delivery Network o Content Distribution Network) e incentrata sull'online transcoding e streaming di contenuti multimediali tenendo presente aspetti di Quality of Service in tali attività.

La sinergia tra tali tecnologie permette di sfruttare le ingenti quantità di risorse offerte dagli ambienti di Grid Computing non solo per memorizzare i contenuti multimediali ma anche per realizzarne l'online transcoding (vedasi capitolo 3 a pagina 16) quando le caratteristiche e le condizioni della rete non supportino la trasmissione del contenuto originale o il client utente non disponga di un player adatto alla ricezione e presentazione del contenuto multimediale nel suo formato originario.

La soluzione proposta, che si pone al di sopra del middleware di Grid Computing, è in grado da un lato di identificare uno Storage Element che mantiene la replica del contenuto richiesto ed è abbastanza vicino all'utente, ove tale replica esista; dall'altro quando il contenuto è memorizzato in un formato non riconosciuto dal player dell'utente, la soluzione proposta seleziona un Computing Element vicino a tale Storage Element e che disponga di una capacità computazionale adeguata ad effettuare l'online transcoding, fornendo così all'utente il contenuto nel formato richiesto. La selezione di tali risorse della Grid è effettuata con un'euristica di prossimità basata su metriche mirate a minimizzare le latenze per migliorare la Quality of Service percepita dall'utente.

Al fine di realizzare tali scopi l'approccio proposto fa ricorso alle tecnologie ad agenti come strumento per orchestrare le risorse fornite dalla Grid e si pone come una possibile alternativa a soluzioni dedicate quali generalmente sono le CDN.

Le CDN [32, 33, 34, 35] sono una famiglia di sistemi distribuiti progettati avendo in mente come scopo primario la diffusione ad utenti finali di dati con un'elevata Quality of Service nell'ambito dell'infrastruttura globale e decentralizzata di Internet, caratterizzata da un'intrinseca difficoltà a garantire prestazioni appropriate o anche a gestire in modo sistematico problematiche prestazionali dando luogo spesso a una Quality of Service percepita dagli utenti finali non sempre soddisfacente e generalmente non predicibile. Lo scopo che si prefiggono le CDN è realizzato essenzialmente tramite la replicazione dei dati dalla loro origine a posizioni sparse su Internet e tali da servire l'utente recuperando i contenuti richiesti dalla replica più vicina alla posizione da cui ha origine la richiesta. Storage Servers ad alte prestazioni mantengono repliche dei contenuti che gli utenti finali

richiedono e la localizzazione delle repliche viene determinata dinamicamente al fine di ottenere le minime distanze tra utenti e dati.

In questo contesto le caratteristiche dei collegamenti di rete diventano preponderanti al fine di rispettare i requisiti di Quality of Service specificati dagli utenti e pertanto si fa ricorso al posizionamento ottimale di componenti software e di replica servers rispetto alle infrastrutture di rete esistenti al fine di usufruire di grandi ampiezze di banda e di ridurre le latenze di rete considerato che la rete può diventare il collo di bottiglia quando l'obiettivo primario è il trasferimento di ingenti quantità di dati con elevati livelli di servizio.

L'infrastruttura software necessaria a fornire un'alta Quality of Service assume un ruolo primario in questo contesto in quanto, pur disponendo di risorse hardware caratterizzate da alte prestazioni, è necessaria un'opportuna orchestrazione software per:

1. la dislocazione su scala geografica dei servers di replica in posizioni strategiche
2. la dislocazione delle repliche
3. la selezione del contenuto replicato più vicino all'utente
4. il mantenimento di alti livelli di hit ratio per le repliche

Esistono varie CDN commerciali[2], che forniscono le loro risorse a importanti fornitori di contenuti su Internet, ciascuna con la propria soluzione proprietaria e di cui non sempre è possibile conoscere i dettagli.

Una generica architettura di una CDN [32, 36] è mostrata in figura 3.1 e si compone di sette componenti:

1. *client*
2. *replica servers*
3. *origin server*
4. *billing organization*
5. *request routing system*

[2]http://www.web-caching.com/cdns.html

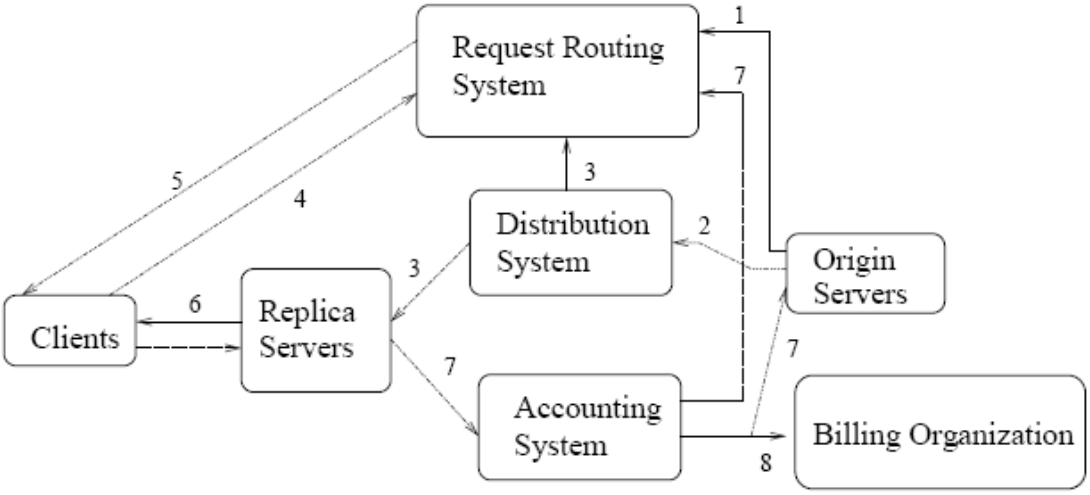

Figura 3.1: Panoramica sulle componenti di una generica CDN

6. *distribution system*

7. *accounting system*

I contenuti dell'*origin server* sono distribuiti sui *replica servers* dal *distribution system*.

La richiesta di un contenuto da parte di un *client* viene redirezionata in modo trasparente da parte del *request routing system* al replica server contenente tale contenuto più prossimo al client ed infine il replica server selezionato trasmette al client il contenuto richiesto.

A supporto di tale attività il *distribution system* effettua aggiustamenti dinamici della posizione delle repliche in funzione di statistiche aggregate dall'*accounting system* e notificate a quest'ultimo dai singoli *replica servers*. Il *request routing system* [37] tiene conto delle informazioni statistiche distillate dall'*accounting system* per effettuare scelte coerenti con l'evoluzione dell'intero sistema di CDN e ottimizzare le prestazioni fornite nel servire le singole richieste dei clients.

Il *request routing system* seleziona il replica server appropriato contenente una copia del contenuto richiesto e redireziona la richiesta su tale server.

I due criteri primari per scegliere il replica server più appropriato sono [38, 39]:

1. la prossimità tra client e replica servers che mantengono copia del contenuto richiesto
2. il carico dei replica servers

La determinazione della prossimità tra client e replica servers è solitamente basata su due metriche per la misurazione della distanza di rete ovvero il *round trip time* calcolato tramite tools come *ping* e l'*hops count* ottenuto tramite tools quali *traceroute*. Tali metriche forniscono informazioni sufficientemente accurate sulla prossimità tra clients e replica servers ma non sono strettamente correlate al carico di rete.

La determinazione del carico dei replica servers è effettuata solitamente con due tecniche:

1. *server push*, secondo cui i replica servers notificano il loro carico a componenti della CDN
2. *client probe*, in cui i replica servers di interesse vengono periodicamente interrogati per conoscerne il carico

Il problema principale connesso all'ottenimento e alla manipolazione dell'informazione sul carico è relativo al raggiungimento di un adeguato compromesso tra frequenza di aggiornamento di tale informazione e il traffico di rete generato per ottenerla.

In generale, mentre tenere conto di aspetti di Quality of Service è un'attività propedeutica alla trasmissione di qualsiasi tipo di dato, diventa un requisito quando i dati da trasferire sono contenuti multimediali, quali video e/o audio. In tal caso, i parametri di Quality of Service non sono solo correlati alla capacità di trasferire i contenuti multimediali in modo da ottenere una loro rappresentazione fluida lato client, ma anche alla possibilità del riproduttore di decodificare e riprodurre tali contenuti. Infatti, esiste una grande varietà di formati multimediali e di schemi di codifica molti dei quali standard e diffusi ma anche di tipo proprietario tali da necessitare di appositi codec o player. Questo significa che un contenuto multimediale seppur possa essere trasferito in tempo al player, al fine di assicurarne una riproduzione fluida lato client, non sia supportato dal player stesso.

Nel pensare un'architettura per lo streaming di contenuti multimediali molteplici sono gli aspetti da considerare:

1. varietà dei formati di codifica

2. necessità di disporre di opportuni strumenti di decodifica e presentazione dei dati multimediali lato client
3. variabilità dei protocolli di trasporto anche in funzione del formato dei contenuti
4. criticità dei collegamenti di rete per il reperimento e lo streaming di contenuti multimediali
5. variabilità delle condizioni di rete
6. eventuali requisiti di QoS da parte dell'utente
7. disseminazione dei contenuti multimediali
8. localizzazione dei contenuti multimediali
9. modalità di streaming dei contenuti multimediali
10. eventuale transcodifica dei contenuti multimediali prima della loro trasmissione
11. eventuale caching dei contenuti transcodificati

In relazione al primo punto, la molteplicità di formati di codifica dei contenuti multimediali può andare ad influire sulla

1. decodifica e presentazione lato client
2. sul protocollo di trasporto e sui requisiti di QoS della rete

I requisiti dell'infrastruttura di rete si manifestano sotto forma di proprietà quali ampiezza di banda, latenza, percentuale di pacchetti persi, ritardo.

Tali proprietà sono spesso specificate sotto forma di requisiti di QoS che se non opportunamente soddisfatti possono causare ritardi e interruzioni durante la presentazione dei contenuti all'utente causando in tal modo un degrado della qualità percepita dall'utente rispetto a quella richiesta.

Sebbene vi siano ben noti player general purpose, quali Windows Media Player, Xine e VLC, che sono in grado di supportare i formati multimediali più diffusi, ci sono schemi (quali QuickTime MOV o RealPlayer RAM) che richiedono player proprietari.

Inoltre, alcuni player non sono in grado di adattare il contenuto alla qualità del collegamento di rete usato dal client, che potrebbe non essere in grado di supportare lo streaming di certi contenuti multimediali per come sono memorizzati sul server di streaming, impedendone così una fluida presentazione all'utente.

Riassumendo, talvolta potrebbe non essere possibile trasmettere e presentare all'utente un contenuto multimediale poiché:

1. il player utente non supporta la decodifica di quel contenuto multimediale
2. il protocollo di trasporto non è supportato dal client
3. il collegamento di rete è inadeguato a supportare lo streaming di tale contenuto multimediale
4. le performance del client non sono adeguate

Al fine di risolvere tale problema, due soluzioni principali possono essere considerate:

1. pubblicare in una CDN varie versioni dello stesso contenuto multimediale usando differenti schemi di codifica
2. fornire alla CDN un meccanismo per effettuare transcodifiche dei contenuti multimediali

Tali due soluzioni sono tra loro compatibili, in quanto una volta transcodificato un contenuto è possibile memorizzarlo nella CDN al pari di altre repliche al fine di evitare successive transcodifiche dello stesso contenuto e trasferire direttamente il contenuto dai replica servers che ne posseggono una copia.

Tuttavia, le CDN tradizionali generalmente non forniscono un supporto al transcoding e ancor meno all'online transcoding in quanto esse sono essenzialmente focalizzate sulla distribuzione e replicazione dei dati per come sono ma non a loro trasformazioni che richiedano grandi capacità di calcolo per avere una transcodifica al volo dei contenuti richiesti.

Queste limitazioni non sono sempre ragionevoli se consideriamo che, in generale, i server sono dotati di capacità computazionali e prestazioni in genere superiori ai dispositivi di calcolo utente tradizionali, per cui sembra più appropriato che la transcodifica

dei contenuti sia effettuata lato server. D'altronde un generico server di streaming è dimensionato per supportare numerose richieste concorrenti per cui concentrare su di un singolo server la transcodifica di ciascun contenuto multimediale richiesto non è pensabile in quanto causerebbe una saturazione e un degrado delle prestazioni del server sull'intero insieme di richieste di streaming servite contemporaneamente.

La soluzione proposta presuppone uno scenario in cui i contenuti multimediali prima di essere inviati all'utente tramite streaming possano essere sottoposti ad online transcoding. Il transcoding di contenuti multimediali è un'attività caratterizzata dall'utilizzo intensivo di risorse computazionali necessarie per la transcodifica di contenuti multimediali da un formato ad un altro o cambiando proprietà spaziali (ad esempio la risoluzione dei frame) e/o temporali (ad esempio il frame rate). Solitamente il transcoding viene effettuato offline ma esistono contesti in cui può rivelarsi necessario l'online transcoding. Preporre l'attività di transcodifica all'attività di streaming introduce un overhead non trascurabile; le Content Delivery Networks almeno nella loro forma tradizionale non forniscono supporto per l'online transcoding.

Come accennato prima, al fine di effettuare l'online transcoding sono necessarie ingenti risorse di calcolo; una possibile soluzione è quella di sfruttare il potenziale offerto da ambienti di Grid Computing sia per la memorizzazione che per l'adattamento dei contenuti.

Tuttavia un'infrastruttura Grid Computing non è, al pari di una CDN, pensata per il provisioning di contenuti multimediali soddisfacendo i requisiti di QoS specificati dagli utenti ed effettuando scelte intelligenti nell'utilizzo della rete per lo streaming di tali contenuti.

La soluzione proposta, parzialmente pubblicata su proceeding internazionali [40, 41, 42], sfrutta il potenziale offerto da infrastrutture Grid general purpose preesistenti per la memorizzazione, il reperimento, l'online transcoding e lo streaming di contenuti multimediali ad utenti finali, fornendo una gestione soft della Quality of Service tramite la sinergia con le tecnologie multi-agente.

Capitolo 4

Motivazioni della sinergia adottata

4.1 Framework concettuale

La distribuzione di contenuti multimediali con ricorso all'online transcoding dei contenuti deve tenere conto di aspetti di Quality of Service (QoS) sia a livello applicazione che a livello di rete; i seguenti parametri di QoS devono essere tenuti presenti:

- *User-QoS* - la QoS specificata dall'utente correlata sia alle capacità del client utente sia alle caratteristiche del contenuto multimediale richiesto. Dato un client multimediale P, i parametri di QoS ad esso relativi sono indicati con $Q^C(P)$ che consiste di tuple nella forma (p_i^P, V_i^P), ove p_i^P è il nome del parametro e $V_i^P = (\rho_{i,1}^P, \ldots, \rho_{i,n}^P)$ è l'insieme dei valori ammissibili per tale parametro. I parametri $Q^C(P)$, che si riferiscono alle capacità del client, rientrano nella *Application-level QoS* (vedasi tabella in Figure 4.1).

Application QoS	**Network QoS**
Schema di codifica	Ampiezza di banda
Schema di compressione	Throughput Jitter
Rapporto di compressione	Latenza di rete
Frame Rate	Jitter
Risoluzione	Packet loss ratio
Numero di colori	

Figura 4.1: Parametri di QoS a livello di applicazione e di rete

- *Content-QoS* - la QoS relativa alle caratteristiche del contenuto multimediale fornito indicate con Q^S. Dato un contenuto multimediale X, i parametri di QoS del contenuto X memorizzati su uno storage S sono indicati con $Q^S(X)$, che consiste di tuple nella forma (p_i^X, V_i^X), ove p_i^X è il nome del parametro e $V_i^X = (\rho_{i,1}^P, \ldots, \rho_{i,n}^P)$ è l'insieme dei valori ammissibili per tale parametro. Nel seguito si fa riferimento ad un insieme di valori ammissibili costituito da un solo elemento $V_i^X = (v_i^X)$ non considerando formati multilayer [43].

- *Network-QoS* - la QoS relativa alle capacità della rete nel percorso dall'utente al server di storage; include parametri quali la banda end-to-end e la latenza di rete (vedasi tabella in Figure 4.1). Dato un collegamento di rete L, si indicano i parametri di QoS associati con $Q(L)$.

Per supportare lo streaming del contenuto multimediale X verso il player P attraverso il percorso di rete L, in primo luogo è necessario che $Q^S(X)$ sia compatibile con $Q^C(P)$, cioè che il valore di ciascun parametro P_i^X appartenga all'insieme di valori previsti per il parametro P_i^P esprimibile col predicato:

$$\forall (p_i^X, v_i^X) \in Q^S(X), \exists (p_i^P, V_i^P) \in Q^C(P) : p_i^X = p_i^P \wedge v_i^X \in V_i^P \tag{4.1}$$

L'utilizzo della rete per lo streaming del contenuto dipende sia dal contenuto che dal formato richiesto dal client, da cui derivano dei requisiti minimi di QoS a livello di rete che se non rispettati possono dar luogo ad uno streaming non fluido del contenuto richiesto.

Per astrarre tale concetto, si consideri una funzione $QNET(\cdot)$ che, dati i parametri di QoS a livello applicazione (quali sono $Q^S(X)$ o $Q^C(P)$), restituisca un insieme di parametri di QoS a livello di rete[1].

Per i parametri di QoS relativi al contenuto, $QNET(Q^S(X))$ fornisca i parametri di QoS necessari a trasmettere il contenuto X (4.2); mentre per i parametri di QoS relativi all'utente, $QNET(Q^C(P))$ restituisca i parametri di QoS a livello di rete necessari a trasmettere il contenuto con i minimi parametri di QoS specificati dall'utente.

$$Q(L) \geq QNET(Q^S(X)) \tag{4.2}$$

[1]La possibilità di definire una tale funzione è documentata dalla letteratura sull'adattamento di contenuti multimediali.

Qui con $A \geq B$, dove A e B rappresentano parametri di QoS a livello di rete, si intende che se la rete è in grado di supportare la fornitura del contenuto A lo è anche per il contenuto B. Pertanto, la formula (4.2) indica che il collegamento di rete L deve avere parametri di QoS almeno pari a $QNET(Q^S(X))$.

Quando il predicato (4.1) non è soddisfatto, un'appropriata operazione di transcoding deve essere effettuata, al fine di adattare $Q^S(X)$ a $Q^C(P)$; in questo caso, si può astrarre tale concetto, supponendo l'esistenza di una funzione di transcoding $transcode(\cdot)$, che preso in input il contenuto X lo trasformi nel contenuto $X\prime$ al fine di soddisfare il predicato (4.1) ed essere compatibile con i parametri di QoS dell'utente $Q^C(P)$:

$$\exists transcode(\cdot) :$$
$$transcode(X) = X\prime \vee Q^S(X\prime) \cap Q^C(P) \neq \oslash \tag{4.3}$$

Il transcoding può essere realizzato sia lato server che lato client, e tale scelta può essere effettuata in relazione alle caratteristiche e condizioni del collegamento di rete.

In particolare, si possono considerare i seguenti casi:

A. $Q(L) \geq QNET(Q^S(X)), Q(L) \geq QNET(Q^C(P))$. In questo caso la rete è in grado di supportare i parametri di QoS relativi sia al contenuto X che ai requisiti dell'utente

B. $QNET(Q^S(X)) \geq Q(L) \geq QNET(Q^C(P))$. In questo caso la rete è in grado di inviare contenuti multimediali che abbiano gli stessi requisiti di QoS dell'utente ma non per lo specifico contenuto X. Se il transcoding è effettuato lato server, adattando il contenuto X tramite trasformazione nel contenuto $X\prime = transcode(X)$, il contenuto può essere trasmesso all'utente. Questo implica che il server conosca i parametri di QoS specificati dall'utente e sia in grado di adattarsi ad essi, ovvero possegga funzionalità di transcoding compatibili con l'operazione di transcodifica necessaria ad ottenere $X\prime$.

C. $QNET(Q^C(P)) \geq Q(L) \geq QNET(Q^S(X))$. Questo caso è simmetrico al caso **B**. La rete è in grado di supportare la trasmissione del contenuto X ma non è in grado di supportare i parametri di QoS specificati dall'utente. In tal caso il transcoding andrebbe effettuato lato client ed il client dovrebbe supportare la decodifica del contenuto X.

D. $QNET(Q^S(X)) \geq Q(L), QNET(Q^C(P)) \geq Q(L)$. Questo è il caso peggiore, perchè la rete non può supportare la trasmissione né del contenuto X né del contenuto $X\prime$ opportunamente adattato al fine di rispettare i requisiti di QoS dell'utente. Per rendere possibile la trasmissione del contenuto X, il contenuto dovrebbe essere reso disponibile in una forma intermedia X_L, tale che $Q(L) \geq QNET(Q(X_L))$. A che questo sia possibile il contenuto dovrebbe essere adattato prima lato server con un'operazione di transcoding $transcode^S(X) = X_L$ e quindi il risultato trasmesso andrebbe transcodificato lato client con un'operazione di transcoding $transcode^C(X_L) = X\prime$ ottenendo un contenuto riproducibile in modo fluido dal player. Tale trasformazione provocherebbe una perdita di informazione del contenuto influendo sulla qualità percepibile dall'utente, ma si rende necessaria per il trasferimento del contenuto attraverso la rete L.

Lo scenario considerato nel caso **D** è il più generale ed opportune funzionalità di transcodifica $transcode^S(\cdot)$ e $transcode^C(\cdot)$, devono essere disponibili al fine di rendere il contenuto X trasmissibile in modo compatibile con i parametri di QoS dell'utente, tenendo presente però che negli altri casi una o entrambe le funzionalità di transcodifica non vengono di fatto utilizzate; specificamente, nel caso **A** e **C** non sono necessarie operazioni di transcoding lato server.

4.2 Ricorso alla sinergia tra tecnologie multi-agente e Grid Computing

Sulla base del framework concettuale considerato nella precedente sezione, ogni volta che un contenuto multimediale deve essere fornito, se le relazioni (4.1) e (4.2) non sono soddisfatte, si rendono necessarie operazioni di transcoding $transcode^S(\cdot)$ e $transcode^C(\cdot)$ compatibili con tutti i requisiti di QoS coinvolti.

Considerata l'ampia messe di formati multimediali diffusi e la variabilità di parametri e condizioni di rete, avere sempre a disposizione tali funzionalità di transcodifica lato server, ed eventualmente lato client, non è un fatto scontato: una soluzione flessibile dovrebbe essere in grado di fornire la giusta combinazione di transcodifica al variare delle caratteristiche del player, del contenuto e della rete. Le tecnologie multi-agent possono

essere efficientemente utilizzate a tale scopo per fornire di volta in volta le opportune funzionalità di transcodifica richieste dal server senza che questo conosca tutti i possibili formati e schemi di codifica in circolazione.

Inquadrando tale approccio in un tradizionale scenario client-server, gli agenti forniscono il paradigma per supportare il deployment di volta in volta delle opportune capacità di transcodifica sul server sotto forma di *agenti di transcoding* che, agendo come proxy tra lo storage e l'utente, intercettano la richiesta e l'adattano tramite transcodifica, avendo la capacità di manipolare il contenuto X, essendo a conoscenza dei parametri di QoS $Q^C(P)$ dell'utente ed essendo pensati per supportare il player P.

Quando l'utente chiede un contenuto, l'agente di transcoding migra sul server e i dati sono trasferiti dal server al client:

- L'agente di transcoding recupera il contenuto multimediale X e lo analizza ricavandone i parametri $Q^S(X)$.
- L'agente di transcoding, sulla base di $Q^S(X)$, $Q^C(P)$ e dei parametri di QoS $Q(L)$ supportati dalla rete, effettua le opportune conversioni e adattamenti della QoS tramite l'operazione di transcoding $transcode^S(\cdot)$ per rendere possibile la trasmissione attraverso il collegamento di rete L del contenuto X dal server S al player P.
- Infine lato player sono effettuate eventuali trasformazioni finali del contenuto ricevuto guidate dai parametri di QoS $Q^C(P)$.

Questo scenario client-server approcciato con una soluzione ad agenti, basato sulla distribuzione dei processi di transcodifica tra client e server, presenta numerosi vantaggi che possono essere riassunti come segue:

Semplificazione della progettazione del server A differenza di soluzioni tradizionali, l'approccio basato su agenti non impone che il server conosca e implementi il più ampio insieme di protocolli, schemi di codifica e compressione, al fine di essere compatibile con un adeguato numero di player.

Migliore distribuzione dei processi di transcoding Considerato che l'agente di transcoding viene trasmesso dal client e incapsula gli appropriati algoritmi per realizzare lato server operazioni di transcoding in linea con i parametri di QoS $Q^C(P)$ specificati dall'utente, la specifica delle attività da effettuare lato client e quelle da

demandare al server è meno rigida, permettendo di distribuire meglio le operazioni coinvolte nella distribuzione del contenuto all'utente.

Migliore gestione della QoS L'agente di transcoding proviene dal client ed incapsula opportuni algoritmi per la trasposizione diretta dei parametri di QoS $Q^S(X)$ del contenuto verso i parametri di QoS $Q^C(P)$ dell'utente, riducendo i rischi di un degrado eccessivo della qualità dovuto al ricorso ad algoritmi troppo generici.

Miglior adattamento alle caratteristiche della rete Gli agenti di transcoding sono a conoscenza dei parametri di QoS $Q^C(P)$ dell'utente e possono monitorare il collegamento di rete per adattare i parametri utilizzati durante le operazioni di transcoding ed evitare ritardi nella riproduzione del contenuto.

Mentre un approccio basato su agenti può apportare i vantaggi descritti, presenta degli svantaggi connessi ai requisiti di potenza computazionale richiesta lato server per supportare operazioni di transcodifica CPU-intensive in presenza di un crescente numero di richieste utente concorrenti: in questo caso, in base alla soluzione delineata per un semplice scenario di tipo client-server ove per ciascuna richiesta utente deve essere eseguito un agente di transcoding, si può avere una saturazione nell'utilizzo delle risorse di calcolo e di I/O del server.

Per questa ragione, una sinergia con le tecnologie di Grid Computing, può fornire un metodo per superare tale problema fornendo una piattaforma provvista di ingenti risorse di calcolo e di storage su cui ospitare una soluzione basata su tecnologie multi-agente.

Al fine di permettere l'adattamento dei contenuti, gli agenti di transcoding devono essere allocati su host con capacità di calcolo appropriate alla loro esecuzione. Gli ambienti di Grid Computing offrono un'ingente quantità di risorse di calcolo e di memorizzazione utilizzabili dagli agenti per portare a compimento le loro attività.

L'idea di effettuare il transcoding dei contenuti in ambienti di Grid Computing diviene non soltanto fattibile ma anche preferibile, grazie alle capacità offerta dalla Grid nel gestire e offrire grandi quantità di spazio di storage e di potenza computazionale. La Grid può essere utilizzata non solo per la memorizzazione dei contenuti multimediali ma anche per effettuare il transcoding degli stessi.

Considerata la natura general purpose degli ambienti di Grid Computing nonché la loro bassa reattività dovuta alla complessità dei middleware e alla distribuzione delle risorse su scala geografica, il semplice deployment su Grid degli agenti per l'online transcoding e streaming dei contenuti non sarebbe sufficiente a fornire una soluzione performante.

A fianco alla possibilità di sfruttare ingenti risorse si pone il problema di orchestrarle per evitare di rendere vano il ricorso a tale mole di risorse per l'adattamento dei contenuti ove poi le risorse vengano ad essere mal utilizzate e possano, in assenza di opportuni accorgimenti, portare ad un degrado della Quality of Service. Nel capitolo che segue vengono presentati tali accorgimenti basati, oltre che sugli agenti di transcoding, sulla selezione delle risorse di calcolo e di storage tramite servizi propri delle tecnologie ad agenti le cui scelte sono indirizzate da un'euristica di prossimità e da un'opportuna orchestrazione delle risorse in funzione di tali scelte.

Capitolo 5

Descrizione dell'architettura

5.1 Panoramica sull'architettura proposta

L'architettura proposta prevede una serie di servizi atti ad emulare alcune proprietà di una Content Delivery Network in un ambiente Grid Computing e a fornire l'orchestrazione necessaria (vedasi figura 5.1 per una panoramica delle principali interazioni):

1. a gestire le richieste di contenuti multimediali degli utenti
2. ad individuare i contenuti multimediali richiesti nell'ambito delle risorse di storage della Virtual Organization
3. a predisporre i processi necessari all'online transcoding dei contenuti multimediali ove necessario
4. ad individuare le risorse computazionali e di storage più adatte a soddisfare le singole richieste utente secondo dei requisiti di Quality of Service
5. a verificare la rispondenza della Quality of Service percepita dall'utente rispetto ai requisiti specificati nella richiesta
6. ad effettuare un adattamento della QoS attraverso:
    - online transcoding dei contenuti al fine di ridurre la quantità di dati che transitano in rete tra utente e sorgente dello stream multimediale mettendo in

atto meccanismi di adattamento della QoS al fine di rispettare, ove possibile, i requisiti di QoS specificati nella richiesta dell'utente

- delega delle operazioni di transcoding e streaming ad altro/altri agenti al fine di ridurre il delay nello streaming dei contenuti multimediali per ogni singola richiesta utente

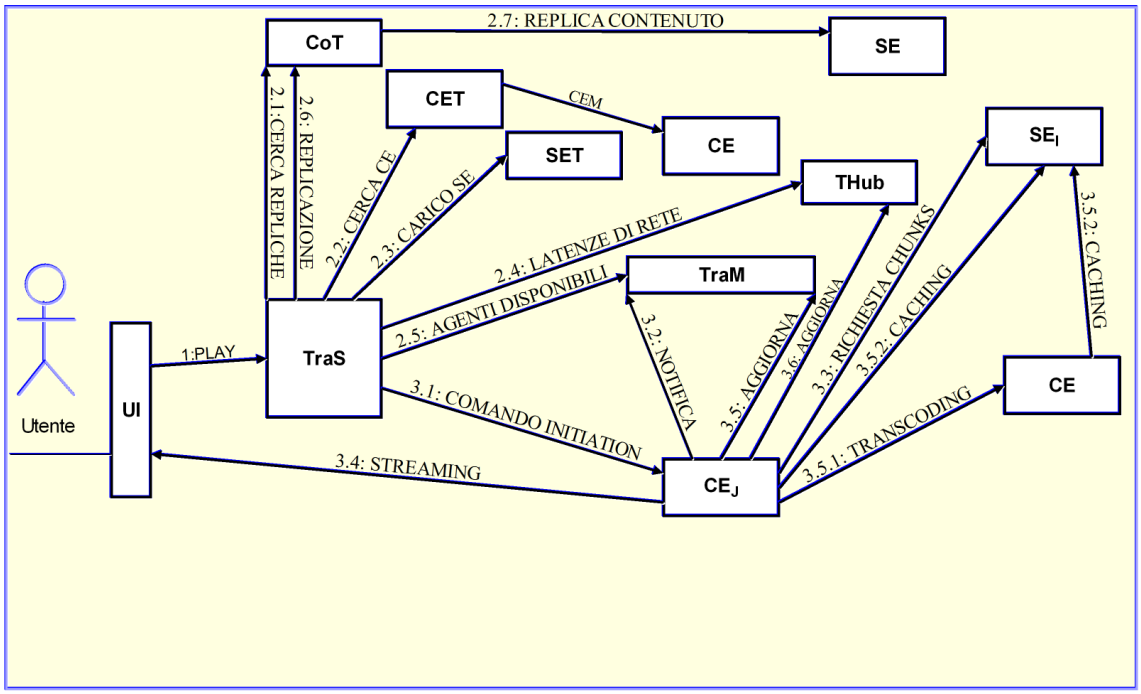

Figura 5.1: Panoramica interazioni

La soluzione proposta si compone dei seguenti servizi:

1. TraS (Transcoder Selector)
2. CoT (Content Tracer)
3. THub (Transcoder Hub)
4. CET (Computing Element Tracer)
5. SET (Storage Element Tracer)

6. TraM (Transcoder Monitor)

Inoltre è costituita da una serie di componenti replicate sui Computing Element:

1. NM (Network Monitor)
2. CEM (Computing Element Monitor)
3. GM (Grid Mediator)

Il servizio CoT in collaborazione con il servizio TraS si occupa della distribuzione dei contenuti negli Storage Element che hanno funzionalità di replica servers.

L'attività di request routing è svolta principalmente da TraS tenendo conto delle attività di transcoding e relative risorse computazionali necessarie ad effettuarle.

Le attività di accounting sono realizzate dalla collaborazione di più servizi ed in particolare dal servizio CoT per tracciare gli accessi ai contenuti, dal servizio TraS tramite i servizi CET e SET per monitorare la disponibilità delle risorse e tramite il servizio THub per monitorare le condizioni della rete.

5.2 Servizi dell'architettura proposta

5.2.1 TraS (Transcoder Selector)

Questo servizio si pone come intermediario tra l'utente e i servizi offerti dall'architettura.

Il servizio TraS tiene traccia delle istanze di Transcoding Agent e relative combinazioni di transcodifica disponibili nell'ambiente Grid e si interfaccia con il sottostante servizio TM (Topic Management service) [44] registrando su di esso tali agenti al fine di tenere traccia delle informazioni sulle combinazioni di transcodifica.

Una lista dei Computing Element disponibili per effettuare l'online transcoding è mantenuta da TraS in collaborazione col servizio CET (Computing Element Tracer) che, sulla base di notifiche da parte dei servizi CEM (Computing Element Monitor) in esecuzione sui Computing Element ed interrogazioni a tali servizi, raccoglie tali informazioni.

La lista di Computing Element disponibili ottenuta dal servizio CET è utilizzata da TraS all'occorrenza per il riutilizzo di istanze di Transcoding Agent ancora in esecuzione

nei Computing Element secondo le informazioni ottenute dal servizio TraM (vedasi sezione 5.2.6).

Il compito critico di selezionare il più appropriato Computing Element tra quelli noti interrogando il servizio CET è proprio di TraS. Questa attività è realizzata al fine di ottenere un compromesso accettabile tra bilanciamento del carico sui Computing Element e l'esigenza di mantenere le latenze di rete al minimo, in modo tale da ottimizzare il tempo di risposta e il tempo di streaming per le singole richieste di contenuti multimediali.

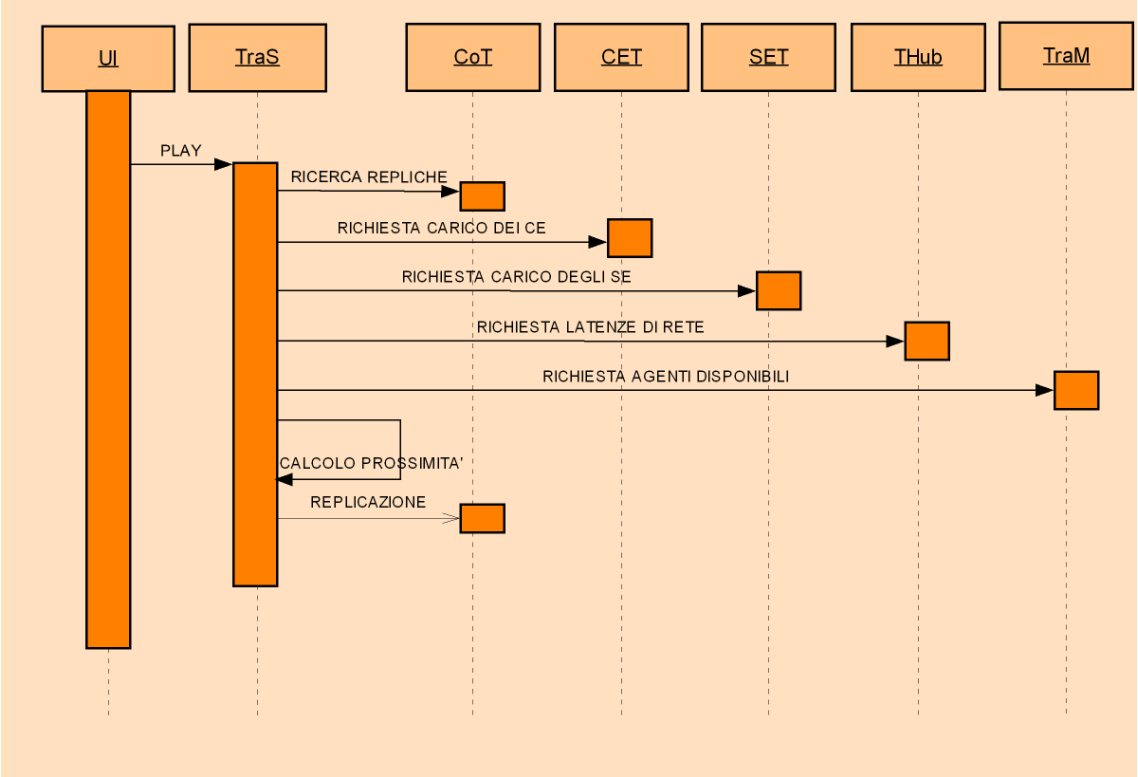

Figura 5.2: Diagramma di sequenza per la selezione di CE ed SE

TraS seleziona Storage Element e Computing Element sulla base di una euristica di prossimità (vedasi sezione 5.5 a pagina 50) basata sulle seguenti metriche:

1. latenza di rete tra Computing Element e Storage Element
2. latenza di rete tra utente e Computing Element
3. carico e caratteristiche dei Computing Element

4. carico degli Storage Element

Le latenze sono assunte come proporzionali alla somma delle seguenti distanze:

1. la distanza d(SE,CE) tra lo Storage Element ove il contenuto richiesto è memorizzato e il Computing Element che verrà ad essere utilizzato per l'esecuzione degli agenti
2. la distanza d(CE,UI) tra Computing Element e client utente

Al fine di ottenere una stima complessiva di come queste distanze varino tra i Computing Element e gli Storage Element disponibili per soddisfare una determinata richiesta utente, TraS interagisce con gli altri servizi ove non disponga di informazioni sufficientemente aggiornate (vedasi figura 5.2); in particolare interroga CoT (vedasi figura 5.1, 2.1) per ottenere una lista degli Storage Element che dispongono di una replica del contenuto multimediale richiesto, interroga CeT (vedasi figura 5.1, 2.2) per avere una lista dei Computing Element e relativo carico computazionale ed integra tali informazioni con informazioni sulle distanze di rete interrogando THub (vedasi figura 5.1, 2.4).

Il servizio SET (Storage Element Tracer) viene periodicamente interrogato (vedasi figura 5.1, 2.3) per ottenere informazioni sul carico degli Storage Element.

Non tutti i Computing Element sono appropriati per realizzare l'online transcoding in relazione ad una determinata richiesta utente. Il carico computazionale associato alla richiesta potrebbe essere eccessivo rispetto alla combinazione di transcodifica necessaria e a carico e profilo hardware del Computing Element. Per decidere se un Computing Element è appropriato va effettuata una valutazione del costo computazionale della richiesta ottenendo una stima del tempo necessario per la specifica transcodifica. Le informazioni sul costo di transcodifica sono mantenute da TraM che a sua volta ottiene tali informazioni dai Transcoding Agent. Le informazioni mantenute da TraM vengono interpolate ed utilizzate da TraS per decidere se esiste un Computing Element che in base alle informazioni in possesso di TraS sia in grado di soddisfare i requisiti di QoS associati ad una specifica richiesta.

Le coppie di Computing Element e Storage Element sono ordinate dal servizio TraS in modo crescente rispetto ai valori stimati tramite l'euristica di prossimità e viene scelta la prima coppia che disponga del contenuto richiesto e la cui capacità residua di calcolo supporti il costo di transcoding presunto.

Ove non esista tale coppia il servizio TraS effettua iterativamente il rilassamento dei parametri di transcoding compatibilmente con la minima qualità accettabile definita dall'utente (vedasi sezione 5.6 a pagina 52) fin quando non seleziona una coppia (CE, SE) adatta o, ove non sia possibile soddisfare i requisiti utente con le risorse disponibili, rifiuta la richiesta.

La transcodifica e lo streaming del contenuto multimediale richiesto è iniziata da TraS che sottomette o riutilizza Transcoding Agent sul Computing Element selezionato indicando nel messaggio di initiation (vedasi figura 5.1, 3.1) dell'agente l'URI necessario a reperire il contenuto multimediale dallo Storage Element selezionato. La sottomissione effettiva degli agenti avviene per il tramite del servizio GM (Grid Mediator) (vedasi sezione 5.3.3 a pagina 44), presente sul Computing Element selezionato, sulla base delle istruzioni ricevute da TraS. Per il riutilizzo di Transcoding Agent compatibili e disponibili sul Computing Element selezionato o vicini, TraS interroga il servizio TraM da cui ottiene l'elenco degli AID (Agent ID) (vedasi figura 5.1, 2.5) potenziali destinatari del messaggio di initiation. TraS si serve dei servizi di mobilità propri della piattaforma ad agenti per migrare o clonare istanze di agenti compatibili sul Computing Element selezionato al fine di aumentare la reattività del sistema ed aggirare la bassa reattività propria del sottostante middleware di Grid Computing.

5.2.2 CoT (Content Tracer)

Il servizio CoT si occupa di trovare lo Storage Element che contiene l'appropriato formato richiesto dall'utente e fornire l'URI di tale contenuto al servizio TraS.

A tal fine CoT mantiene, interfacciandosi con AMGA [45, 46] per il tramite del servizio GM (Grid Mediator), un repository dei contenuti multimediali memorizzati negli Storage Element dell'ambiente Grid.

I contenuti inseriti nel repository sono sottoposti a replicazione durante la selezione delle risorse da parte del servizio TraS basata sull'euristica di prossimità (vedasi sezione 5.5 a pagina 50); una volta scelta la coppia di (CE, SE) utilizzata dagli agenti per servire la richiesta utente se lo Storage Element selezionato non è il più vicino al Computing Element selezionato viene scelto lo Storage Element più vicino al Computing Element e privo di tale contenuto (vedasi la figura 5.2).

Il servizio TraS nel valutare la migliore coppia di Computing Element e Storage Element per servire una determinata richiesta ordina in modo crescente le coppie (CE_j, SE_i) secondo una metrica di prossimità. La prima coppia di tale ordinamento può fornire il riferimento ad uno Storage Element che non contiene il contenuto richiesto ma che fornirebbe un miglior compromesso tra carico e latenza di rete. Il servizio TraS notifica tale opportunità al servizio CoT (vedasi figura 5.1, 2.6) che si occupa di iniziare la replicazione del contenuto richiesto (vedasi figura 5.1, 2.7) verso tale Storage Element registrando la nuova replica nel suo repository a replicazione ultimata. Il servizio CoT si occupa di eliminare eventuali repliche dallo Storage Element selezionato ove la sua capacità sia al limite adottando una politica di tipo LRU. Le attività di gestione delle repliche sono effettuate da CoT per il tramite del servizio GM (Grid Mediator).

Per quanto concerne il caching (vedasi la sezione 5.4.5 a pagina 47) viene realizzato direttamente dagli agenti software (vedasi figura 5.1, 3.5.2); a caching ultimato, il servizio CoT viene notificato per mantenere aggiornato il suo repository.

5.2.3 THub (Transcoder Hub)

Questo servizio mantiene il monitoring delle distanze di rete tra gli attori direttamente coinvolti nella gestione e soddisfacimento delle singole richieste utente.

Durante la sua esecuzione un Transcoding Agent comunica periodicamente con il servizio THub notificandogli le latenze percepite rispetto all'utente e allo Storage Element. Queste misurazioni vengono utilizzate nell'euristica di prossimità adottata dal servizio TraS.

THub inoltre monitora periodicamente le latenze di rete tra coppie di Computing Element e Storage Element, eventualmente non ancora coinvolte nel soddisfacimento di richieste utente, inviando richieste di ping tra tali coppie. Tali richieste di ping sono inviate al servizio Network Monitor (vedasi sezione 5.3.1 a pagina 44), di cui è mantenuta al più un'istanza per Computing Element, indicando lo Storage Element rispetto a cui misurare la latenza di rete.

L'invio periodico di tali richieste di ping avviene per singola coppia (CE, SE) selezionata in modo casuale tra le coppie di CE ed SE conosciuti da THub in modo da evitare overhead non desiderati; la conoscenza della rete da parte di THub avviene in

modo incrementale e, per evitare una degradazione della qualità delle informazioni da esso mantenute, si utilizza un meccanismo di caching per indirizzare meglio la selezione casuale delle coppie di CE ed SE da monitorare al fine di ridurre il peso del monitoring sull'effettivo carico di rete ed un meccanismo di aging delle latenze per coppie già note per ridurre il peso di momenti di picco nel carico di rete presente al momento della misurazione della latenza di rete tra una coppia di CE ed SE.

5.2.4 CET (Computing Element Tracer)

Questo servizio mantiene traccia di informazioni sul workload, numero di processori non utilizzati dagli agenti, potenza di calcolo di ciascun processore e agenti attivi nei singoli Computing Element.

CET è principalmente inteso a monitorare i Computing Element dal punto di vista della loro disponibilità e della loro capacità di carico residua.

CET si interfaccia, con periodicità limitata in virtù della scarsa fluttuazione temporale delle risorse hardware disponibili in un ambiente Grid, con i servizi informativi della Grid, per ottenere informazioni concernenti i working node di cui si compone ciascun Computing Element, in termini di numero di host, numero di CPU e profilo hardware (capacità della CPU e RAM disponibile).

CET riceve dai servizi CEM, operanti sui singoli Computing Element, aggiornamenti di tali informazioni. Questa attività è svolta in background come notifica dai servizi CEM al servizio CET e viene mantenuta in una cache locale a CET. Quando le informazioni relative ad un Computing Element sono diventate obsolete nella cache sono richieste in modalità *pull* dal servizio CET all'appropriato servizio CEM.

CET viene utilizzato per ottenere informazioni su Computing Element i cui host sono idle o poco carichi. Inoltre CET è informato da parte di TraS dell'utilizzo dei nodi di un Computing Element per l'allocazione di Transcoding Agent in modo che sia CET a scremare un'informazione di sintesi con un valore compreso tra zero e uno sul carico del singolo Computing Element, informazione usata dal servizio TraS tramite l'euristica di prossimità.

5.2.5 SET (Storage Element Tracer)

Questo servizio mantiene traccia di informazioni sul carico dei singoli Storage Element.

SET riceve tali informazioni dai servizi Network Monitor presenti sui Computing Element che aggregano le notifiche ricevute dai Transcoding Agent presenti sui Computing Element e da essi periodicamente trasmesse al servizio di Network Monitor in base ai dati di dettaglio da tali agenti mantenuti sulla dimensione dei chunk recuperati dagli Storage Element prima di procedere alla loro transcodifica.

5.2.6 TraM (Transcoder Monitor)

Il servizio TraM o Transcoder Monitor, tiene traccia delle istanze di Transcoding Agent in esecuzione nei vari Computing Element e delle tipologie di transcodifica supportate da essi.

Inizialmente su ogni Computing Element sono attivate istanze di Transcoding Agent per renderle disponibili in relazione ad eventuali richieste utente; quando tali istanze non sono sufficienti a servire il carico corrente possono essere affiancate da nuovi agenti attivati dal servizio TraS.

TraM mantiene una lista delle istanze di agenti in esecuzione sulla base delle informazioni gestite dal servizio Directory Facilitator. Il servizio Directory Facilitator, intrinseco di un'architettura multi-agente, viene qui utilizzato in quanto fornisce il meccanismo di naming e localizzazione primario fornito dalle tecnologie multi-agent ed utilizzato per la registrazione degli agenti e per trovarne di già presenti nella Virtual Organization, riutilizzandoli al fine di ridurre l'overhead introdotto dall'inizializzazione di nuovi agenti.

Gli agenti, necessari per le attività di transcoding, streaming e adattamento della QoS, sono sottomessi per il tramite del servizio TraS (vedasi sezione 5.2.1 a pagina 36) e registrati sull'Agent Platform presso cui il servizio Directory Facilitator è in esecuzione.

TraM si basa sul servizio di Topic Management [44] nel fornire supporto a comunicazioni multicast basate su topic; il Topic Management Service è utilizzato come complementare al servizio Directory Facilitator e fornisce un meccanismo per estendere le informazioni presenti nelle pagine gialle dell'Agent Platform con informazioni sui formati

di transcodifica supportati dai singoli agenti di transcoding (vedasi sezione 5.4 a pagina 45).

Le comunicazioni tra agenti sono basate su convenzioni di naming uniforme che permettono di mandare messaggi indirizzati a specifici agenti o ad agenti registratisi su uno specifico topic.

L'utilizzo combinato di AID relativi a nomi di agenti e AID relativi ad argomenti (tipo di transcodifica supportata dai Transcoding Agent) è utilizzato nell'ambito della soluzione proposta:

1. per inviare messaggi a singoli agenti al fine di
   - istruirli sull'attività da compiere per soddisfare singole richieste di streaming e/o transcodifica tramite l'invio di un messaggio di **initiation**
   - interrompere l'esecuzione di un agente tramite l'invio di un messaggio di **termination**
2. per inviare messaggi a gruppi di agenti che supportino analoghe combinazioni di transcodifica per
   - richiedere ad altri agenti di transcodifica presenti su un Computing Element la collaborazione nella transcodifica parallela di chunk relativi ad una richiesta utente
   - trovare altri agenti sullo stesso Computing Element cui girare una richiesta quando un agente è troppo impegnato per iniziare il lavoro di transcodifica relativo ad un nuovo contenuto multimediale
   - riutilizzare chunk già transcodificati da altri Transcoding Agent, ancora presenti sull'host su cui gira il Transcoding Agent o eventualmente sottoposti a caching parziale su un vicino Storage Element

Inoltre TraM mantiene un repository, basato su AMGA [45, 46], degli algoritmi di transcodifica supportati dagli agenti, che aggiorna con nuovi elementi quando un utente o il provider dei contenuti pubblicano nuovi algoritmi di transcodifica utilizzati poi da TraS per l'inizializzazione di agenti di transcodifica.

5.3 Servizi sui singoli Computing Element

5.3.1 NM (Network Monitor)

Il servizio Network Monitor si occupa di effettuare il monitoraggio attivo delle latenze di rete tra Computing Element e Storage Element e il monitoraggio passivo del carico degli Storage Element.

Il servizio THub invia al servizio Network Monitor di un Computing Element una richiesta di ping tra tale Computing Element e uno Storage Element; il servizio Network Monitor si occupa di espletare tale richiesta misurando la latenza di rete espressa come round trip time dal Computing Element corrente allo Storage Element indicato nella richiesta e notifica la misurazione al servizio THub.

Inoltre il servizio Network Monitor riceve dai Transcoding Agent notifiche sul carico di singoli Storage Element basate sul lavoro di transcoding effettuato da tali agenti su contenuti reperiti da tali Storage Element. Le informazioni sul carico degli Storage Element sono periodicamente inviate al servizio SET o da questo espressamente richieste.

5.3.2 CEM (Computing Element Monitor)

Il servizio CEM raccoglie informazioni su carico e caratteristiche del Computing Element cui fa riferimento ed in particolare il carico complessivo del Computing Element, il numero di processori liberi, la capacità di calcolo di ciascun processore e sulla dislocazione degli agenti sugli host del Computing Element.

Tali informazioni sono aggregate dal servizio CET che le riceve per notifica dalle istanze del servizio CEM o interrogandole singolarmente all'occorrenza.

5.3.3 GM (Grid Mediator)

Il servizio GM si occupa di interfacciarsi con il sottostante middleware di Grid Computing per sottomettere il middleware degli agenti sotto forma di jobs e per sfruttare le funzionalità di trasferimento dati dell'ambiente di Grid Computing. E' presente un'istanza del servizio GM su ciascun Computing Element ivi incluso il Computing Element ospitante l'Agent Platform. La delega delle credenziali e la gestione degli agenti come job è effettuata su ciascun Computing Element dal servizio GM.

Il comportamento del servizio GM si differenzia sull'Agent Platform avendo la funzione di inizializzare i servizi di NM (Network Monitor) e di CEM (Computing Element Tracer) sui Computing Element della Virtual Organization.

A regime invece il servizio GM presente su ciascun Computing Element si occupa di allocare nuovi Transcoding Agent sulla base delle istruzioni ricevute dal servizio TraS.

Inoltre il servizio GM si pone come intermediario per gli accessi ad AMGA per i servizi CoT e TraM.

5.4 Agenti Software

Il Transcoding Agent si occupa di reperire in chunk il contenuto multimediale da un determinato Storage Element, effettuarne la transcodifica secondo le specifiche utente ed eventuali adattamenti della QoS e inviare in streaming i contenuti all'utente finale.

Il Transcoding Agent si compone di più behaviour:

1. Monitoring Behaviour
2. Collaboration Behaviour
3. Fetching Behaviour
4. Transcoding Behaviour
5. Caching Behaviour
6. Streaming Behaviour
7. QoS Behaviour

Ciascun Transcoding Agent si aspetta dal servizio TraS uno dei seguenti messaggi:

1. **initiation** che contiene
   - la localizzazione del contenuto multimediale originale da transcodificare
   - i parametri di transcodifica comprensivi di eventuali intervalli di chunk in caso di transcoding parziale finalizzato al caching

- informazioni relative al destinatario dell'eventuale attività di streaming
- parametri sul caching
- la qualità minima accettabile per l'utente richiedente

2. **termination** che termina l'agente liberando risorse sull'host da questo utilizzato

La scelta se terminare o meno un Transcoding Agent è effettuata dal servizio TraS in funzione della ratio tra tasso di richieste da questo gestite e numero di Transcoding Agent idle.

5.4.1 Monitoring Behaviour

Il Monitoring Behaviour misura le latenze di rete percepite dall'agente rispetto allo Storage Element e all'utente durante l'espletamento di una richiesta di quest'ultimo e notifica tali informazioni a THub.

Inoltre il Monitoring Behaviour misura il ritardo percepito dall'utente durante il soddisfacimento della sua richiesta e tale informazione è utilizzata dal QoS Behaviour per decidere eventuali adattamenti dei parametri di transcodifica.

Il Monitoring Behaviour effettua misurazioni sui costi di transcoding effettuati dall'agente e prima della sua terminazione notifica tali informazioni al servizio TraM per rendere più realistiche le informazioni utilizzate dal servizio TraS durante la selezione di Computing Element e Storage Element sulla base dell'euristica di prossimità.

5.4.2 Collaboration Behaviour

Il Collaboration Behaviour svolge le seguenti funzioni:

1. realizzare il transcoding parallelo di una richiesta
2. chiedere aiuto ad altri Transcoding Agent compatibili rispetto ad una combinazione di transcodifica che sta processando con tempi non adeguati a soddisfare i requisiti di QoS utente al fine di partizionare tale lavoro (5.6)
3. girare la richiesta ad altro agente già attivo sul Computing Element

Per realizzare il transcoding parallelo di una richiesta viene utilizzato un modello di tipo master-worker [47, 48, 49, 50] in cui un singolo agente coordina le attività degli

altri agenti, si occupa di riordinare i chunk transcodificati ed effettuarne lo streaming verso l'utente (vedasi figura 5.3).

5.4.3 Fetching Behaviour

Il Fetching Behaviour si occupa di recuperare il contenuto multimediale originale sotto forma di chunk dallo Storage Element indicato nel messaggio di **initiation** (vedasi figura 5.1, 3.3).

Il Fetching Behaviour recupera tali chunk in modo asincrono rispetto allo stato di transcodifica e streaming dei chunk precedenti in modo da rendere disponibili alla transcodifica i chunk possibilmente prima che debbano essere transcodificati dal Transcoding Behaviour al fine di non introdurre ritardi dovuti a chunk non ancora recuperati.

Il Fetching Behaviour si occupa inoltre di mantenere sempre disponibile in memoria primaria il prossimo chunk che il Transcoding Behaviour andrà a processare e di memorizzare sulla memoria secondaria gli altri chunk già ottenuti dallo Storage Element.

5.4.4 Transcoding Behaviour

Il Transcoding Behaviour si occupa di transcodificare sequenzialmente i chunk recuperati dal Fetching Behaviour e memorizzati localmente all'host nella memoria primaria da parte del Fetching Behaviour.

I chunk transcodificati sono mantenuti in memoria primaria da parte del Transcoding Behaviour per renderli immediatamente disponibili allo Streaming Behaviour o, in caso di transcoding parallelo (vedasi figura 5.3), sono inviati all'agente che svolge il ruolo di master.

5.4.5 Caching Behaviour

Per quanto concerne il caching viene utilizzato il caching totale dei contenuti transcodificati secondo diversi livelli di qualità e verso lo Storage Element da cui tali contenuti sono stati reperiti in formato originale; ad ogni variazione di qualità effettuata dal Transcoding Agent sulla base dei meccanismi di adattamento della QoS (vedasi sezione 5.6 a pagina 52) la richiesta di transcoding con i parametri non adattati viene girata ad

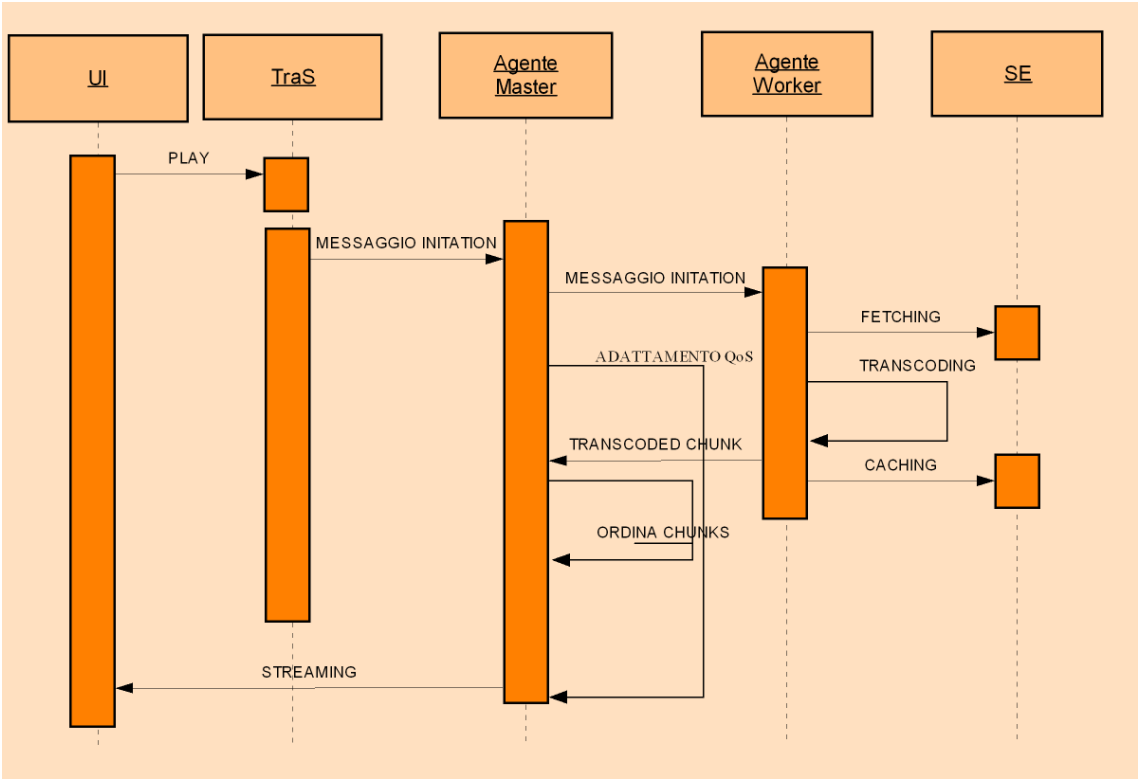

Figura 5.3: Diagramma di sequenza per il transcoding

un altro transcoding agent prossimo allo Storage Element utilizzato al fine di completare il transcoding.

Quando un Transcoding Agent adatta, a partire dal chunk i-esimo, i parametri di transcoding ad un nuovo livello di QoS, selezionato da parte del QoS Behaviour:

se per tale livello di QoS non esistono già copie del contenuto corrente negli Storage Element, invia un messaggio basato su topic indirizzato a tutti gli agenti con capacità di transcodifica corrispondenti ai nuovi parametri di transcodifica; ottiene una risposta dagli agenti disponibili e ne sceglie uno prossimo allo Storage Element corrente; trasmette a tale agente la richiesta di transcodificare il contenuto dal chunk 1 al chunk i-1 secondo i nuovi parametri di transcoding ma senza requisiti di QoS (vedasi figura 5.1, 3.5.1);

se i precedenti parametri di transcoding erano stati oggetto di un adattamento della QoS, invia un messaggio basato su topic indirizzato a tutti gli agenti con capacità di transcodifica corrispondenti ai precedenti parametri di transcodifica; ottiene una risposta dagli agenti disponibili e ne sceglie uno prossimo allo Storage Element corrente; trasmette a tale agente la richiesta di transcodificare il contenuto dal chunk i-esimo all'ultimo chunk secondo i precedenti parametri di transcoding ma senza requisiti di QoS (vedasi figura 5.1, 3.5.1)

Il ricorso al caching secondo una quantità discreta di livelli di qualità, è motivato dall'ampia gamma di parametri di transcoding che danno luogo ad una miriade di combinazioni; tuttavia, a più combinazioni di parametri di transcoding possono corrispondere livelli di qualità equivalenti o similari; pertanto è possibile adottare una strategia di caching [51, 52] basata su più livelli di qualità compatibile con l'approccio soft alla QoS.

5.4.6 Streaming Behaviour

Lo Streaming Behaviour si occupa di trasmettere il contenuto finale all'utente (vedasi figura 5.1, 3.4) e di monitorare variabili relative all'attività di streaming, in particolare il delay utilizzato dal QoS Behaviour per l'adattamento della QoS.

Lo Streaming Behaviour accede tramite memoria primaria al prossimo chunk transcodificato da trasmettere all'utente finale e inizia lo streaming dello stesso in singoli frame.

Il delay corrente viene utilizzato dal QoS Behaviour per un frame o cumulativamente per un gruppo di frame consecutivi.

5.4.7 QoS Behaviour

Il QoS Behaviour si occupa di definire variazioni della QoS nell'attività di transcoding funzionali al mantenimento di delay accettabili per l'utente. Nello specifico si è fatto riferimento in particolare a [53] per definire un fuzzy controller minimale per l'adattamento della QoS in funzione delle variazioni del delay (vedasi sezione 5.6 a pagina 52).

5.5 Euristica di prossimità

L'euristica di prossimità viene utilizzata dal servizio TraS (vedasi sezione 5.2.1 a pagina 36) per selezionare il Computing Element e lo Storage Element adatti a servire una determinata richiesta di un contenuto multimediale V da parte di un generico utente U e, ove non possibile, a rigettarla.

Il servizio TraS dispone di un elenco di Storage Element SE_i per $i = 1..m$, in parte dei quali è a conoscenza della disponibilità del contenuto multimediale V richiesto dall'utente U, e di un elenco di Computing Element CE_j per $j = 1..n$.

Per ogni Storage Element è nota una stima del suo carico espressa come $ioload_i$ ed ottenuta dal servizio SET.

Per ogni Computing Element sono note stime relative al suo carico espresso come $ceload_j$, al numero di processori liberi $freepe_j$ e alla capacità di calcolo di ciascun processore $MIPS_j$, espressa per l'appunto in MIPS, ottenuta dal servizio CET.

Per il contenuto multimediale richiesto V è nota una stima del costo di transcoding T_{C_k} di ciascun chunk C_k calcolata secondo il modello proposto in [54] sulla base di dati tratti dalla letteratura [55, 54, 50] e dagli esperimenti di transcoding condotti (vedasi capitolo 11 a pagina 100). Il repository gestito da TraM sugli algoritmi di transcodifica e le statistiche raccolte durante il monitoraggio dei Transcoding Agent forniscono inoltre un ulteriore input per il raffinamento delle stime dei costi di transcoding basandosi sui costi effettivamente sperimentati dal sistema durante la sua attività.

Per l'intero contenuto multimediale si ha una stima del suo costo complessivo di transcoding espresso come:

$$TC = \sum_{k=1}^{n_V} T_{C_k} \tag{5.1}$$

Per $i = 1..m$ e $j = 1..n$ il servizio TraS dispone di stime delle latenze di rete tra lo Storage Element SE_i e il Computing Element CE_j e tra il Computing Element CE_j e l'utente U. La latenza di rete tra SE_i e CE_j è espressa con $d(SE_i, CE_j)$ e quella tra utente U e CE_j con $d(CE_j, U)$. La latenza di rete relativa all'utente U per ogni coppia di Storage Element e Computing Element (SE_i, CE_j) viene espressa come:

$$d(U, SE_i, CE_j) = d(SE_i, CE_j) + d(CE_j, U) \tag{5.2}$$

Per $i = 1..m$ e $j = 1..n$ viene calcolato il massimo di tale distanza ottenendo:

$$d^* = max(d(SE_i, CE_j) + d(CE_j, U)) \tag{5.3}$$

Per quanto concerne il carico dei Computing Element si effettua una stima della capacità di calcolo residua espressa come:

$$cpuavailable_j = freepe_j * MIPS_j \tag{5.4}$$

La stima del carico $ceload_j$ del Computing Element e il carico $seload_i$ dello Storage Element vengono utilizzati come metrica di carico tramite la somma $ceload_j + seload_i$ espressa dalla formula:

$$load(SE_i, CE_j) = ceload_j + seload_i \tag{5.5}$$

Per $i = 1..m$ e $j = 1..n$ viene calcolato il massimo di tale carico ottenendo:

$$load^* = max(load(SE_i, CE_j)) \tag{5.6}$$

Le due metriche sono combinate nella formula di prossimità:

$$h(SE_i, CE_j, U) = (\frac{load(SE_i, CE_j)}{load^*} + \frac{d(SE_i, CE_j, U)}{d^*}) \tag{5.7}$$

Le coppie di Computing Element e Storage Element sono ordinate dal servizio TraS in modo crescente rispetto ai valori assunti dalla stima di prossimità basata sulla

formula 5.7 e viene scelta la prima coppia che disponga del contenuto richiesto e la cui capacità residua di calcolo (5.4) supporti il costo stimato di transcoding (5.1).

Ove non esista tale coppia il servizio TraS effettua iterativamente il rilassamento dei parametri di transcoding compatibilmente con la minima qualità accettabile definita dall'utente (vedasi sezione successiva) fin quando non seleziona una coppia (CE, SE) adatta o, ove non sia possibile soddisfare i requisiti utente con le risorse disponibili, rifiuta la richiesta.

5.6 Adattamento della QoS

Per l'adattamento dei livelli di Quality of Service percepiti dai fruitori dell'architettura proposta si è fatto riferimento alla letteratura [56, 57, 58, 59, 53] nell'ambito dello streaming end-to-end di contenuti multimediali in ambienti best-effort privi di meccanismi di reservation. In particolare al fine di simulare l'effetto della QoS adaptation sull'architettura proposta si è preso a riferimento l'articolo [53] centrato sul controllo fuzzy [60] scaturito dalla teoria degli insiemi fuzzy di Lofti A. Zadeh [61].

Nell'architettura proposta il modello di adattamento della Quality of Service integra:

1. una funzione di *quality degree*, per mappare tutti i parametri di Quality of Service [62] con una singola metrica

2. un *controllore fuzzy* che si occupa di modificare i parametri di Quality of Service utilizzati nello streaming end-to-end di contenuti multimediali

L'adattamento dei livelli di Quality of Service parte dall'assunto che si è in un ambiente best-effort sia per quanto concerne le risorse di rete che per quanto concerne le risorse di calcolo come può essere considerato un ambiente di Grid Computing general purpose e privo di meccanismi di reservation.

In questo contesto l'adattamento dei livelli di Quality of Service viene a porsi come parte integrante dell'architettura proposta fornendo i meccanismi utili ad aggiustare dinamicamente i requisiti delle richieste di contenuti multimediali sia in termini di risorse computazionali per quanto concerne l'online transcoding sia in termini di risorse di rete per quanto riguarda lo streaming dei contenuti opportunamente transcodificati.

Il modello di adattamento della Quality of Service ha come oggetto le singole richieste di contenuti multimediali (una sorgente invia lo stream ad un singolo destinatario).

Ciascuno stream è caratterizzato dal contenuto multimediale richiesto, dalle capacità del client e dai parametri di Quality of Service a livello applicativo che vengono mappati tramite la funzione di quality degree nell'intervallo [0,1].

5.6.1 Funzione di quality degree

Considerata la molteplicità dei parametri di Quality of Service utilizzabili per i contenuti multimediali, quali frame rate, risoluzione, color depth, si utilizza una funzione di quality degree.

Dati n parametri di Quality of Service ρ_i per $i = 1, 2, \ldots n$ a ciascun parametro è associato un dominio di valori $\Omega_{\rho_i} = [\rho_{i_{min}}, \rho_{i_{max}}]$.

Un livello L di Quality of Service è espresso come una tupla di valori $< \rho_1, \rho_2, \ldots, \rho_n >$.

L'insieme di tutti i possibili livelli di Quality of Service è espresso come:

$$\Omega_{QoS} = \Omega_{\rho_1} \times \Omega_{\rho_2} \times \ldots \Omega_{\rho_n}$$

La funzione di quality degree è una metrica che mappa le tuple di parametri di Quality of Service sull'intervallo $[0, 1]$:

$$\Phi : \Omega_{QoS} \rightarrow [0, 1]$$

A tuple differenti in Ω_{QoS} possono corrispondere valori equivalenti di Φ.

La funzione di quality degree Φ viene calcolata come somma pesata delle utility functions [63, 64] per ciascuna delle dimensioni di Quality of Service considerate.

5.6.2 Controllore fuzzy

Il modello considerato è user-aware in quanto tiene conto dei livelli di Quality of Service percepiti dall'utente e da questo ritenuti accettabili al momento della richiesta di un contenuto multimediale: alla singola richiesta viene associato un valore minimo $QoS_{v_{min}}$ del grado di qualità che l'utente ritiene un requisito minimo accettabile.

$QoS_{v_{min}}$ è un valore compreso nell'intervallo]0,1[ed indica la massima perdita di QoS percepita dall'utente ed accettabile in funzione dei requisiti di QoS specificati

dall'utente nella singola richiesta. Le funzioni di appartenenza della variabile linguistica *qualityloss* sono parametrizzate rispetto a tale valore per adattare il comportamento della QoS adaptation alle singole richieste utente.

Si assume che ciascun frame è inviato con un quality degree di emissione QoS_e e perviene all'utente con un quality degree di ricezione QoS_r e con un delay Δt; il contenuto ricevuto dall'utente viene visualizzato con un quality degree di visualizzazione QoS_v con $QoS_v \leq QoS_r \leq QoS_e$.

L'informazione sul delay Δt è misurata dalla sorgente di emissione che utilizza tale valore ed il valore corrente di QoS_e per calcolare il nuovo quality degree di emissione QoS'_e.

Il nucleo centrale di questa operazione di adattamento della Quality of Service QoS_e è un controllore fuzzy che è stato definito con due variabili linguistiche e sette regole fuzzy.

Le variabili linguistiche utilizzate sono relative a:

1. *delay* (vedasi figura 5.4 a pagina 56)
2. *quality loss* (vedasi figura 5.5 a pagina 57)

Il delay Δt inviato dall'utente alla sorgente di emissione è calcolato come:

$$\Delta t = T_r - T_e$$

ove T_r è la durata dello streaming misurata lato utente fino all'ultimo frame ricevuto, mentre T_e è il tempo in cui tale frame si colloca all'interno del contenuto multimediale.

Il valore di Δt viene diviso per T_e ed è passato come input della variabile linguistica *delay* al controllore fuzzy assumendo:

1. valori compresi in]-1,0[se il frame è stato ricevuto in anticipo da parte dell'utente
2. valori positivi se il frame è stato ricevuto in ritardo da parte dell'utente

La perdita di qualità viene calcolata come output crisp Δq della variabile linguistica *qualityloss* ottenuta dal controllore fuzzy tramite defuzzificazione.

Il nuovo valore di emissione QoS'_e diventa:

$$QoS'_e = QoS_e - \Delta q$$

Se QoS'_e è maggiore di 1 viene forzato come nuovo valore di emissione:

$$QoS'_e = 1.0$$

Se QoS'_e è minore di QoS_{vmin} va allocato un nuovo agente di transcoding e viene forzato come nuovo valore di emissione:

$$QoS'_e = QoS_{vmin}$$

Il valore di emissione QoS_e incide sia sulle risorse di rete utilizzate per lo streaming, secondo il framework concettuale visto nella sezione 4.1, sia sulle risorse computazionali richieste dall'online transcoding (vedasi sezione 5.5).

Tra le tuple di parametri di Quality of Service con lo stesso quality degree viene selezionata quella che richiede minori risorse di rete, secondo una stima di QNET (vedasi sezione 4.1 a pagina 27) basata in particolare sulla stima di frame rate e bitrate, per lo streaming e minori risorse di calcolo per l'online transcoding (vedasi sezione 5.5 a pagina 50).

Le regole fuzzy utilizzate dal controllore sono le seguenti:

1. *if delay is NEGATIVEHIGH then qualityloss is DECREASEDHIGH*
2. *if delay is NEGATIVELOW then qualityloss is DECREASEDLOW*
3. *if delay is NEGATIVE then qualityloss is DECREASED*
4. *if delay is ZERO then qualityloss is STATIONARY*
5. *if delay is POSITIVE then qualityloss is INCREASED*
6. *if delay is POSITIVELOW then qualityloss is INCREASEDLOW*
7. *if delay is POSITIVEHIGH then qualityloss is INCREASEDHIGH*

L'adattamento della quality of service è subordinato all'assenza di altri transcoding agent che possano prendersi carico di parte dell'attività di transcoding quando il

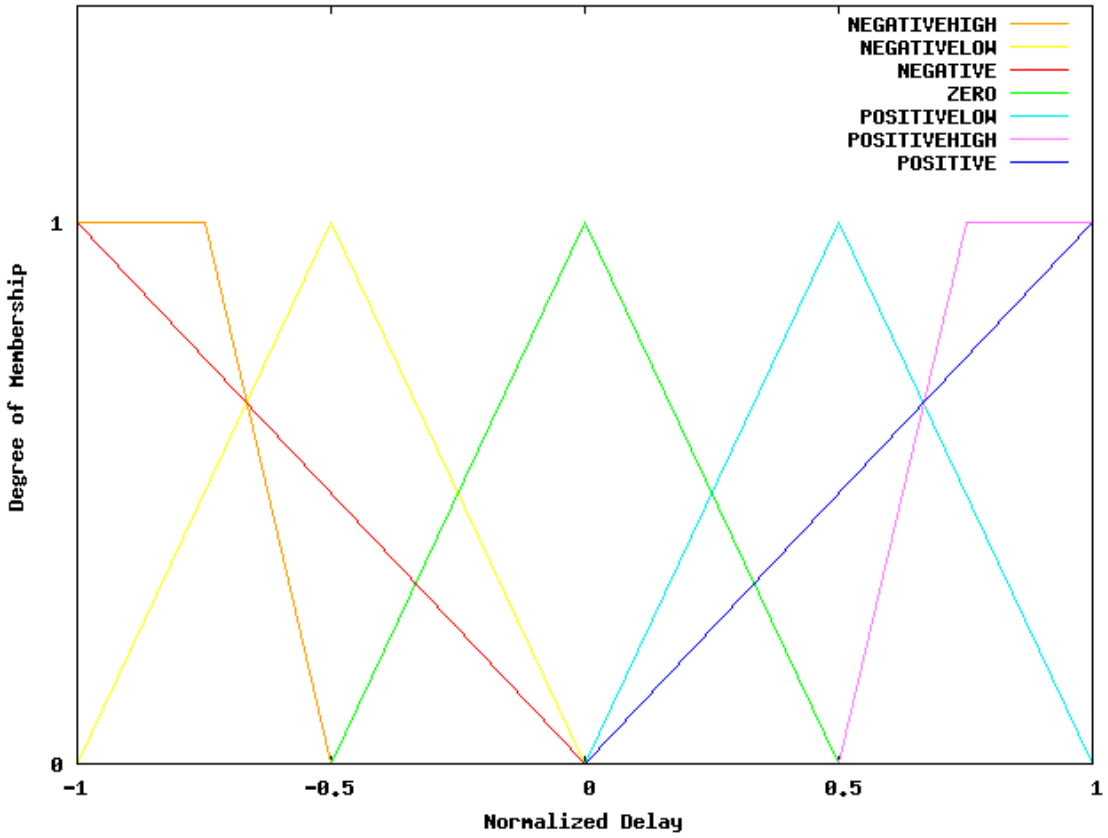

Figura 5.4: Variabile linguistica: delay

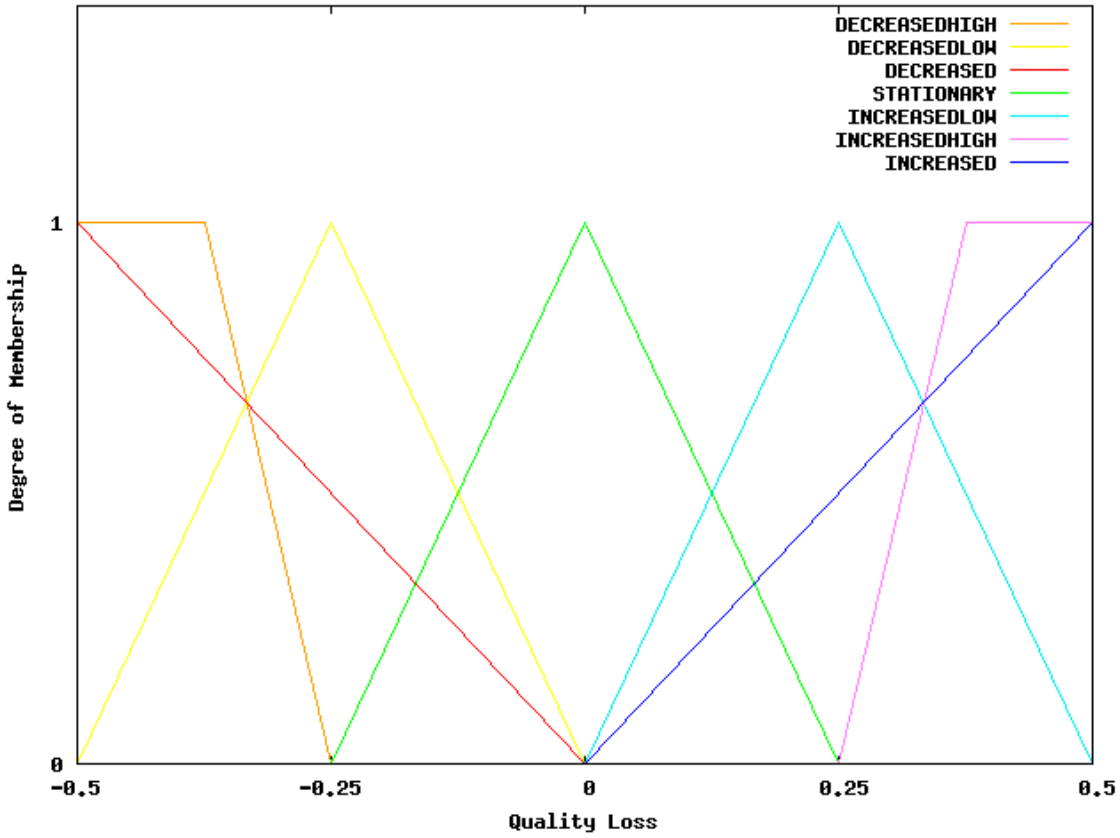

Figura 5.5: Variabile linguistica: quality loss ($QoS_{v_{min}} = 0.5$)

delay è imputabile principalmente ad un sottodimensionamento della capacità di calcolo dei transcoding agent attualmente deputati alla richiesta e nello stesso Computing Element non sono disponibili altri transcoding agent inattivi e compatibili con la specifica attività di transcodifica.

Capitolo 6

Deployment

Come descritto nel capitolo precedente, la soluzione proposta fornisce una gestione soft della Quality of Service senza fare ricorso a risorse computazionali, di storage e di rete dedicate ma sfruttando il potenziale offerto da infrastrutture Grid general purpose preesistenti attraverso la sinergia tra agenti e grid computing.

Se da un lato tale sinergia risulta desiderabile in certi contesti, come quello presentato, un accoppiamento stretto tra middleware multi-agente e middleware di grid computing a livello implementativo sarebbe meno desiderabile ove imponesse un'integrazione diretta col middleware di Grid Computing tramite sue estensioni.

La soluzione proposta si basa su un accoppiamento più lasco tra middleware multi-agente e middleware di grid computing al fine di avere una soluzione architetturale più flessibile ed adattabile a componenti software per agenti e per grid computing tra di loro distinti e che siano più facilmente intercambiabili.

Il deployment a runtime del middleware ad agenti è stato sperimentato su testbed di Grid Computing quali GILDA [1] [65] e COMETA [2], sfruttando le API offerte dal middleware gLite [66, 67] per la sottomissione delle piattaforme ad agenti e degli agenti stessi come normali jobs.

Tale approccio di deployment ha sia i vantaggi brevemente accennati sopra sia gli svantaggi propri di una soluzione non dedicata anche in relazione alla realizzazione

[1] https://gilda.ct.infn.it/
[2] http://www.consorzio-cometa.it/

della sinergia tra due tecnologie nate con motivazioni di fondo diverse ed evolutesi per strade diverse secondo standard differenti.

Ciò non toglie che, anche in relazione alla sperimentazione effettuata per verificare la fattibilità del deployment della sinergia tramite tale accoppiamento lasco, è possibile far coesistere middleware multi-agente con ambienti grid computing preesistenti senza che il middleware per grid computing sia a conoscenza delle peculiarità delle tecnologie ad agenti.

Il deployment del middleware ad agenti in ambito Grid Computing è stato sperimentato sottomettendo i componenti del middleware ad agenti come semplici job sui singoli Computing Element di cui tali componenti ad agenti sfruttano e gestiscono in maniera collaborativa le risorse computazionali e di storage fornite dalla Virtual Organization.

Il middleware ad agenti preso in esame durante la sperimentazione è JADE [15, 68, 16], un framework multi-agente conforme alle specifiche FIPA e caratterizzato da weak mobility (vedasi sezione 2.2 a pagina 11).

Il primo Agent Middleware, l'Agent Platform, viene sottomesso come job sul Computing Element con funzionalità di Resource Broker nell'ambito della Virtual Organization; su tale Agent Platform viene creato il Main-Container contenente i servizi base di AMS (*Agent Management System*) e di DF (*Directory Facilitator*) ed i servizi propri della soluzione proposta ovvero TraS (*Transcoder Selector*), CoT (*Content Tracer*), THub (*Transcoder Hub*), CET (*Computing Element Tracer*), SET (*Storage Element Tracer*) e TraM (*Transcoder Monitor*) (vedasi figura 6.1). Tali servizi sfruttano i servizi base dell'Agent Platform ed in particolare il Directory Facilitator [69] e la funzionalità di Topic Management [44]. Il primo funziona come un servizio di pagine gialle presso cui si registrano gli agenti mentre la seconda fornisce comunicazioni multicast basate su *topic*.

I successivi Agent Middleware sono inviati agli altri Computing Element della Virtual Organization dall'Agent Platform ed all'interno di ciascun Computing Element viene creato un container con i servizi NM (*Network Monitor*), CEM (*Computing Element Monitor*) e GM (*Grid Mediator*) (vedasi figura 6.1).

Il servizio GM (Grid Mediator) fornisce l'accesso alle funzionalità di sottomissione di job, di trasferimento dati, replicazione e di metadata repository resi disponibili dal middleware di Grid Computing; nello specifico, considerato che la sperimentazione è stata effettuata in testbed basati sul middleware gLite, le sue funzionalità sono prototi-

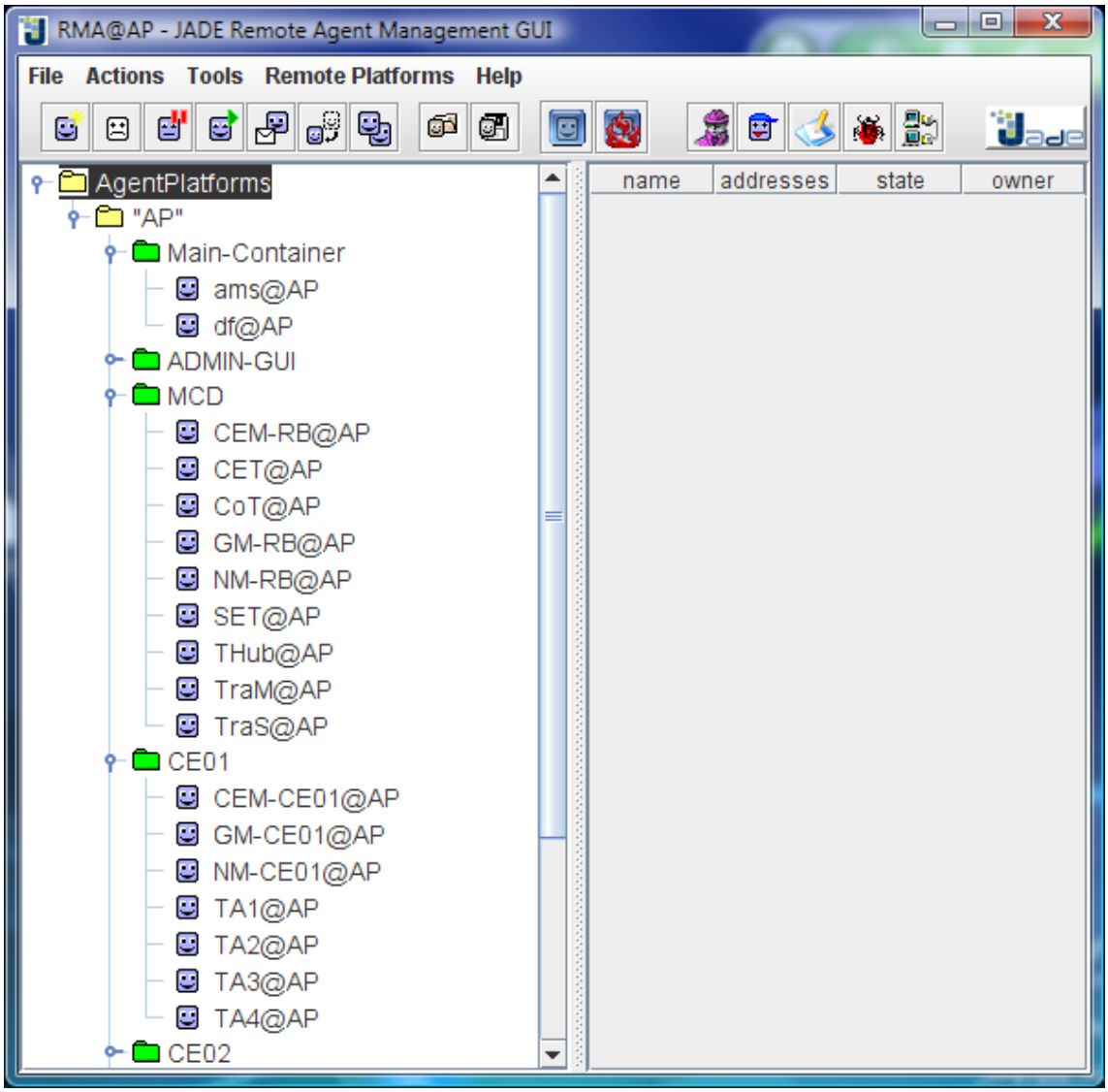

Figura 6.1: Deployment agenti

pate sulla base di API fornite da gLite ed in particolare le API fornite col middleware gLite per la descrizione e sottomissione di jobs e la delega delle credenziali [13, 70, 71]; le funzionalità di trasferimento dati [72, 73, 74, 75] sono state realizzate tramite la libreria COG (*Commodity Grid Kits*) JGlobus [76, 77].

I metadata repository utilizzati da CoT e da TraM sono basati su AMGA [45, 46] ed implementati con l'AMGA Java API.

Per quanto riguarda gli agenti le funzionalità di streaming sperimentate sono state implementate con le API JMF (*Java Media Framework*) [78] e sono basate sul protocollo RTP (*Real-time Transport Protocol*) [79]. Le attività di adattamento della QoS sono state realizzate tramite un fuzzy controller implementato con la libreria *Fuzzy Engine* [80]. Le funzionalità di transcodifica sono state realizzate basandosi su *mencoder* [3] e *transcode* [4].

Una volta che l'Agent Middleware gira su un Computing Element, gli agenti vengono fatti partire sui Working Node[5] del Computing Element sottomettendo gli agenti come jobs ai Working Node del Computing Element facendo ricorso a meccanismi di delega delle credenziali forniti dal middleware di Grid Computing. Gli agenti utilizzati supportano una forma di weak mobility basata sul ricorso al package JADE *jade.domain.mobility*.

La maggior parte dei middleware ad agenti, come per l'appunto JADE, sono sviluppati in Java in particolare per l'alta portabilità della JVM. Un problema che è stato affrontato durante la sperimentazione riguarda la possibilità di far girare in modo omogeneo la JVM nell'ambito di Virtual Organization caratterizzate da risorse computazionali per definizione eterogenee.

La soluzione adottata si basa sul ricorso alle risorse di storage della Virtual Organization per memorizzare copie della JVM per le varie architetture hardware di cui si compone la singola Virtual Organization.

La singola JVM viene avviata su un generico working node della Virtual Organization facendo uso del sottosistema di sottomissione dei job fornito dal middleware di Grid Computing.

[3] http://www.mplayerhq.hu/DOCS/HTML/en/mencoder.html

[4] http://www.transcoding.org

[5] Con il termine Working Node si intende un host del Computing Element disponibile per la sottomissione di job.

L'avvio della JVM è realizzato inviando all'ambiente Grid un job che si compone di uno shell script che:

1. controlla se sia già presente sull'host la versione di JVM richiesta per l'esecuzione del middleware ad agenti

2. ove la JVM non sia quella corretta o non sia affatto presente, verifica a runtime le caratteristiche dell'host su cui viene eseguito e va a selezionare la JVM adatta all'architettura hardware dell'host

Ciascuna JVM e relativo agent middleware sono indirizzati nell'ambito dell'ambiente di Grid Computing attraverso dei logical file names per ottenere un accesso trasparente [71].

La soluzione proposta non assume il requisito che nell'ambiente Grid Computing siano presenti installazioni di JVM necessarie per l'avvio del middleware ad agenti in quanto, sebbene talvolta sui Working Node dei Computing Element di una singola Virtual Organization siano già presenti ambienti di esecuzione predisposti per l'esecuzione di applicazioni Java

1. possono esserci Working Node non provvisti di alcuna installazione JVM

2. queste installazioni non sono necessariamente mantenute sincronizzate tra di loro e possono fare riferimento a versioni Java differenti

Come premesso sopra il deployment prevede che l'Agent Platform sia il primo componente del middleware multi-agente ad essere sottomesso sulle risorse computazionali della Virtual Organization.

Una volta che l'Agent Platform è attiva vengono avviati i servizi propri dell'architettura che sono stati presentati nel precedente capitolo.

Va notato che, sebbene nella sperimentazione si sia fatto ricorso al deployment a runtime delle piattaforme ad agenti per quanto riguarda l'Agent Platform è desiderabile che essa sia installata come applicazione disponibile 24/7 nell'ambito della Virtual Organization; ciò richiederebbe un intervento su un singolo Computing Element ma non è fattibile lato provider senza un intervento dei gestori della Grid. Un'altra alternativa considerata è quella di dislocare l'Agent Platform su un server esterno alla Grid, tuttavia entrambe le alternative sono state considerate poco rilevanti per la ricerca.

La sperimentazione su testbed della soluzione proposta, realizzata in forma prototipale, è stata finalizzata principalmente a studiare la fattibilità del deployment e dell'interoperabilità degli agenti con un middleware di Grid Computing e ottenere misure su cui basare la simulazione tramite la libreria GAP (Grid Agents Platform) (vedasi capitolo 9 a pagina 80) al fine di ottenere feedback qualitativi e quantitativi in condizioni controllate e ripetibili (vedasi capitolo 12 a pagina 103).

Capitolo 7

Letteratura

Relativamente alla soluzione proposta in questa tesi, pochi sono i lavori correlati che sfruttino le tecnologie di Grid Computing e ancor meno una sinergia di queste con tecnologie multi-agente; sono da richiamare in particolare gli approcci presentati in [81, 82, 83, 84, 85, 86, 87, 88].

In [81] gli autori propongono una soluzione proprietaria di service grid specializzata nella distribuzione di contenuti multimediali. In tale soluzione si concentrano sull'aspetto della distribuzione e caching dei contenuti multimediali in locazioni prossime all'utente secondo un modello di distribuzione gerarchico su tre livelli: providers dei contenuti, media centers regionali e media centers residenziali che agiscono come proxy cache per i contenuti più popolari. Nell'articolo viene effettuata un'analisi delle potenzialità della soluzione proposta tramite simulazione. La soluzione proposta secondo le simulazioni effettuate dagli autori fornisce una riduzione del carico presso i providers di contenuti e una riduzione significativa del traffico di rete complessivamente generato.

In [84] gli autori propongono una soluzione distribuita per il *video on demand* basata sul ricorso alle tecnologie di Grid Computing. In particolare gli autori ricorrono al Grid Computing per realizzare la distribuzione dei contenuti multimediali sugli Storage Element tramite tecniche di splitting lineare dei contenuti su più nodi e il request routing tramite uno streaming proxy che si occupa di coordinare il recupero e l'ordinamento dei chunks secondo una policy di schedulazione mirata al bilanciamento del carico dei server di storage.

In [82] gli autori propongono mmGrid, un middleware di Grid Computing pen-

sato specificamente per il supporto ad applicazioni multimediali. mmGrid fornisce alcune delle principali funzionalità proprie di un ambiente di Grid Computing ovvero autenticazione, servizio di directory, ricerca delle risorse, scheduling e logging sebbene in modo non standard senza fare riferimento quindi ad esempio a GSI (Grid Security Infrastructure) [89] per quanto concerne l'autenticazione e supportando in modo minimale funzionalità di scheduling.

In [85] gli autori propongono una soluzione per la distribuzione di contenuti multimediali che sfrutta gli ambienti di Grid Computing per fornire le risorse necessarie allo storage e allo streaming dei contenuti; nell'articolo gli autori presentano i risultati sperimentali di un'implementazione prototipale basata su Globus e mostrano l'efficienza di tale soluzione.

In [87] gli autori partono dall'assunto che vi siano benefici nello strutturare una CDN sotto forma di servizi Grid trattando gli ambienti di Grid Computing come strumento per l'outsourcing delle risorse necessarie al deployment di una CDN da parte di provider concentrandosi sull'offerta di contenuti multimediali secondo un approccio di CDN. Inoltre gli ambienti Grid forniscono meccanismi di monitoraggio e allocazione delle risorse, di sicurezza in termini di autenticazione e autorizzazione e di replicazione dei contenuti che sono funzionali ad un'architettura di CDN. Infine gli autori ritengono che la visione OGSA [3, 90] basata su Grid Web Services possa fornir una semplificazione del processo di sviluppo, configurazione ed adattamento del software impiegato nel fornire funzionalità di CDN al di sopra di risorse numerose ed eterogenee proprie di una Grid.

In [83, 86] gli autori presentano SGMS (Scalable Grid-based Multimedia Server), una soluzione di multimedia streaming verso dispositivi eterogenei con supporto al transcoding per l'adattamento dei contenuti all'eterogeneità propria di tali dispositivi in termini di visualizzazione e riproduzione audio. La loro soluzione sfrutta gli ambienti di Grid Computing per supportare gli elevati costi computazionali propri delle attività di transcodifica e per memorizzare contenuti multimediali di alta qualità negli Storage Element. L'accesso ai contenuti avviene secondo un'architettura di VS (Virtual Server) che nasconde l'interazione tra la UI dell'utente basata su browser e i servizi distribuiti in ambiente Grid. La soluzione presentata dagli autori prevede un servizio di Media Adaptation basato su transcoding parallelo distribuito tra più nodi ciascuno dei quali si occupa della transcodifica di parte del contenuto opportunamente suddiviso in grossi chunk. Tale

soluzione non presenta funzionalità di CDN in quanto i tempi di risposta variano tra circa 50 e 400 secondi necessari alla completa transcodifica del contenuto prima che questo sia reso disponibile tramite streaming all'utente finale.

In [88] gli autori considerano i benefici derivabili dalla convergenza di tre tecnologie ovvero sistemi multi-agente, grid computing e P2P, per realizzare una CDN. L'architettura da loro proposta, denominata UPGRADE-CDN, presenta parallelismi con la soluzione presentata in questa tesi. UPGRADE-CDN è basata su componenti di cui tre sono chiave ovvero *AMonitor*, *GRedirector* e *Co-Delivery*.

AMonitor, sviluppato con tecnologia ad agenti, ha il compito di monitorare lo stato dei replica servers, la rete di accesso da parte degli utenti alla CDN e l'infrastruttura di rete interna alla CDN tra i vari replica servers.

GRedirector, basato su tecnologie di Grid Computing, fornisce funzionalità di request routing basate su un approccio di DNS-redirection [91, 35, 39] secondo scelte informate dalle statistiche di monitoraggio fornite dal componente AMonitor.

Co-Delivery, basato sul framework P2P JXTA [92], si occupa della distribuzione collaborativa dei contenuti agli utenti tramite clustering dei replica servers e scelte di replicazione informate su tali gruppi.

La soluzione proposta in questa tesi si differenzia da UPGRADE-CDN in tre aspetti in particolare. In UPGRADE-CDN non viene utilizzato l'online transcoding per l'adattamento di contenuti e le risorse Grid sono essenzialmente utilizzate per fornire una dislocazione ottimale dei contenuti sugli Storage Element della Grid mentre gli agenti sono utilizzati esclusivamente per il monitoraggio delle risorse di storage e di rete; inoltre è richiesta una stretta integrazione col middleware Grid tramite lo sviluppo di servizi Grid distribuiti gerarchicamente e in modo diffuso sulla Grid per l'implementazione del componente GRedirector deputato al request routing; tuttavia è previsto il ricorso al paradigma P2P, aspetto questo che rende UPGRADE-CDN molto interessante estendendo la sinergia a tre tecnologie differenti.

Per quanto concerne sinergie in senso lato tra tecnologie multi-agente e di Grid Computing esiste una messe più ampia di lavori. Il ricorso a sinergie tra tecnologie multi-agente e Grid Computing si ritrova principalmente adoperato per il *resource discovery* [93, 94, 95], per lo *scheduling* [93, 96, 97, 98, 99, 100, 101, 102, 103], per la sicurezza [104, 105], per la negoziazione delle risorse [106, 107, 108, 109], per la *fault toulerance*

[110, 111] e l'astrazione dell'ambiente di Grid Computing [112, 93, 113].

In [114] viene presentato CoABS (Control of Agent Based System) un middleware di Grid Computing di tipo SOA (Service-Oriented Architecture) [11], basato su tecnologie multi-agente, sviluppato dalla DARPA, incentrato su un MOM (Message-Oriented Middleware) caratterizzato dal supporto allo scambio asincrono, persistente e sicuro di messaggi per fornire le funzionalità proprie di una Grid con capacità autonomiche [115].

In [112] gli autori ricorrono alle tecnologie ad agenti per astrarre e modellare l'eterogeneità tra Grid esistenti al fine di delegare agli agenti il compito di interagire con le peculiarità dei singoli middleware e di fornire all'utente e al programmatore un modello uniforme di Grid permettendo loro di concentrarsi sul problema da risolvere sfruttando la potenza di calcolo offerta dagli ambienti di Grid Computing.

In [116] gli autori propongono il ricorso al framework multi-agente JADE per la modellazione e la prototipazione di workflow in ambienti di Grid Computing e per l'astrazione delle sottostanti peculiarità del middleware di Grid Computing e del sistema di workflow.

In [93] gli autori propongono ARMS (Agent-based Resource Management System), un'architettura di gestione delle risorse per ambienti di Grid Computing basata su tecnologie multi-agente; in tale lavoro gli autori sfruttano le tecnologie multi-agente per astrarre le risorse della Grid tramite agenti associati alle specifiche risorse, per monitorare l'utilizzo delle risorse al fine di effettuare previsioni sul carico di risorse necessarie alle singole applicazioni e per effettuare meta-scheduling delle risorse; inoltre gli autori presentano ARMSim, un simulatore delle prestazioni di ARMS.

In [94] gli autori studiano le soluzioni di gestione delle risorse in ambito Grid Computing proponendo un sistema multi-agente per un'efficiente ricerca delle risorse anche in caso di condizioni di picco del carico delle risorse sfruttando le capacità collaborative degli agenti.

In [104] gli autori propongono una sinergia tra agenti e Grid Computing principalmente mirata alla formazione dinamica di Virtual Organization tenendo conto di aspetti di fiducia, sicurezza e rispetto dei livelli di servizio, attraverso meccanismi di calcolo e memorizzazione distribuita della reputazione per il tramite di agenti.

In [106] l'autore presenta un'architettura ad agenti per la gestione di risorse in ambienti di Grid Computing concentrandosi su protocolli di negoziazione delle risorse

basati su modelli economici.

In [96] gli autori propongono il ricorso a tecnologie multi-agente per la gestione delle risorse in ambienti di Grid Computing al fine di supportare ambienti Grid più dinamici ed orientati ad applicazioni interattive.

In [110] gli autori propongono un framework multi-agente ottimizzato per ambienti di Grid Computing, al fine di gestire in modo collaborativo i workflow e ottimizzare le prestazioni complessive e la tolleranza ai guasti attraverso migrazioni tra Computing Element distinti.

In [108] gli autori studiano un modello di collaborazione tra agenti in un ambiente di Grid Computing mirato a sfruttare le capacità collaborative e di rappresentazione delle azioni, obiettivi e utilità degli agenti per ottenere politiche emergenti e globalmente efficienti nell'utilizzo delle risorse della Grid e presentano AGrIP, una piattaforma basata su tale modello e su loro precedenti lavori sulla sinergia tra tecnologie multi-agente e ambienti di Grid Computing [117, 118].

In [107] gli autori propongono una rappresentazione degli utenti e delle risorse di un ambiente di Grid Computing basato sulla collaborazione tra agenti e finalizzato a massimizzare l'utilità degli utenti e il guadagno dei fornitori di risorse della Grid secondo modelli economici.

In [111] viene presentato AgentTeamwork, un middleware di Grid Computing che sfrutta tecnologie ad agenti mobili per coordinare l'utilizzo delle risorse da parte dei jobs in modo decentralizzato fornendo un bilanciamento dinamico delle risorse utilizzate, tolleranza ai guasti e forme di checkpointing per jobs distribuiti basati su implementazioni di MPI[1] [119] in Java.

In [109] presenta un modello di negoziazione ad agenti per servizi applicativi in ambienti di Grid Computing mirato ad ottimizzare l'utilità di utenti e fornitori di servizi nell'allocazione delle risorse.

In [97] viene proposta una infrastruttura di calcolo basata sulla sinergia tra agenti e paradigmi di Grid Computing mirata ad ottimizzare l'allocazione delle risorse tramite politiche di scheduling locali gestite dagli agenti più vicini alle risorse e coordinamento tra agenti per ottenere un sistema di scheduling ad alto livello più efficiente e flessibile nel tollerare fluttuazioni dinamiche della disponibilità delle risorse nella Grid.

[1] http://www-unix.mcs.anl.gov/mpi/

In [95] viene proposto un middleware di Grid Computing basato su tecnologie ad agenti mobili e mirato a sfruttare i cicli idle dei desktop, aumentando un ambiente tradizionale di Grid Computing; gli agenti vengono utilizzati per monitorare le risorse disponibili sui singoli nodi e bilanciarne il grado di utilizzo tramite i servizi di mobilità.

In [98] è proposta un'architettura di scheduling delle risorse di un ambiente di Grid Computing basato su agenti ed algoritmi genetici che mirano a minimizzare i tempi di esecuzione dei jobs.

In [99] gli autori propongono lo scheduling di jobs su Grid per il tramite di sistemi multi-agente attraverso protocolli di negoziazione orientati ai livelli di servizio e basati su un modello gerarchico di interazione tra agenti che schedulano le risorse sui singoli Computing Element e agenti che mantengono una visione di alto livello sulle risorse distribuite geograficamente.

In [105] gli autori propongono un potenziale approccio alle problematiche di sicurezza inerenti gli ambienti di Grid Computing presentando un modello di IDS (Intrusion Detection System) basato sugli agenti e su meccanismi mitigati dall'immunologia.

In [100] viene proposto un modello multi-agente per ambienti di Grid Computing, finalizzato al problema dell'ottimizzazione delle risorse in scenari caratterizzati da alta variabilità della connettività attraverso partizionamento delle risorse in cluster logici per lo scheduling di jobs caratterizzati da elevato parallelismo.

In [101] presenta un middleware di Grid Computing basato su tecnologie multi-agente incentrato sullo scheduling decentralizzato delle risorse della Grid per il tramite di agenti e mirato all'ottimizzazione delle latenze di rete.

In [102] viene presentato ASF (Agent-based Scheduling Framework), un framework multi-agente per l'utilizzo di risorse Grid non dedicate e poco affidabili, che utilizza un approccio gerarchico basato su agenti di meta-scheduling e agenti di scheduling locale alle risorse ottimizzando i tempi di completamento dei jobs.

Nell'articolo [120] viene presentata un'architettura di gestione dei servizi applicativi in ambito Grid basata su agenti software che astraggono le risorse e i servizi che girano su tali risorse per il tramite di agenti.

In [113] viene proposta l'architettura MA-MSMA (Mobile Service Management Architecture based on Mobile Agent) per la ricerca e l'accesso a servizi mobili della Grid

per il tramite di agenti mobili che forniscono un'astrazione delle risorse e dei servizi dell'ambiente di Grid Computing.

In [121] viene presentato un approccio alla tolleranza ai guasti di tipo proattivo, basato su tecnologie multi-agente, che sfrutta gli agenti per monitorare le condizioni dell'hardware, della rete e del software e l'utilizzo delle risorse, per predire guasti al fine di ridurre i tempi di indisponibilità, per ottimizzare l'utilizzo delle risorse di rete e le tempistiche relative alle operazioni di ripristino di jobs interrotti prematuramente.

In [103] viene proposto un modello gerarchico ad agenti per la ricerca, il monitoraggio e l'allocazione di web services con requisiti di QoS in ambienti di Grid Computing conformi ad OGSA [90, 12].

Parte III

Simulazione

Il ricorso alla simulazione da parte dei ricercatori per lo studio di scenari complessi è sempre più frequente per svariati motivi quali la possibilità di ottenere un feedback sulla fattibilità, sul comportamento e sulle prestazioni di un sistema senza la sua effettiva implementazione e di ridurre le tempistiche necessarie ad un'analisi qualitativa e quantitativa dei problemi modellati.

Per quanto concerne l'ambito dei sistemi distribuiti e nello specifico degli ambienti di Grid, è possibile scegliere tra più strumenti di simulazione. Lo sviluppo di strumenti di simulazione non è di per sé un'attività semplice soprattutto se mirata al riutilizzo e non al singolo utilizzo o alla singola ricerca. Modelli di simulazione realistici sono spesso difficili da progettare e mantenere anche in relazione al fatto che non di rado i costi di sviluppo di strumenti di simulazione per modelli complessi sono comparabili a quelli necessari per l'implementazione del sistema modellato.

L'esistenza di strumenti di simulazione sufficientemente realistici ed adatti alla modellazione di classi di modelli è auspicabile in quanto:

1. permette di concentrarsi sul sistema da modellare rendendo più agevole e veloce la creazione di un modello realistico per estrapolare informazioni utili sul sistema prima della sua effettiva implementazione

2. nell'ambito della ricerca scientifica l'utilizzo di strumenti di simulazione comuni permette di ottenere un confronto tangibile tra lavori condotti da gruppi di ricerca differenti

Nell'ambito dello studio dei tools di simulazione per ambienti di Grid Computing (vedasi capitolo 8 a pagina 74), avendo in mente la sinergia tra tecnologie ad agenti ed ambienti di Grid Computing, non mi è stato possibile disporre di uno strumento di simulazione pronto all'uso per la modellazione di scenari caratterizzati da una tale sinergia. Pertanto, dopo una fase iniziale di studio degli strumenti disponibili, in funzione dell'attività che mi prefiggevo di svolgere, ho ritenuto opportuno sviluppare una libreria Java, denominata GAP (Grid Agents Platform) (vedasi capitolo 9 a pagina 80), pensata come estensione di un tool di simulazione esistente ovvero GridSim [122] (vedasi sezione 8.0.4 a pagina 78), basato a sua volta sul tool di simulazione ad eventi discreti SimJava2 [123, 124, 125, 126] (vedasi sezione 8.0.5 a pagina 79).

Capitolo 8

Software di simulazione disponibile

Gli ambienti di Grid Computing sono sistemi distribuiti per certi versi più difficili da modellare rispetto ad altre classi di sistemi distribuiti in relazione al fatto che:

1. sono distribuiti geograficamente
2. coinvolgono una quantità di risorse computazionali e di storage talmente eterogenee e numerose da essere potenzialmente non tracciabili in modo esaustivo da parte dell'utente generico dell'ambiente di Grid Computing
3. attraversano domini amministrativi con politiche di utilizzo differenti
4. sono caratterizzati da una complessità intrinseca dovuta alla molteplicità di funzionalità e relative componenti hardware e software necessarie a soddisfare i requisiti minimali di un sistema distribuito di tal fatta

In particolare, tra gli elementi di complessità di un ambiente di Grid Computing sono da considerare:

1. individuazione delle risorse computazionali e di memorizzazione di massa
2. selezione e utilizzo delle risorse computazionali per l'esecuzione di jobs distribuiti
3. impatto delle prestazioni di rete sull'esecuzione dei jobs
4. problematiche di sicurezza
5. eterogeneità delle risorse coinvolte

6. requisiti di tolleranza ai guasti, scalabilità, concorrenza e trasparenza

La progettazione di middleware di Grid Computing o anche semplici estensioni dei middleware esistenti richiede pertanto il riconoscimento di tali elementi di complessità e l'individuazione di modalità e strumenti per attaccare tale complessità.

Il gran numero di risorse hardware coinvolte, la loro eterogeneità, la loro distribuzione geografica, la variabilità dei carichi di lavoro delle risorse computazionali, delle latenze di rete e del traffico di fondo della rete in presenza di altri carichi di lavoro, considerato che gli ambienti di Grid Computing non sono ambienti dedicati, rendono difficile ottenere ambienti controllabili e situazioni ripetibili, condizioni queste necessarie a condurre ricerche con risultati realistici e coerenti.

Una soluzione usata non di rado per testare middleware di Grid Computing o estensioni degli stessi sono i testbed, ovvero il deployment di tali sistemi di Grid Computing su risorse hardware e software con finalità di sperimentazione. L'utilizzo dei testbed tuttavia se da un lato si mostra utile per un'analisi quantitativa e qualitativa del sistema e come strumento per testare con utenti reali il sistema prima del suo deployment in produzione al fine di ottenere feedbacks e correggere difetti del software, ha come svantaggi il fatto che i testbed:

1. non forniscono adeguata ripetibilità e controllabilità
2. sono caratterizzati da dimensioni ridotte rispetto ai sistemi di produzione pur presentando elevati costi di gestione

Nel selezionare i tool di simulazione per ambienti di Grid Computing che ho ritenuto adeguati per un'analisi qualitativa e quantitativa sufficientemente realistica per modellare il comportamento e le prestazioni di una sinergia tra Agenti Mobili e Grid Computing, ho considerato i seguenti requisiti:

1. Usabilità
2. Velocità della simulazione
3. Configurabilità
4. Estensibilità

5. Raccolta di misure e loro rappresentazione

6. Ripetibilità e controllabilità degli esperimenti di simulazione

7. Accesso a strumenti di debugging e di validazione

Lo studio degli strumenti di simulazione si è basato sull'utilizzo di tali strumenti per modellare scenari più o meno semplici e sullo studio della relativa letteratura di cui tali strumenti sono solitamente corredati in quanto solitamente frutto del lavoro di ricercatori nell'ambito del Grid Computing o dei sistemi distribuiti in senso lato.

In particolare ho trovato illuminante la lettura dell'articolo [127] che identifica tre categorie di classificazione dei sistemi di simulazione per sistemi distribuiti ovvero:

1. Proprietà di utilizzo

2. Proprietà di simulazione

3. Proprietà di progettazione

Per quanto concerne le proprietà di utilizzo, tale tassonomia distingue tra simulatore ed emulatore. Un simulatore modella e rappresenta un sistema attraverso le sue proprietà essenziali a ottenere misure qualitative e quantitative sufficientemente realistiche sul sistema stesso. Un emulatore invece implementa il sistema stesso emulandone in toto o in parte le risorse da questo utilizzate quali risorse di calcolo, di memorizzazione e di rete, quale ad esempio MicroGrid [128, 129, 130] che emula il middleware Globus.

Per quanto concerne le proprietà di simulazione risulta essere determinante la modalità di rappresentazione del tempo e del comportamento della simulazione. La raccolta di misure temporali sul sistema modellato è importante per la raccolta di informazioni sulle prestazioni del sistema stesso. Il comportamento della simulazione deve essere deterministico se si vogliono ottenere esperimenti ripetibili.

Per quanto concerne la progettazione del simulatore l'aspetto cardine da considerare è l'engine di simulazione a che le simulazioni siano caratterizzate da un sufficiente grado di realismo, ripetibilità e controllabilità.

La classe di engine di simulazione più utilizzata per modellare e simulare ambienti di Grid Computing è quella dei DES (Discrete Event Simulators).

I DES adottano reti di code ove differenti code di eventi aspettano la loro attivazione e si suddividono in:

1. trace-driven, ove la simulazione procede sulla base di tracce di eventi ottenute dal monitoraggio di sistemi reali
2. time-driven, ove la simulazione procede ad intervalli regolari
3. event-driven, ove la simulazione è guidata da eventi che possono verificarsi in qualsiasi istante

I DES event-driven sono più efficienti rispetto ai time-driven e forniscono le basi per ottenere DES trace-driven; infatti solitamente i DES sono sia event-driven che trace-driven.

Altri elementi da considerare per quanto concerne la scelta di un simulatore sono:

1. l'ambiente di programmazione
2. l'ausilio alla modellazione visuale e non
3. il supporto al debugging

Per quanto concerne l'ambiente di programmazione il tool di simulazione può fornire un linguaggio di programmazione ad alto livello, essere basato sullo stesso linguaggio di programmazione con cui è realizzato il tool di simulazione o fornire un accesso ibrido ad entrambi. L'accesso a strumenti di modellazione visuale è sicuramente vantaggioso riducendo la curva di apprendimento e fornendo un feedback visivo sul modello da simulare. Il debugging si rivela utile in fase di simulazione del sistema modellato per individuare e correggere difetti del modello costruito.

Tra gli strumenti di simulazione da me studiati ce ne sono tre su cui ho effettuato le mie valutazioni per arrivare ad una scelta, ovvero:

1. SimGrid
2. GridSim
3. SimJava

8.0.3 SimGrid

SimGrid [131, 132, 133, 134] è implementato in C/C++ ed è pensato per lo studio di scenari di scheduling su cluster e ambienti eterogenei di Grid Computing.

SimGrid è un tool event-driven e trace-driven basato su un engine di simulazione ad eventi discreti, caratterizzato da una rappresentazione realistica [135] del traffico di rete, molto usato per la simulazione di ambienti di Grid Computing anche in funzione dei contributi ed estensioni che sono stati resi pubblici nel tempo [1], che però è pensato per scenari con singolo client di sottomissione dei jobs.

SimGrid non supporta nativamente la simulazione di Data Grids [136] e relativi aspetti di distribuzione, replicazione e reperimento di eseguibili e dati dalla Grid, centrali nella soluzione qui presentata.

8.0.4 GridSim

GridSim [122, 137, 138, 139] è una libreria Java sviluppata nell'ambito delle ricerche del team di Rajkumar Buyya[2] su modelli di gestione delle risorse e di scheduling su Grid basati su paradigmi economici [139].

GridSim, un completo ed estensibile tool di simulazione per ambienti Grid Computing, è basato su SimJava2 di cui ben sfrutta sia l'engine di simulazione che le potenzialità statistiche e di reporting.

GridSim offre supporto alla modellazione e simulazione delle principali caratteristiche di un ambiente di Grid Computing quali:

1. risorse computazionali
2. risorse di storage
3. topologia della rete
4. cataloghi di replica
5. sottomissione e checkpointing dei jobs

[1] http://simgrid.gforge.inria.fr/doc/contrib.html
[2] http://www.buyya.com/

GridSim supporta a pieno la simulazione del networking [140, 141] e la simulazione di scenari di Data Grids [136] ponendosi come tool completo general purpose alternativo a tools specifici per la simulazione di Data Grids, quali ad esempio OptorSim [142].

8.0.5 SimJava2

SimJava2 [123, 124, 125, 126] è un libreria Java che fornisce un engine di simulazione puro di tipo DES (Discrete Event Simulator) pensato per lo studio di reti di code ed incentrato su entità per modellare risorse e processi di un sistema ed eventi per simulare lo scambio di messaggi tra dette entità.

Fornisce un ampio package di distribuzioni probabilistiche utili in particolare a modellare tempi di interarrivo degli eventi e tempi di servizio delle entità mantenendo al contempo un comportamento deterministico della simulazione per la ripetibilità degli esperimenti [126].

Fornisce strumenti per l'analisi dell'output basati su tecniche di *batch means* e *indipendent replications*. La tecnica dei batch means, è basata su una singola simulazione in cui le misure raccolte sono divise in lotti sui quali sono calcolate le medie. Rispetto alla tecnica delle replicazioni indipendenti è più efficiente ma non è esente da correlazioni tra i lotti della simulazione. La tecnica delle replicazioni indipendenti si basa essenzialmente sulla ripetizione della simulazione variando i semi dei generatori di numeri casuali utilizzati dalle entità modellate coniugando così apparente indeterminismo della simulazione con la sua ripetibilità, tecnica quest'ultima scelta per le simulazioni da me effettuate.

Capitolo 9

GAP Modeling and Simulation Toolkit

GAP è una libreria Java sviluppata come estensione alla libreria GridSim e pensata per modellare e simulare scenari basati sulla sinergia tra tecnologie di MAS (Multi-Agent Systems) e di Grid Computing.

GAP permette agli utenti di delegare la sottomissione di jobs a servizi ed agenti di una generica piattaforma ad agenti al fine di sfruttare la prossimità degli agenti alle risorse della Grid e le loro peculiarità di entità intelligenti, autonome e collaborative nonché la loro capacità di migrare in modo weak o strong tra host differenti.

Lo sviluppo di GAP è cominciato da una necessità di riordinare alcune classi inizialmente sviluppate per la simulazione di agenti in ambiente Grid nell'ambito dello scenario di distribuzione di contenuti multimediali presentato in questa tesi, al fine di renderne più agevole la modellazione, con una separazione delle funzionalità di base della piattaforma ad agenti dai servizi da essi forniti nell'ambito della soluzione proposta, e per fornire un tool utilizzabile per scenari accomunati esclusivamente da una sinergia in senso lato tra tecnologie multi-agente e ambienti di Grid Computing.

GAP è strutturata in una serie di package principali ovvero:

1. *gap.agents*
2. *gap.constants*
3. *gap.distributions*

4. *gap.factories*
5. *gap.grid*
6. *gap.messages*
7. *gap.simulation*
8. *gap.users*
9. *gap.util*
10. *gap.xml*

Il package *gap.agents* raggruppa classi con differenti responsabilità, tra cui quella della rappresentazione degli agenti (per la gerarchia di tali classi vedasi figura 9.3 a pagina 85):

tutti gli agenti sono derivati dalla classe base *gap.agents.AbstractAgent* che a sua volta estende la classe *gridsim.datagrid.DataGridUser*; quest'ultima fornisce metodi per la gestione delle Gridlets (nome utilizzato in GridSim come sinonimo di Job) e delle DataGridlets (jobs caratterizzati dall'utilizzo di Storage Element per reperire o memorizzare dati di input o output della computazione); fornisce inoltre metodi per la gestione da parte degli utenti delle repliche sugli Storage Element. La classe *gap.agents.AbstractAgent* mantiene per ciascuna istanza di agente le informazioni essenziali su di essa ovvero: Computing Element su cui l'agente è presente, Agent Platform dell'agente, dimensione dell'agente in bytes per simulare l'overhead introdotto dal trasporto dell'agente in caso di sottomissione o di migrazione, tipo di agente, stato dell'agente, identificatore univoco dell'agente, invariante in seguito a migrazioni dello stesso, l'AgentHistory (vedasi figura 9.1 a pagina 83) che mantiene uno storico delle informazioni caratterizzanti l'agente nel corso del suo intero ciclo di vita migrazioni incluse.

la classe *gap.agents.Agent* estende la classe *gap.agents.AbstractAgent* e si occupa di gestire il ciclo di vita dell'agente in relazione ai suoi stati e alle sue interazioni con il Directory Facilitator prevedendo le modalità di sottomissione, sospensione, ripristino,

disattivazione dell'agente e migrazione verso altri host notificando tali cambiamenti di stato dell'agente al Directory Facilitator.

la classe *gap.agents.GridAgent* estende la classe *gap.agents.Agent* ed è responsabile per la sottomissione, il monitoraggio e il checkpointing di jobs da parte dell'agente

infine la classe *gap.agents.PluggableAgent* estende la classe *gap.agents.GridAgent* permettendo di definire nuove classi di agenti con i loro Behaviours attraverso plugging degli stessi (vedasi figura 9.2 a pagina 84). Questa classe fornisce la base per la modellazione e simulazione di nuovi tipi di agenti.

Il package *gap.agents.gridlets* si occupa della delega agli agenti nella gestione dei jobs per la loro sottomissione sulla Grid e contiene l'astrazione di JobScheduler (vedasi figura 9.4 a pagina 86).

Il package *gap.agents.services* si occupa della gestione dei servizi che girano sull'Agent Platform e contiene le classi astratte *Service* e *PlatformService* per definire nuovi servizi relativi rispettivamente ad un singolo Computing Element o ad un'intera Virtual Organization (vedasi figura 9.5 a pagina 87).

Il package *gap.constants* contiene tre classi:

1. la classe *AgentStates* utilizzata per la caratterizzazione degli stati degli agenti ovvero:
   - INITIATED, agente appena creato
   - ACTIVE, agente attivo
   - SUSPENDED, agente sospeso, nessun behaviour in esecuzione
   - WAITING, agente bloccato in attesa di un evento
   - DELETED, agente distrutto
   - TRANSIT, stato in cui entra un agente durante la migrazione verso una nuova locazione
2. la classe *EntityTypes* utilizzata per la caratterizzazione delle entità che interagiscono in modo attivo con gli agenti ovvero gli utenti e gli agenti stessi
3. la classe *Tags* utilizzata per la caratterizzazione degli eventi scambiati tra le entità in relazione a:

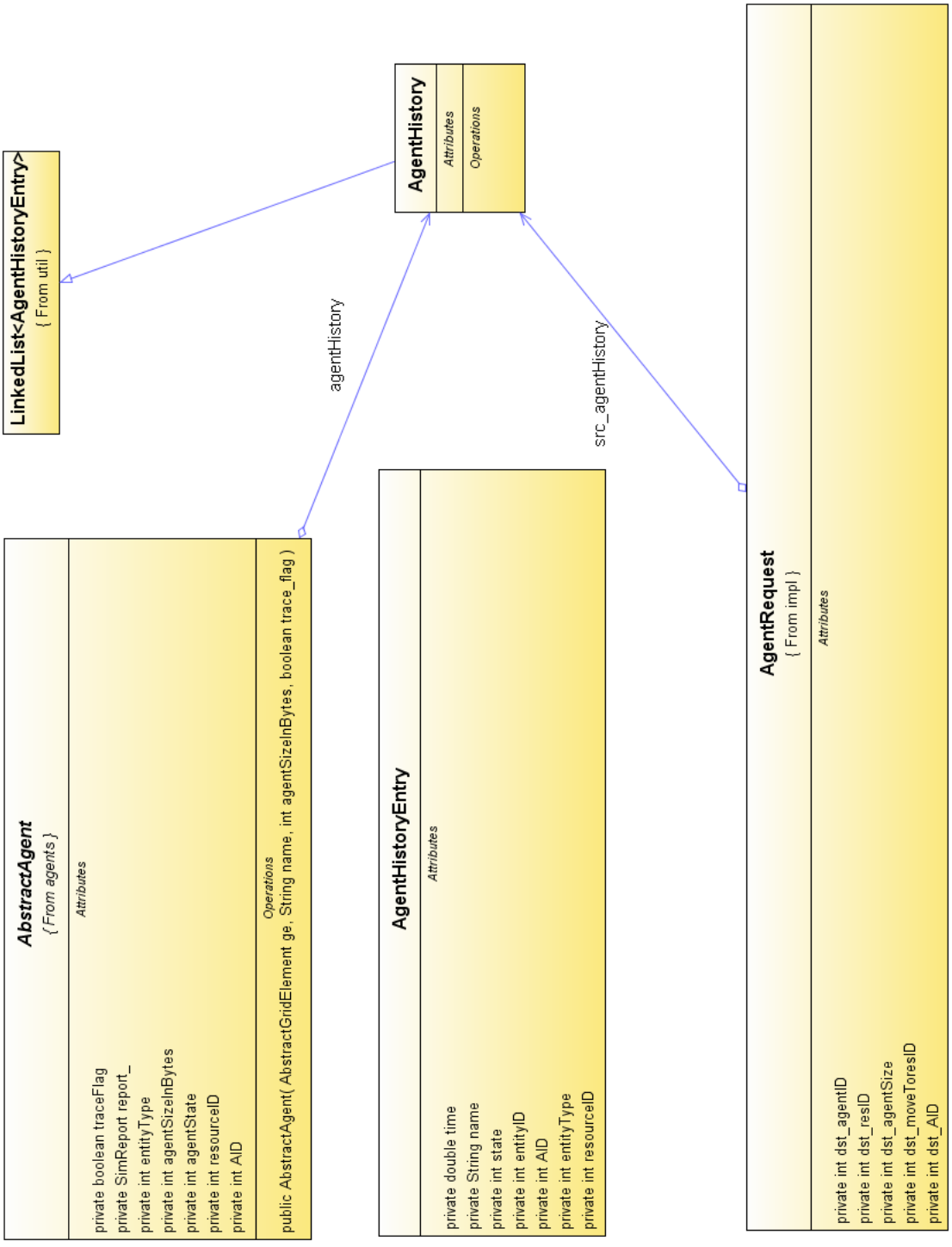

Figura 9.1: AgentHistory

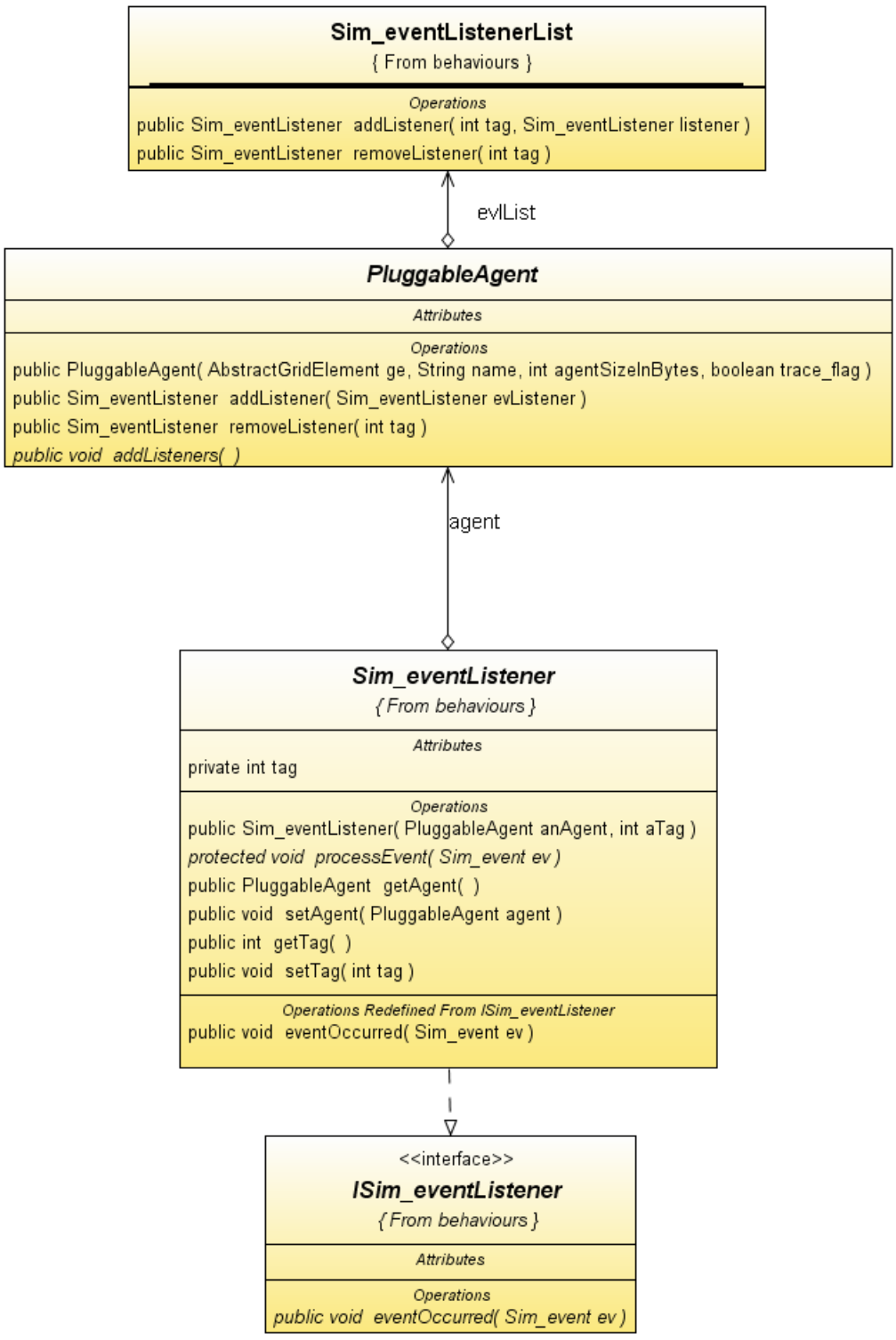

Figura 9.2: PluggableAgent

Figura 9.3: Gerarchia delle classi relative agli agenti

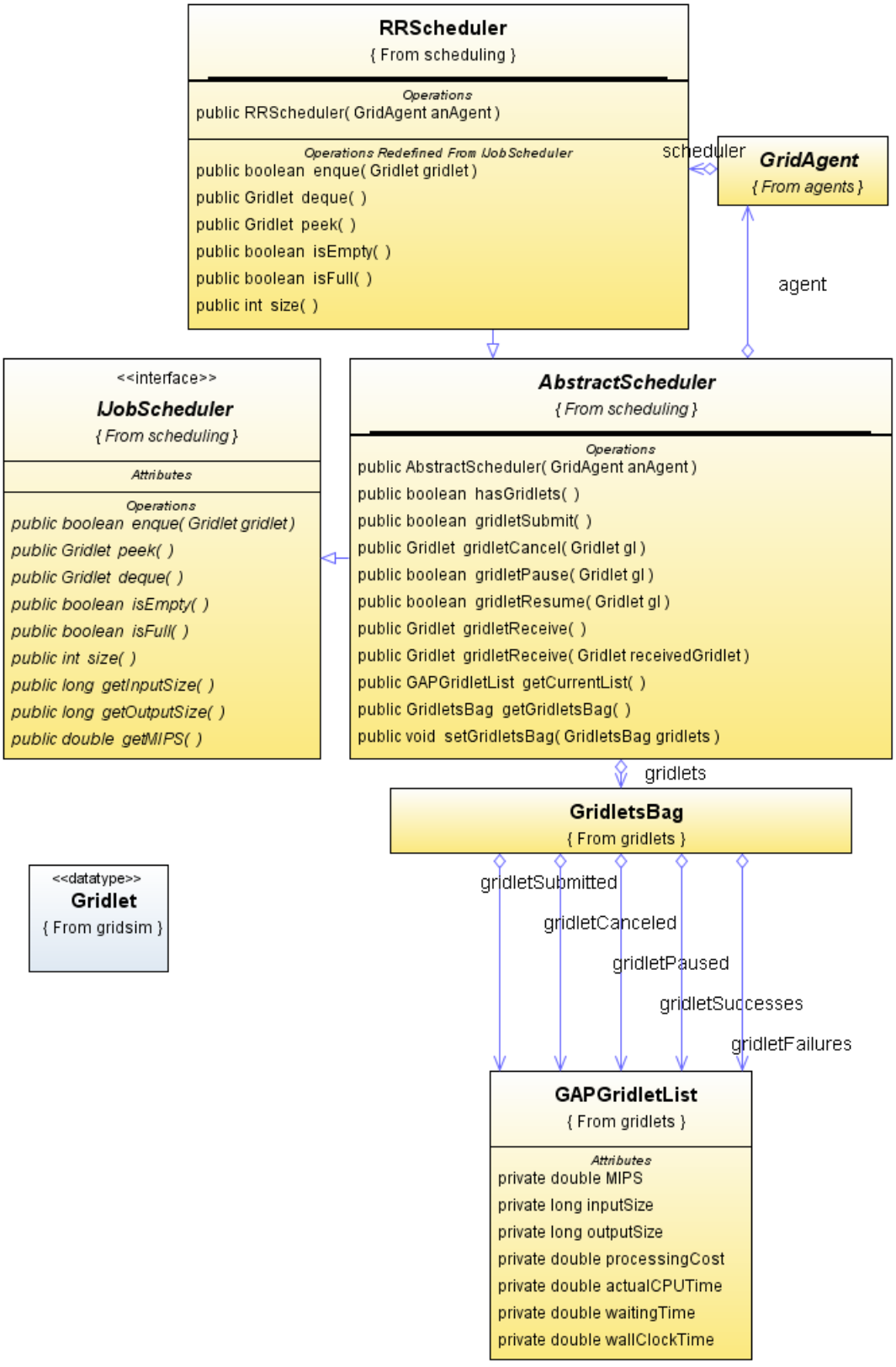

Figura 9.4: Sottomissione e gestione di jobs tramite agenti

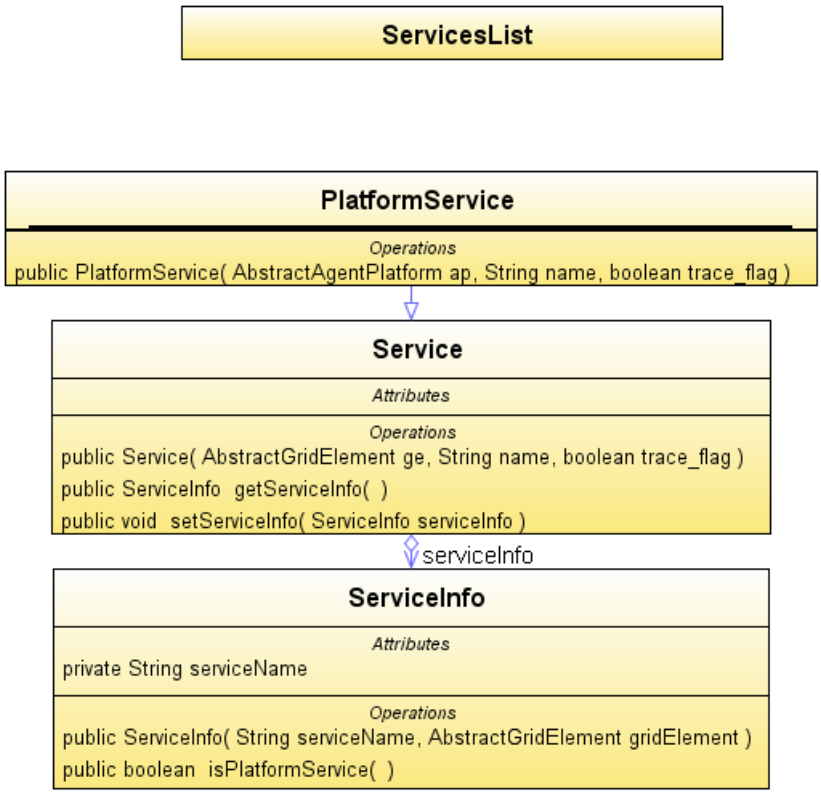

Figura 9.5: Servizi per l'Agent Platform

- interrogazione dei servizi forniti dalla piattaforma ad agenti
- accesso a informazioni di monitoraggio delle risorse su cui gli agenti girano o con cui gli agenti interagiscono quali gli Storage Element e le risorse di rete
- gestione del ciclo di vita degli agenti per la loro sottomissione, sospensione, ripristino, migrazione, terminazione e la loro registrazione nel Directory Facilitator
- gestione della sottomissione, monitoraggio e checkpointing di jobs sulle risorse dell'ambiente di Grid Computing per il tramite della piattaforma ad agenti

Il package *gap.distributions* contiene classi derivate dal package *simjava.distributions* della libreria SimJava2 (vedasi sezione 8.0.5 a pagina 79) con distribuzioni utili alla generazione di numeri casuali.

Il package *gap.factories* contiene classi per la generazione di nuove risorse Grid e relativi collegamenti di rete.

Il package *gap.grid* contiene le classi responsabili della rappresentazione in modo unificato dei Computing Element e Storage Element come istanze della classe *GridElement*; contiene inoltre un'astrazione di Virtual Organization (class *VirtualOrganization*), responsabile di istanziare un ambiente di Grid Computing con singola Virtual Organization e in cui sono attivi agenti e utenti, e la rappresentazione della topologia di rete (classe *NetworkTopology*) (vedasi figura 9.6 a pagina 89).

Il package *gap.messages* contiene la gerarchia di classi responsabile della rappresentazione dei messaggi essenziali tra agenti in un ambiente di Grid Computing.

Le due classi base di questo package sono:

1. classe *Request* (vedasi figura 9.7 a pagina 89) che mantiene riferimenti univoci alla richiesta e alla relativa conversazione, il riferimento al mittente del messaggio e della risorsa Grid ospitante il mittente.

2. classe *Reply* (vedasi figura 9.8 a pagina 90) che mantiene il riferimento alla richiesta che lo ha attivato ed i riferimenti univoci alle relative risposte.

Il package *gap.simulation* contiene la classe astratta *AbstractSimulation* che ha la responsabilità di definire livello di confidenza, accuratezza ed il ricorso a tecniche di output analysis, ereditate dalla libreria SimJava2.

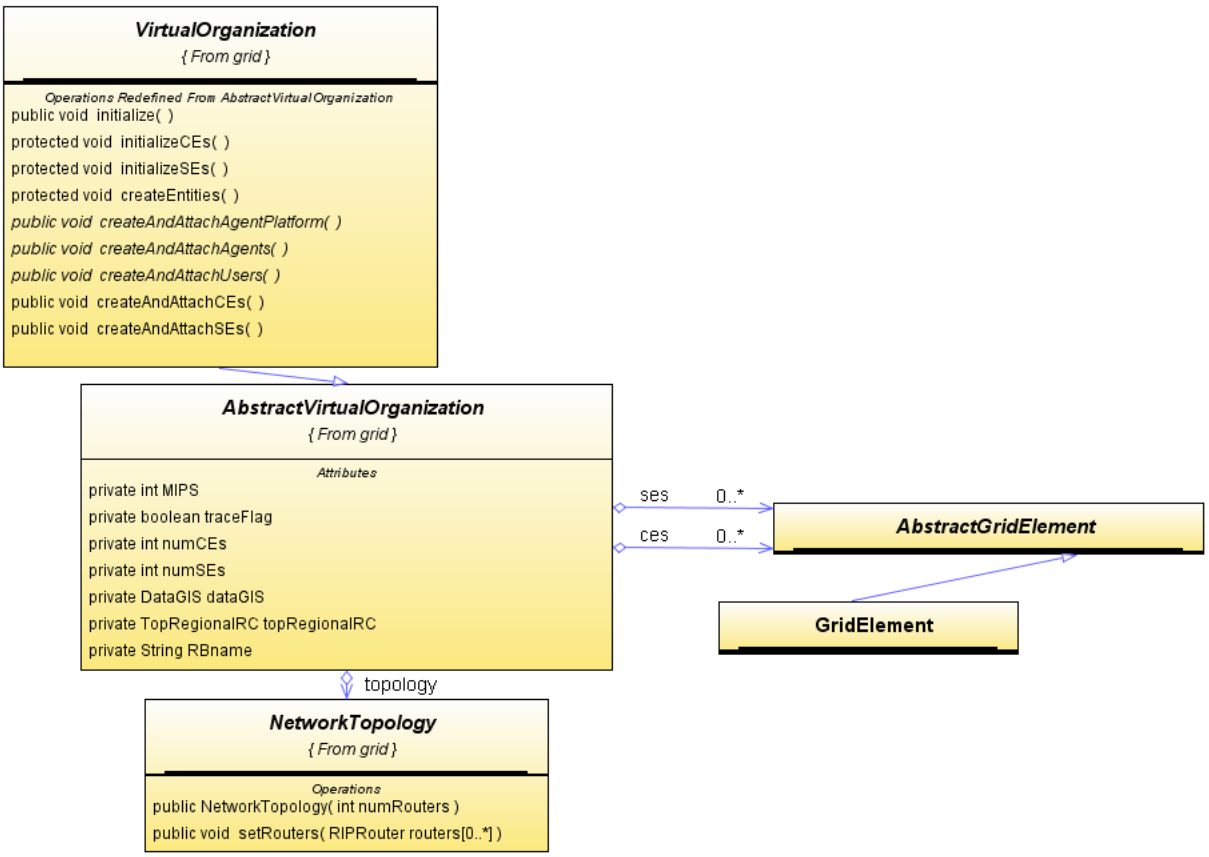

Figura 9.6: Classi relative all'ambiente di Grid Computing

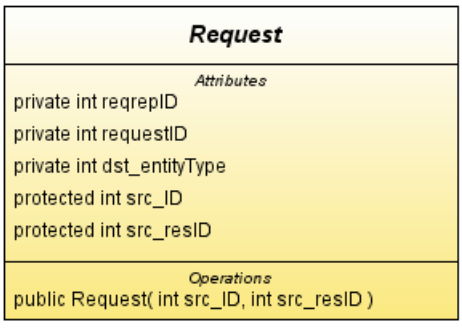

Figura 9.7: Classe Request

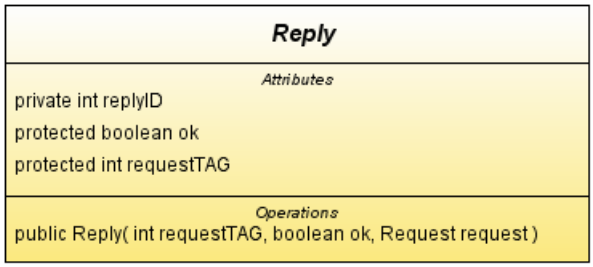

Figura 9.8: Classe Reply

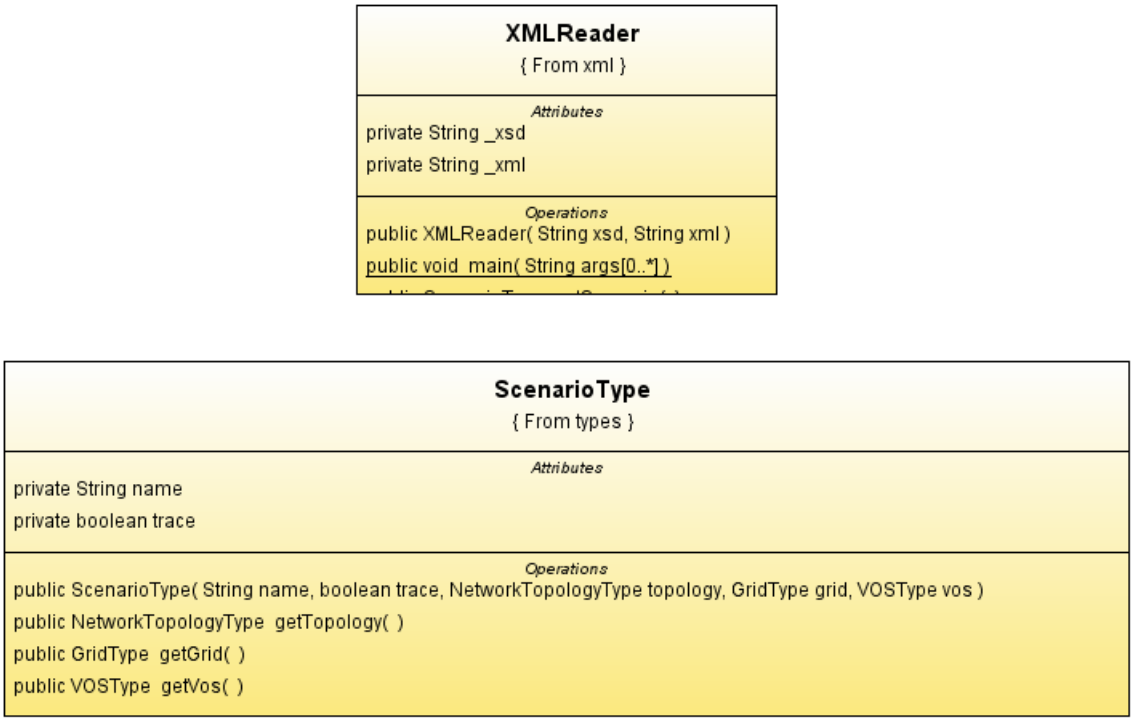

Figura 9.9: Classe XMLReader

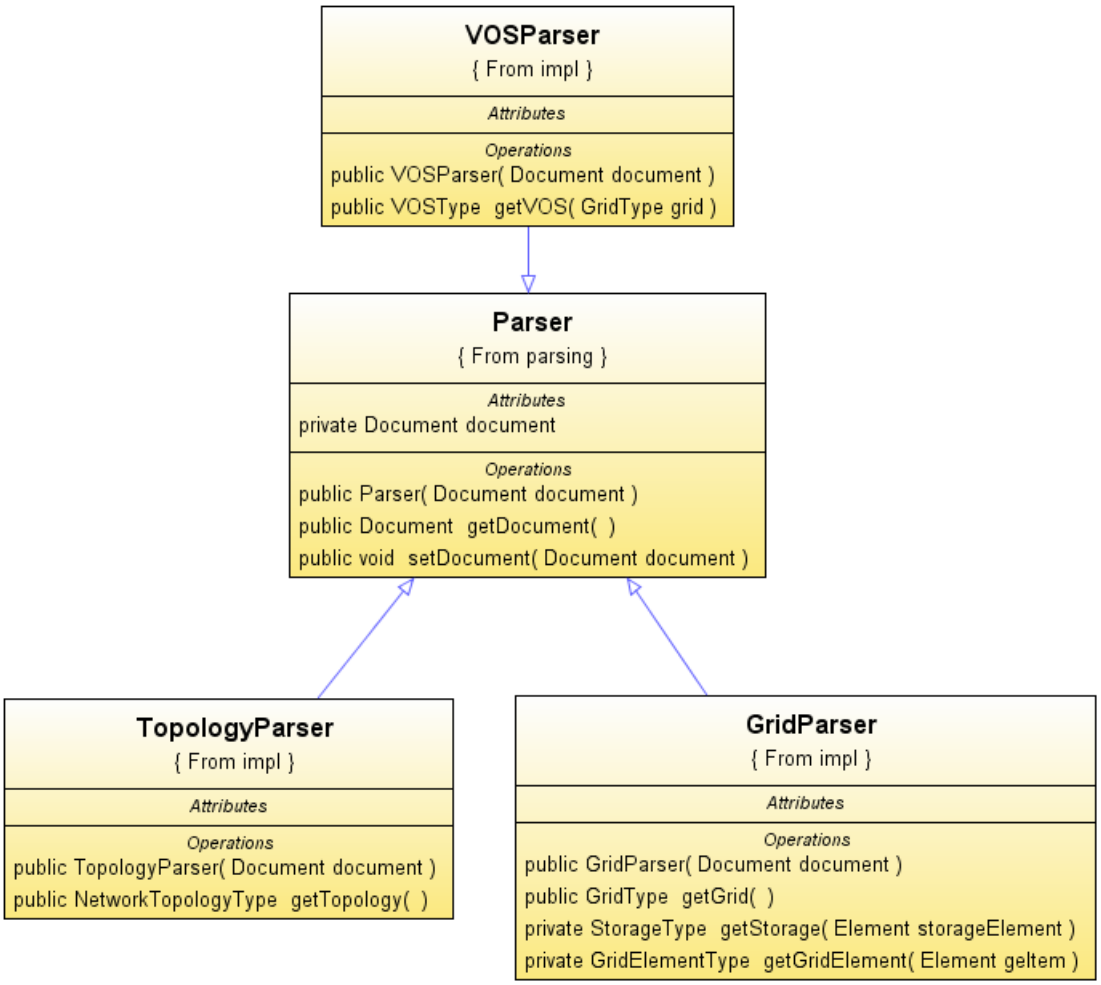

Figura 9.10: Classi per il parsing XML

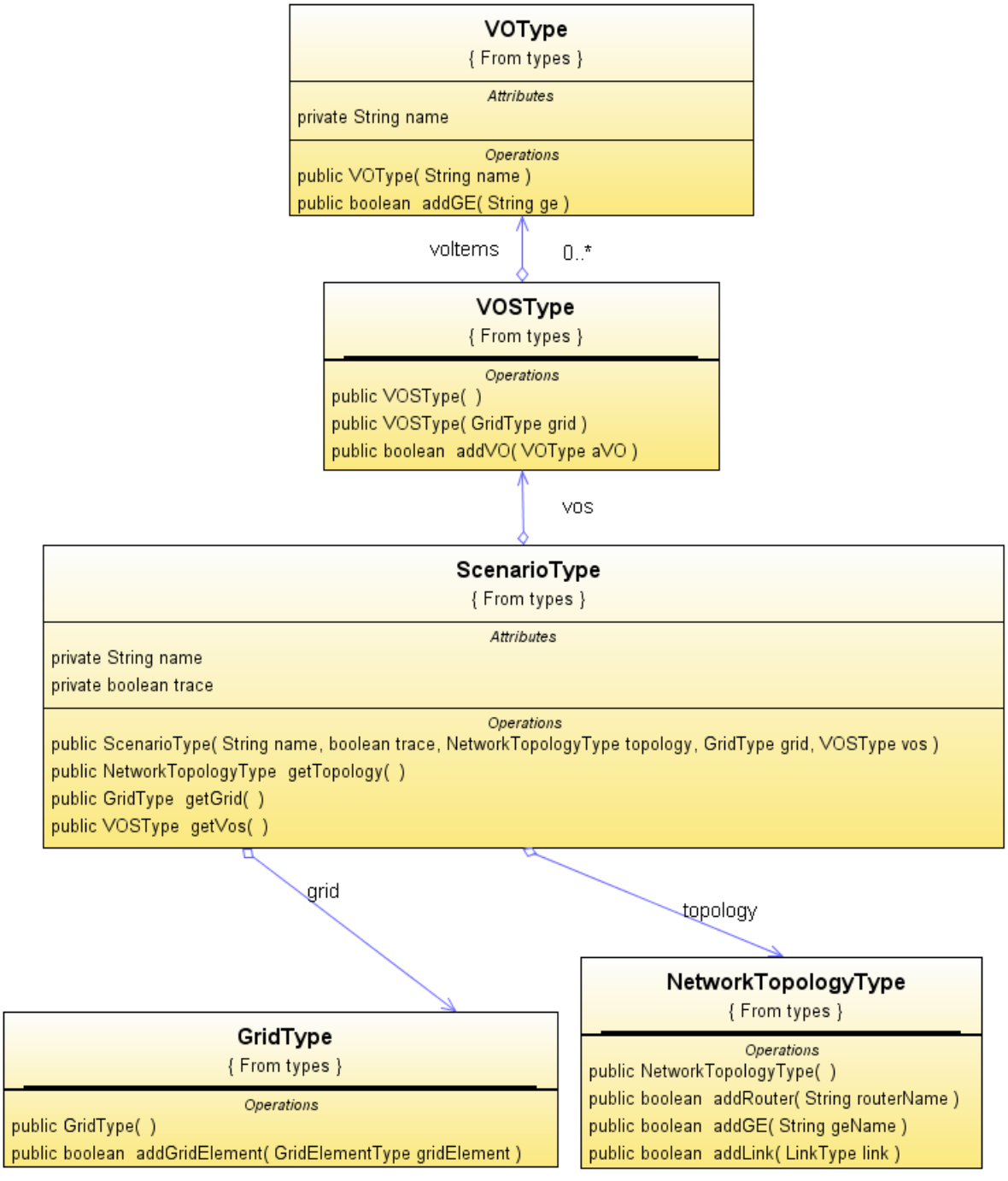

Figura 9.11: Ambiente di Grid Computing

Il package *gap.users* contiene la classe astratta *AbstractUser* che ha tra le sue responsabilità:

1. la sottomissione, sospensione, ripristino, migrazione e terminazione di agenti da parte degli utenti
2. la delega della sottomissione di jobs agli agenti da parte degli utenti
3. l'interrogazione dello stato degli agenti e dei servizi della piattaforma ad agenti
4. fornire dei metodi astratti, corrispondenti a fasi distinte della simulazione, in cui inserire le attività dell'utente

Il package *gap.util* contiene la classe *EntitiesCounter* usata per identificare univocamente entità e relativi messaggi.

Per semplificare la modellazione degli ambienti di Grid Computing in GAP è stato sviluppato il package *gap.xml* per la lettura delle VO, dei Computing Element e degli Storage Element e relativi collegamenti di rete da file XML (eXtensible Markup Language) [143, 144] permettendo di rappresentare in XML tutte le entità previste dalla sottostante libreria GridSim ovvero:

1. architettura delle risorse
2. distribuzione geografica e carico delle risorse
3. supporto per più Virtual Organization
4. Computing Element e Storage Element
5. supporto per memorie di massa secondarie (dischi) e terziarie (nastri)
6. topologia di rete
7. supporto per più tipologie di router

Il package *gap.xml* si compone pertanto di una trentina di classi che forniscono le funzionalità di parsing necessarie a ricavare da un file XML, predisposto una rappresentazione di un ambiente di Grid Computing. La classe principale di tale package è

gap.xml.XMLReader (vedasi figura 9.9 a pagina 90) che viene istanziata con i percorsi del file XML di input e del relativo schema XSD (XML Schema Definition) [145, 146].

Il parsing (vedasi figura 9.10 a pagina 91) restituisce un'istanza della classe *ScenarioType* che mantiene una rappresentazione dell'ambiente di Grid Computing relativamente alla topologia di rete, alle risorse di Grid Computing coinvolte e alla loro eventuale appartenenza a Virtual Organization differenti (vedasi figura 9.11 a pagina 92).

Un estratto dello schema XSD è riportato in 9.12 a pagina 95, listato relativo alla corrispondente classe *ScenarioType*.

Entrando più nel dettaglio lo schema XSD fornisce una rappresentazione delle caratteristiche delle risorse di calcolo presenti nell'ambiente di Grid Computing simulato con riferimento all'architettura hardware, al sistema operativo, alla policy di scheduling, al fuso orario ed eventuale prezzo delle risorse di calcolo per MIPS (vedasi figura 9.13 a pagina 96).

Inoltre tramite XSD è definita una rappresentazione del carico di sottofondo delle risorse presenti nell'ambiente di Grid Computing (vedasi figura 9.14 a pagina 96).

```
<xsd:complexType name="ScenarioType">
    <xsd:sequence>
        <xsd:element name="topology" type="NetworkTopologyType"/>
        <xsd:element name="calendars" minOccurs="0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="calendarItem" maxOccurs="unbounded"
                                 type="ResourceCalendarType"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="characteristics" minOccurs="0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="characteristicsItem"
                                 maxOccurs="unbounded"
                                 type="ResourceCharacteristicsType"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="grid" minOccurs="0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="gridElement" maxOccurs="unbounded"
                                 type="GridElementType" minOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="vos" minOccurs="0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="voItem" type="VOType" minOccurs="1"
                                 maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="trace" type="xsd:boolean" use="optional"/>
</xsd:complexType>
```

Figura 9.12: Tipo XSD ScenarioType

```
<xsd:complexType name="ResourceCharacteristicsType">
    <xsd:sequence>
        <xsd:element name="Architecture" type="xsd:string" minOccurs="0"
                     default="System Architecture"></xsd:element>
        <xsd:element name="OS" type="xsd:string" minOccurs="0"
                     default="Operating System"/>
        <xsd:element name="machinesList" type="MachineListType"/>
        <xsd:element name="Policy" type="xsd:int"
                     default="0" minOccurs="0"/>
        <xsd:element name="Timezone" type="xsd:double"
                     default="12.0" minOccurs="0"/>
        <xsd:element name="Cost" type="xsd:double" minOccurs="0"
                     default="1.0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>
```

Figura 9.13: Tipo XSD ResourceCharacteristicsType

```
<xsd:complexType name="ResourceCalendarType">
    <xsd:sequence>
        <xsd:element name="TimeZone" type="xsd:double"/>
        <xsd:element name="PeakLoad" type="xsd:double"/>
        <xsd:element name="OffPeakLoad" type="xsd:double"/>
        <xsd:element name="RelativeHolidayLoad" type="xsd:double"/>
        <xsd:element name="weekendList" type="WeekendListType"/>
        <xsd:element name="holidayList" type="HolidayListType"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>
```

Figura 9.14: Tipo XSD ResourceCalendarType

Capitolo 10

Simulazione dell'architettura proposta

La simulazione dell'architettura proposta è stata realizzata con una applicazione basata sulla libreria GAP (vedasi capitolo 9 a pagina 80).

L'applicazione si compone di una serie di package principali:

1. *mcd.agents*
2. *mcd.multimedia*
3. *mcd.stats*
4. *mcd.users*

Il package *mcd.agents* contiene la classe principale *TranscodingAgent*, corrispondente all'agente software descritto nella sezione 5.4 a pagina 45, e le classi relative ai servizi con cui si interfaccia tale agente software. La classe *TranscodingAgent* è derivata dalla classe *PluggableAgent* (vedasi capitolo 9 a pagina 82) e ad essa sono associati i seguenti behaviours:

1. MonitoringBehaviour
2. CollaborationBehaviour
3. FetchingBehaviour

4. TranscodingBehaviour
5. CachingBehaviour
6. StreamingBehaviour
7. QoSBehaviour

Il package *mcd.agents.services* è utilizzato per modellare i servizi della soluzione proposta derivandone la rappresentazione dalla classe astratta *PlatformService* utilizzata per i servizi TraS (*Transcoder Selector*), CoT (*Content Tracer*), THub (*Transcoder Hub*), CET (*Computing Element Tracer*), SET (*Storage Element Tracer*) e TraM (*Transcoder Monitor*) e dalla classe astratta *Service* per i servizi NM (*Network Monitor*), CEM (*Computing Element Monitor*) e GM (*Grid Mediator*).

All'interno del package *mcd.agents* vi è il package *mcd.agents.fuzzy* in cui è definito il controllo fuzzy utilizzato per l'adattamento della Quality of Service e basato sulla libreria *Fuzzy Engine* [1] [80].

Infine il package *mcd.multimedia* contiene classi per:

1. rappresentare i contenuti multimediali come sequenze di chunk ed i relativi parametri di interesse per la simulazione delle attività di online transcoding e di streaming
2. leggere tali parametri da file CSV

Il package *mcd.users* contiene classi per:

1. la sottomissione diretta di jobs sull'ambiente di Grid Computing per l'online transcoding e streaming di contenuti multimediali
2. la richiesta di contenuti multimediali all'architettura proposta tramite sottomissione di richieste al servizio TraS (vedasi sezione 5.2.1 a pagina 36).

[1] http://www.intelligent-systems.info/FuzzyEngine.htm

Parte IV

Risultati sperimentali

Capitolo 11

Raccolta di misure sul transcoding

Al fine di ottenere stime sull'entità delle risorse coinvolte per l'attività di online transcoding è stata effettuata una serie di esperimenti di transcoding da un formato ad un altro considerando filmati con risoluzioni differenti. I tempi di transcoding osservati e le correlazioni individuate sono stati utilizzati come input per le simulazioni (vedasi capitolo 12 a pagina 103).

StartFrame	EndFrame	FPS	W	H	SIZE bytes	FD msec	TT msec
1001	1050	25	320	240	114500	2000	113
1051	1100	25	304	228	105654	2000	138
1101	1150	20	292	216	56556	2000	103
1151	1200	15	292	216	37432	2000	105
1201	1250	15	304	228	50940	2000	192
1251	1300	20	320	240	61078	2000	162

Tabella 11.1: **Transcodifica in formato mpeg4**

Le misure riportate nelle tabelle 11.1,11.2, 11.3 e 11.4 si riferiscono ad esperimenti su un singolo filmato, tra i vari considerati negli esperimenti effettuati, con frame rate di 25 frame al secondo, risoluzione iniziale di 320 per 240 pixels ed una dimensione del file originario di circa 21 MB. Per tale filmato è stato effettuato il transcoding di 300 frame per una durata complessiva di 12 secondi di riproduzione e le misure di transcoding sono state ricavate variando i seguenti parametri di transcoding:

1. formato di output

2. risoluzione

3. frame rate

Nelle tabelle si utilizzano una serie di abbreviazioni:

1. *StartFrame* e *EndFrame* indicano il frame iniziale e il frame finale di un chunk

2. *FPS* sta ad indicare il numero di frame al secondo

3. *W* e *H* indicano larghezza ed altezza dei frame

4. *SIZE bytes* è la dimensione in bytes del chunk dopo la transcodifica

5. *FD msec* è la durata del chunk espressa in millisecondi

6. *TT msec* è la durata della transcodifica in millisecondi

StartFrame	EndFrame	FPS	W	H	SIZE bytes	FD msec	TT msec
1001	1050	25	320	240	190854	2000	113
1051	1100	25	304	228	192678	2000	126
1101	1150	20	292	216	109030	2000	090
1151	1200	15	292	216	73238	2000	091
1201	1250	15	304	228	168722	2000	146
1251	1300	20	320	240	288918	2000	142

Tabella 11.2: **Transcodifica in formato mjpeg**

Le misure dei tempi di transcoding sono state effettuate su chunk di 50 frame con una durata di riproduzione pari a 2 secondi. Per ciascun chunk è stata misurata la dimensione in byte del chunk ottenuto dalla transcodifica e il tempo di transcodifica. Per la transcodifica sono stati utilizzati tool di transcodifica [147, 50, 148] *mencoder* [1] e *transcode* [2] opportunamente compilati per includere staticamente librerie e codecs necessarie per gli esperimenti di transcoding, in particolare *ffmpeg* [3] e *libavcodec* [4]. Al fine di ridurre l'impatto del caricamento di tali tool in memoria primaria sono state effettuate per ciascun chunk più misurazioni consecutive escludendo la prima misurazione dal calcolo

[1] http://www.mplayerhq.hu/DOCS/HTML/en/mencoder.html
[2] http://www.transcoding.org
[3] http://ffmpeg.mplayerhq.hu/
[4] http://www.mplayerhq.hu/DOCS/HTML/it/video-codecs.html

della media del tempo di transcoding ed utilizzandola come stima di latenza nell'avvio delle operazioni di transcodifica.

StartFrame	EndFrame	FPS	W	H	SIZE bytes	FD msec	TT msec
1001	1050	25	320	240	201262	2000	272
1051	1100	25	304	228	177086	2000	171
1101	1150	20	292	216	111382	2000	129
1151	1200	15	292	216	75428	2000	122
1201	1250	15	304	228	48930	2000	181
1251	1300	20	320	240	63788	2000	168

Tabella 11.3: **Transcodifica in formato h263p**

Tali risultati sono stati raccolti da esperimenti effettuati su un PC con processore AMD Athlon 64 3500+, frequenza di CPU impostata a 1 Ghz durante gli esperimenti, sistema operativo Linux Ubuntu Dapper e kernel 2.6.16.

StartFrame	EndFrame	FPS	W	H	SIZE bytes	FD msec	TT msec
1001	1050	25	320	240	201496	2000	313
1051	1100	25	304	228	176870	2000	162
1101	1150	20	292	216	111610	2000	119
1151	1200	15	292	216	75188	2000	116
1201	1250	15	304	228	48344	2000	169
1251	1300	20	320	240	63382	2000	235

Tabella 11.4: **Transcodifica in formato rv10**

Capitolo 12

Scenari

Con l'applicazione sviluppata (vedasi capitolo 10 a pagina 97), facendo ricorso alla libreria GAP (vedasi capitolo 9 a pagina 80), sono state effettuate diverse simulazioni considerando principalmente tre scenari:

1. sottomissione diretta dei jobs di online transcoding e streaming da parte degli utenti
2. la soluzione proposta in assenza dell'euristica di prossimità
3. la soluzione proposta inclusiva dell'euristica di prossimità

Tali tre scenari sono stati simulati separatamente e le misure raccolte dalla simulazione sono state comparate tra loro al fine di ottenere conferme qualitative e quantitative sulla bontà della soluzione proposta.

Gli scenari simulati, al fine di risultare comparabili, differiscono solamente nelle modalità di sottomissione ed espletamento delle richieste degli utenti, restando accomunati rispetto alle altre variabili simulate ovvero:

1. infrastruttura di rete
2. infrastruttura di Grid Computing
3. dislocazione dei servizi su tali risorse
4. numero e posizione degli utenti
5. modalità di generazione delle richieste utente

6. QoS minima accettabile da parte degli utenti

L'infrastruttura di rete e l'infrastruttura di Grid Computing sono state fornite in input all'applicazione di simulazione in formato XML tramite il package Java *gap.xml* della libreria GAP descritto nel capitolo 9 a pagina 93.

L'infrastruttura simulata (vedasi figura 12.1 a pagina 105) è costituita da :

1. 22 routers di cui:
   - 18 routers cui sono connessi, tramite collegamenti di rete da 155 Mbps, i Computing Element e gli Storage Element
   - 4 routers principali interconnessi tra loro con collegamenti da 1 Gbps
2. 9 Computing Element
3. 9 Storage Element

La medesima infrastruttura è stata utilizzata nella simulazione di scenari differenti; l'utente U_i è collegato al router Ri per $i = 0, 1, 2, 3$ ed indicante uno dei quattro routers principali.

Durante la simulazione sono state osservati vari indicatori tra cui:

1. attività di I/O degli Storage Element
2. QoS loss calcolata per tutte le richieste nel loro complesso
3. numero di richieste contemporanee
4. numero di richieste servite
5. informazioni di dettaglio sul singolo stream, tra cui
   - tempo di invio della richiesta da parte dell'utente
   - tempo di risposta per l'inizializzazione dei jobs necessari ad espletare la richiesta
   - tempo per il completamento del transcoding del primo chunk
   - tempi di ricezione del primo e dell'ultimo frame
   - delay percepito dall'utente sul singolo chunk
   - QoS loss per ogni chunk

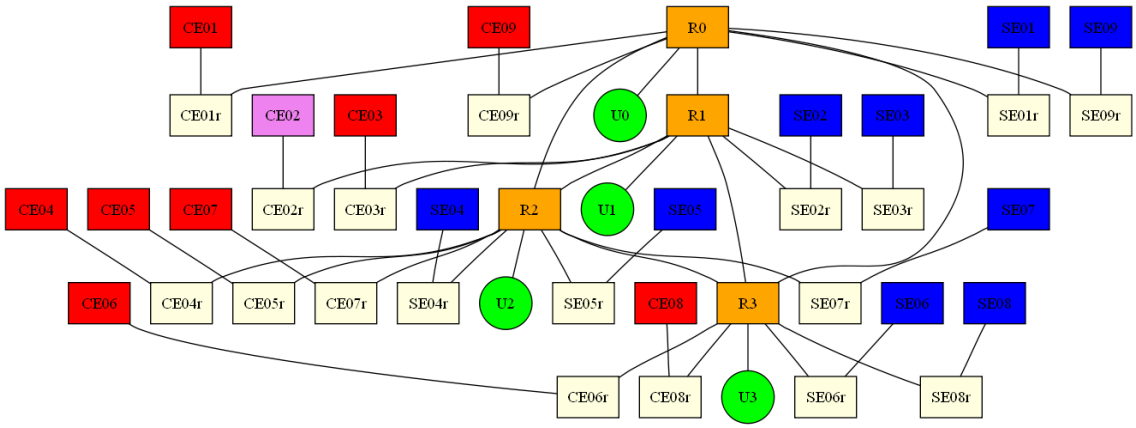

Figura 12.1: Grafo delle risorse

12.1 Sottomissione diretta dei jobs di online transcoding e streaming da parte degli utenti

In questo scenario l'utente invia all'ambiente di Grid Computing un job che effettua le seguenti operazioni:

1. reperimento del contenuto multimediale da uno Storage Element
2. online transcoding del contenuto multimediale per chunk
3. streaming dei chunk al client richiedente
4. adattamento della Quality of Service sulla base dei requisiti utente

Il Computing Element e lo Storage Element che intervengono nell'espletamento della richiesta sono selezionati casualmente tra quelli disponibili nell'infrastruttura di Grid Computing simulata.

Il grafico 12.2 a pagina 107 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il rapporto tra il tempo di ricezione dell'intero stream e la durata dello stream (*normalized streaming time*). Al fine di ottenere una ricezione fluida degli stream, condizione necessaria è che il valore dell'ordinata si mantenga minore o al più uguale ad una soglia accettabile pari ad 1, mentre dalla lettura del grafico si evince che tale valore supera ampiamente tale soglia

crescendo fino a picchi di 10 nel corso della simulazione. Tali valori elevati sono dovuti ai seguenti fattori concomitanti:

1. le attività connesse alla sottomissione, scheduling ed esecuzione del job sulla Grid introducono un ritardo che tende a crescere al sovraccaricarsi delle risorse di calcolo utilizzate

2. le latenze di rete, tra utente e Computing Element e tra Computing Element e Storage Element, non sono ottimizzate per via di una selezione di tali risorse svincolata dagli scopi che il job di online transcoding e streaming eseguito su di esse si prefiggerebbe, ovvero soddisfare la richiesta utente non solo in termini del suo espletamento ma anche di tempi di streaming e qualità dello stream

3. a tali latenze si aggiunge l'overhead introdotto dall'attività di online transcoding che, se effettuata su risorse inappropriate, incide negativamente sulla qualità della ricezione aggiungendo un ulteriore ritardo ad ogni chunk elaborato sia per quanto concerne l'attività CPU intensive di transcodifica che per quanto concerne l'attività di reperimento dei chunk dallo Storage Element

Il grafico 12.3 a pagina 108 presenta le medie dei valori riportati nel grafico 12.2; in ascissa si ha il router cui uno o più utenti sono collegati e in ordinata il valore medio durante l'intera simulazione del rapporto tra il tempo di ricezione dell'intero stream e la durata dello stream (*normalized streaming time*).

Il grafico 12.4 a pagina 109 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il tempo di risposta da parte dei job espresso in secondi (*response time in seconds*). Il tempo di risposta è calcolato rispetto all'intervallo tra il momento in cui l'utente invia il job e il momento in cui il job inizia la sua esecuzione sulla Grid. Da una lettura del grafico si evince che il tempo di risposta tende a crescere da valori di pochi secondi a picchi di 20 secondi nel corso della specifica simulazione.

Il grafico 12.5 a pagina 110 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il tempo di transcoding del primo chunk espresso in secondi (*first chunk time in seconds*). Il tempo di transcoding del primo chunk è comprensivo del tempo di risposta ed è calcolato rispetto all'intervallo tra

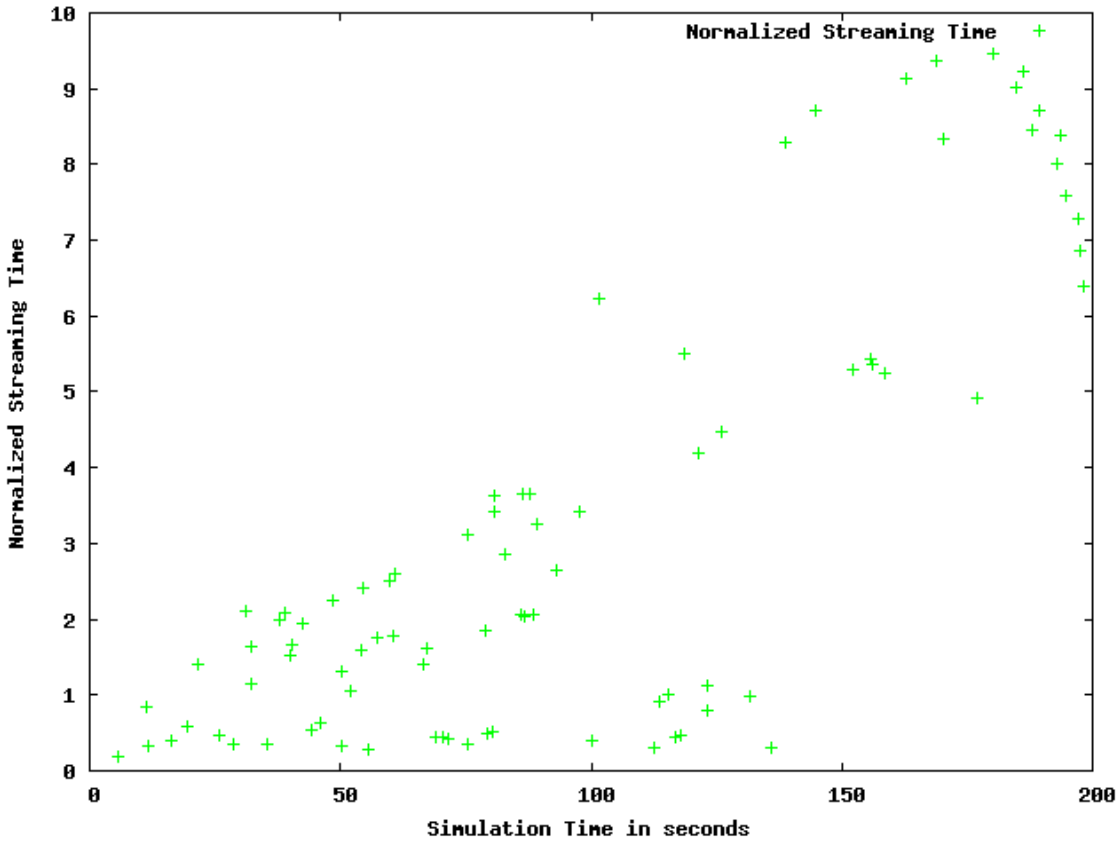

Figura 12.2: Streaming: singoli stream

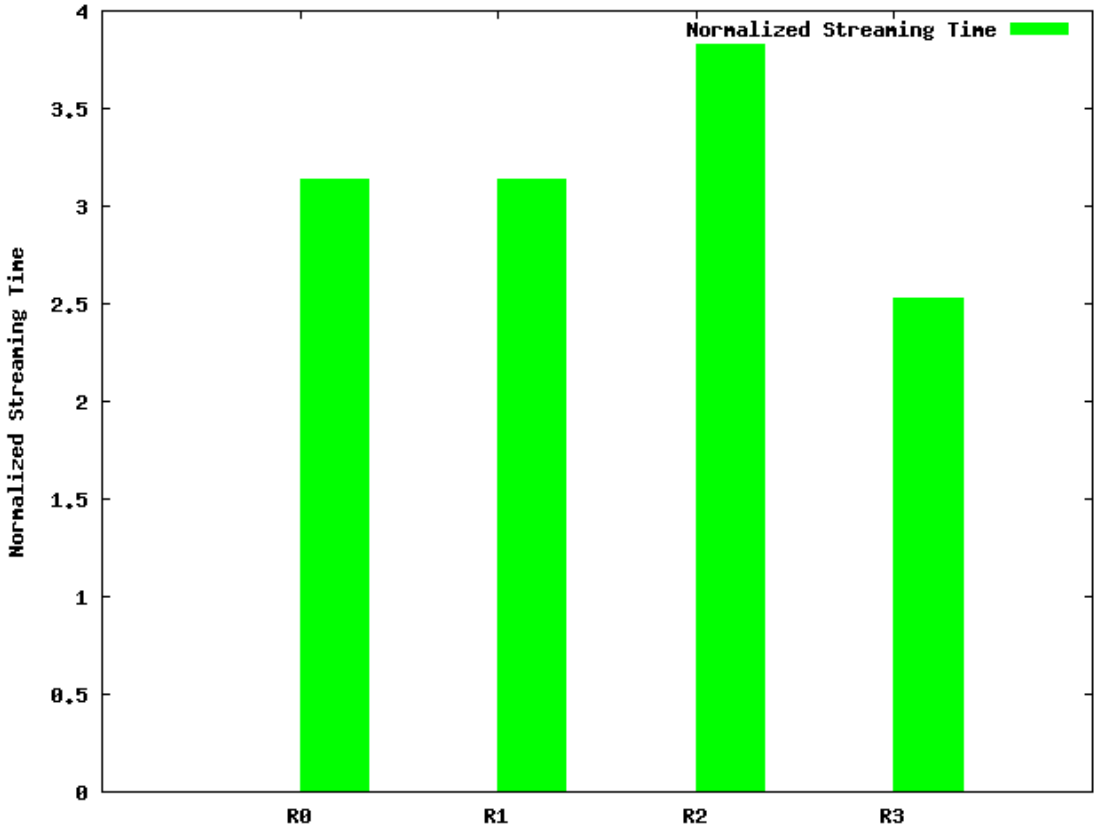

Figura 12.3: Streaming: valori medi

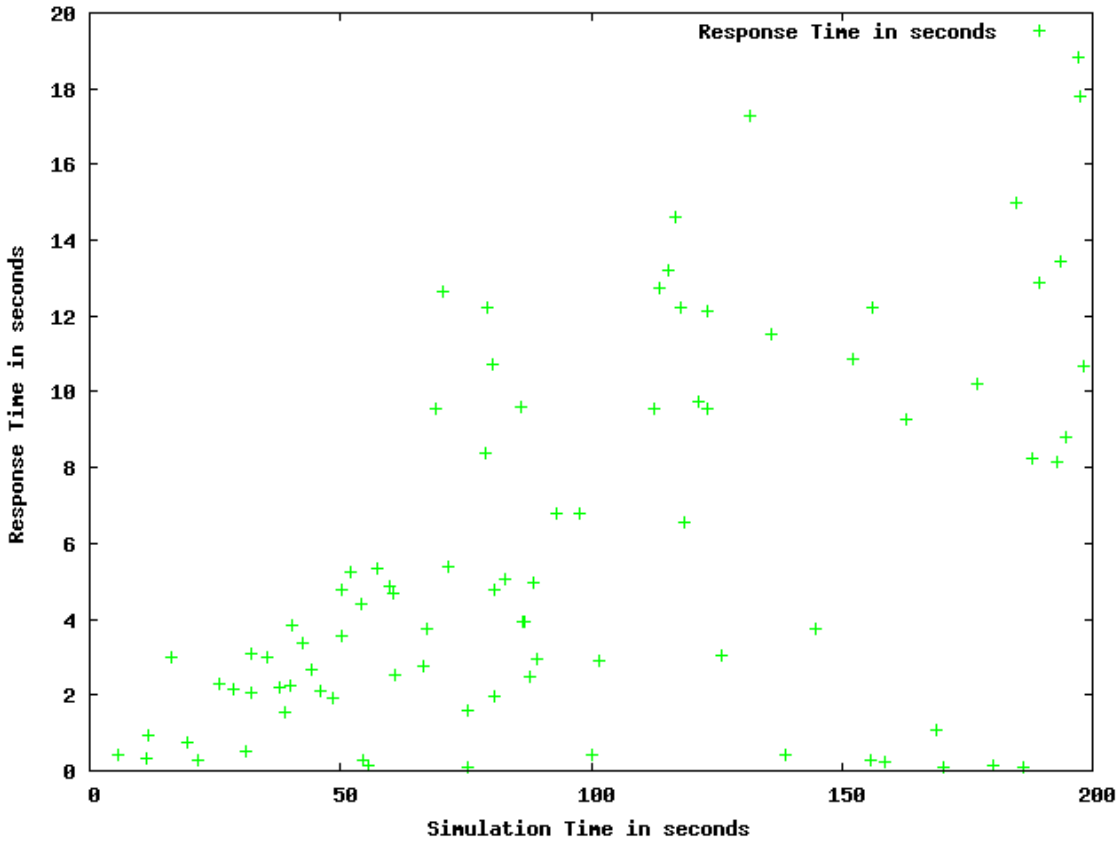

Figura 12.4: Tempo di risposta: singoli stream

il momento in cui l'utente invia il job e il momento in cui il job completa il transcoding del primo chunk del contenuto multimediale richiesto dall'utente. Da una lettura del grafico si evince che il tempo di transcoding del primo chunk tende a crescere da valori di pochi secondi a picchi di 20 secondi nel corso della specifica simulazione. Tale crescita è spiegabile con i seguenti fattori in particolare:

1. crescita del carico degli host in cui i job di online transcoding vengono eseguiti
2. latenza di rete non ottimizzata tra host e Storage Element da cui il contenuto multimediale viene reperito in chunk

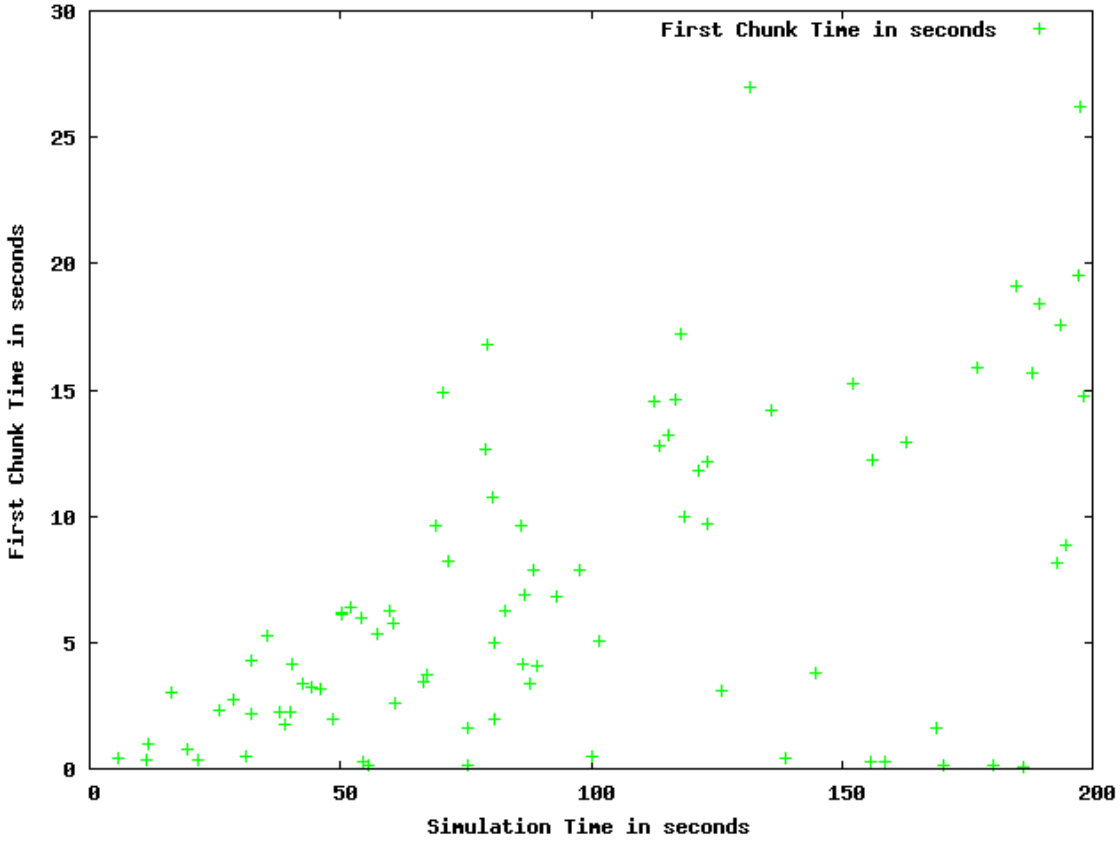

Figura 12.5: Primo chunk: singoli stream

Il grafico 12.6 a pagina 111 presenta le medie dei valori riportati nei grafici 12.4 e 12.5; in ascissa si ha il router cui uno o più utenti sono collegati e in ordinata i valori

medi del tempo di risposta e del tempo di transcoding del primo chunk durante l'intera simulazione.

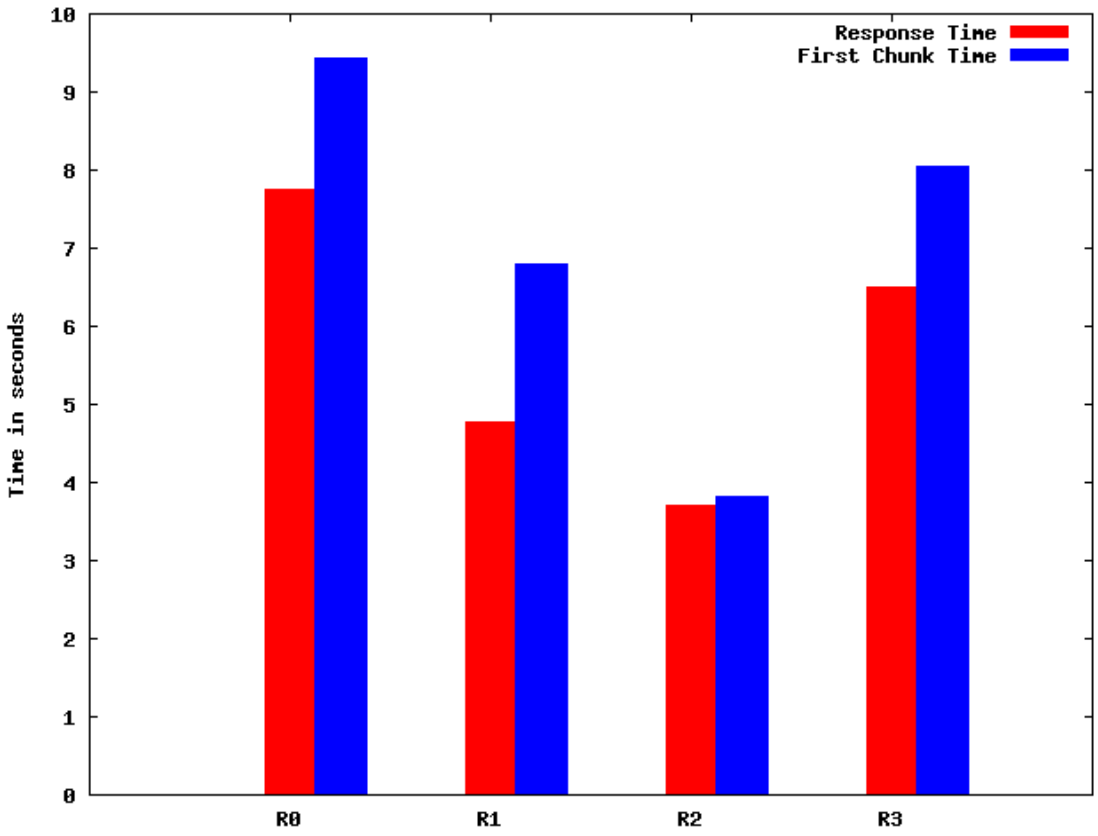

Figura 12.6: Tempi di risposta e transcoding del primo chunk: valori medi

12.2 La soluzione proposta in assenza dell'euristica di prossimità

In questo scenario l'utente invia al servizio TraS (vedasi sezione 5.2.1 a pagina 36) la richiesta di un contenuto multimediale specificando un livello minimo di QoS.

Il servizio TraS non utilizza l'euristica di prossimità nel selezionare il Computing Element e lo Storage Element che intervengono nell'espletamento della richiesta;

tali risorse sono selezionate casualmente tra quelle disponibili nell'infrastruttura di Grid Computing simulata.

Il grafico 12.7 a pagina 113 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il rapporto tra il tempo di ricezione dell'intero stream e la durata dello stream (*normalized streaming time*). Al fine di ottenere una ricezione fluida degli stream, condizione necessaria è che il valore dell'ordinata si mantenga minore o al più uguale ad una soglia accettabile pari ad 1, mentre dalla lettura del grafico si evince che tale valore, sebbene sia ridotto di oltre il 50% rispetto allo scenario di sottomissione diretta in forma di job delle richieste utente (vedasi sezione 12.1 a pagina 105), supera ampiamente tale soglia crescendo fino a picchi di 4 nel corso della simulazione. Tali valori elevati sono dovuti ai seguenti fattori concomitanti:

1. le latenze di rete, tra utente e Computing Element e tra Computing Element e Storage Element, non sono ottimizzate per via del mancato utilizzo dell'euristica di prossimità

2. l'overhead introdotto dall'attività di online transcoding se effettuata su risorse inappropriate

La riduzione di oltre il 50% dei tempi di streaming rispetto allo scenario considerato in sezione 12.1 sono spiegabili in relazione alla riduzione dell'overhead introdotto dalle attività connesse alla sottomissione, scheduling ed esecuzione del job sulla Grid che grazie all'utilizzo degli agenti vengono svolte dall'interno della Grid per la delega dell'esecuzione di job agli agenti; inoltre agenti compatibili con la richiesta, quando ancora in esecuzione sul Computing Element selezionato da TraS, sono riutilizzati per l'espletamento delle operazioni necessarie all'effettivo soddisfacimento della richiesta ed in tali casi eliminando del tutto tale overhead.

Il grafico 12.8 a pagina 114 presenta le medie dei valori riportati nel grafico 12.7; in ascissa si ha il router cui uno o più utenti sono collegati e in ordinata il valore medio durante l'intera simulazione del rapporto tra il tempo di ricezione dell'intero stream e la durata dello stream (*normalized streaming time*). In questo caso si ha una riduzione di quasi il 50% dei tempi medi di streaming rispetto allo scenario considerato in sezione 12.1. Da una lettura combinata dei grafici 12.8 e 12.7 si evince inoltre che la soluzione proposta, anche in assenza di euristica di prossimità, introduce un miglioramento che si

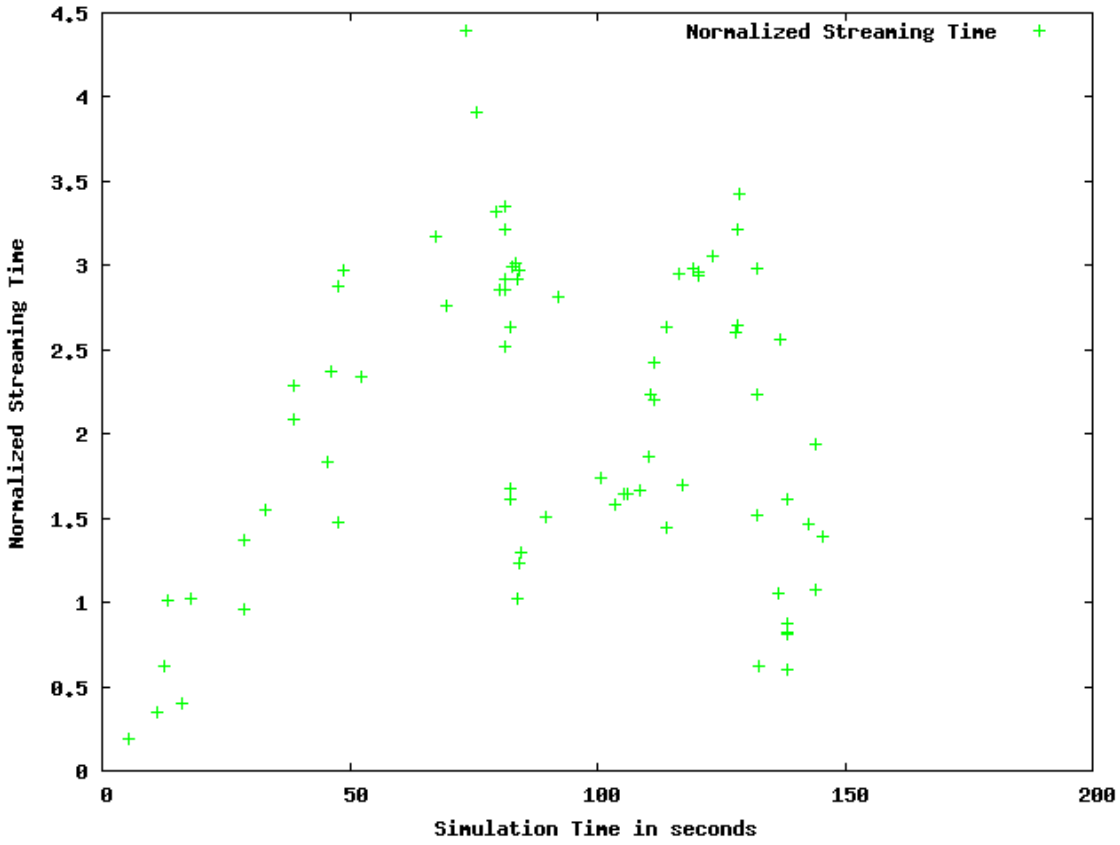

Figura 12.7: Streaming: singoli stream

rende più palpabile durante il corso della simulazione e spiegabile in virtù del riutilizzo di agenti già presenti sulla Grid tramite redirezione di nuove richieste ad agenti già attivi e le proprietà di mobilità proprie degli agenti (a tal proposito vedasi anche il grafico 12.11 a pagina 117).

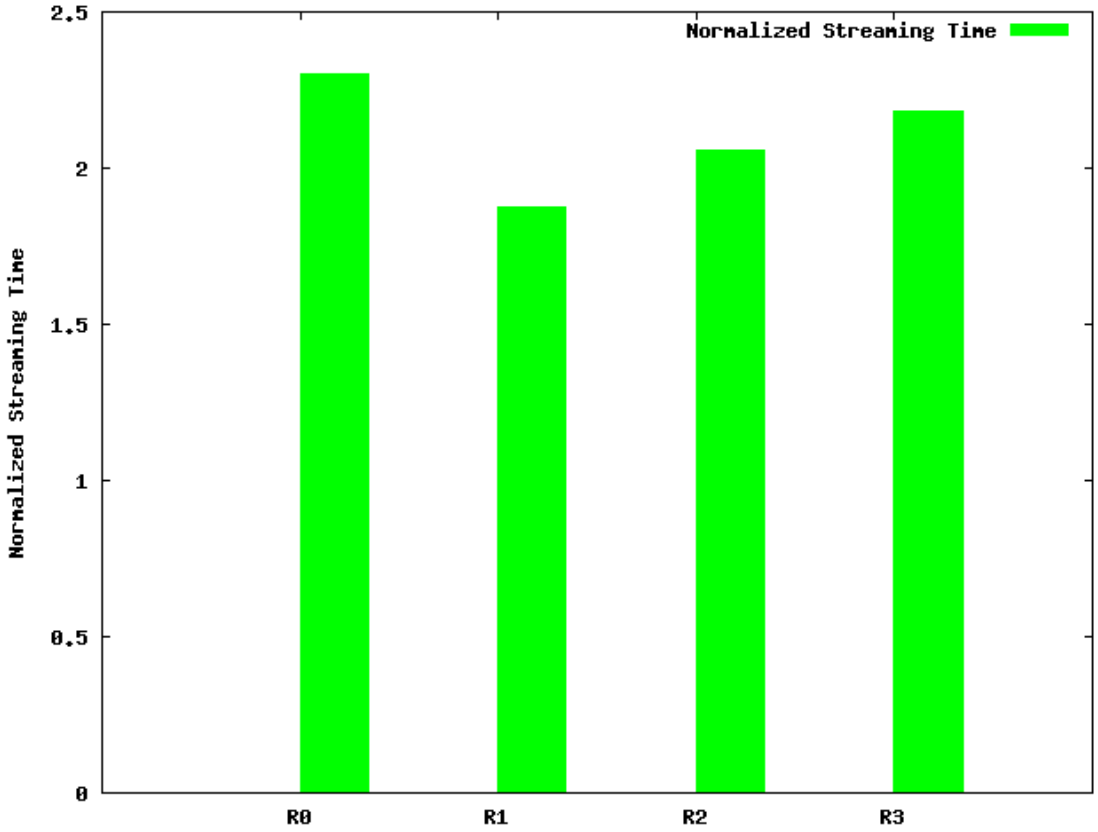

Figura 12.8: Streaming: valori medi

Il grafico 12.9 a pagina 115 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il tempo di risposta espresso in secondi (*response time in seconds*). Il tempo di risposta è calcolato rispetto all'intervallo tra il momento in cui l'utente invia la richiesta ed il momento in cui l'agente inizia ad espletare tale richiesta utilizzando le risorse della Grid. Da una lettura del grafico si evince che il tempo di risposta tende a crescere notevolmente nel corso della simulazione e, se comparato col grafico 12.4 a pagina 109, si nota che si hanno valori anche superiori

rispetto a quelli ottenuti nello scenario di sottomissione diretta di job riportato in sezione 12.1. I tempi di risposta superiori per singoli stream sono ascrivibili all'overhead introdotto dall'interazione tra i servizi dell'architettura e gli agenti, che se non accuratamente selezionati, danno luogo ad un'ulteriore crescita del tempo di risposta e conseguentemente, come si evince dal grafico 12.10, del tempo di transcoding del primo chunk.

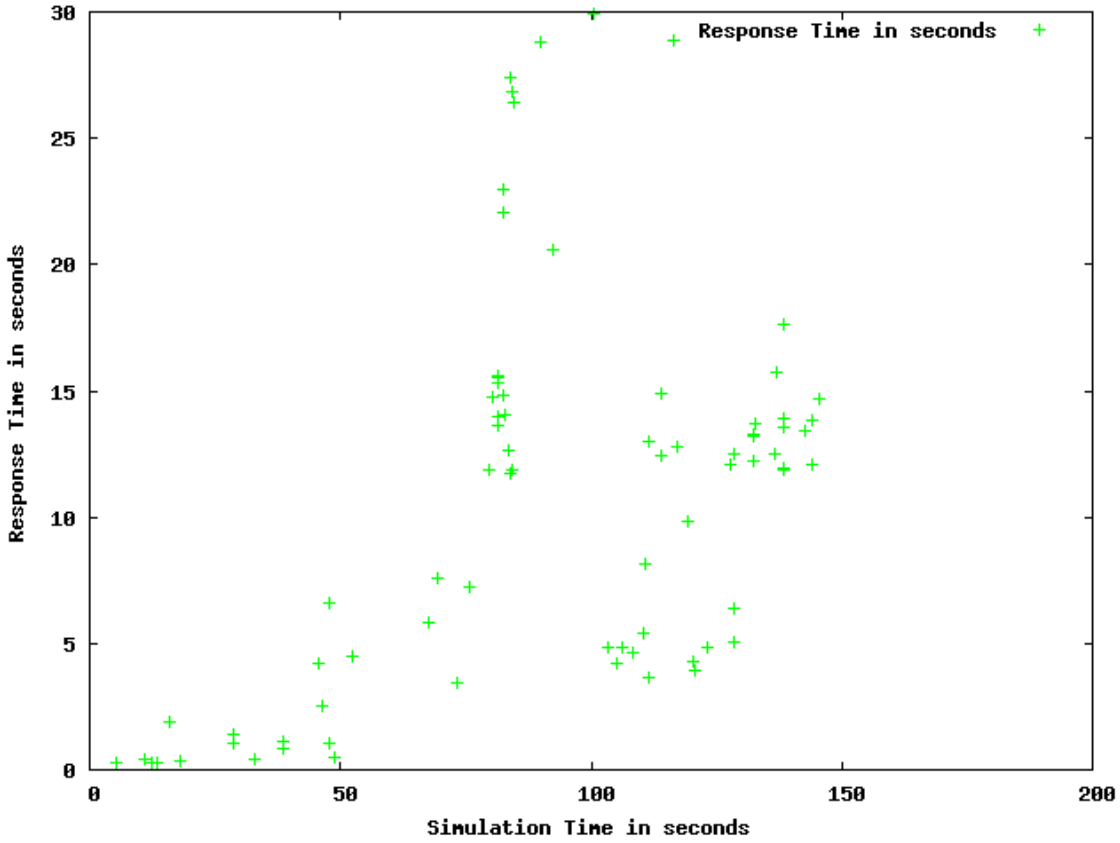

Figura 12.9: Tempo di risposta: singoli stream

Il grafico 12.10 a pagina 116 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il tempo di transcoding del primo chunk espresso in secondi (*first chunk time in seconds*). Il tempo di transcoding del primo chunk è comprensivo del tempo di risposta ed è calcolato rispetto all'intervallo tra il momento in cui l'utente invia il job e il momento in cui il job completa il transcoding del primo chunk del contenuto multimediale richiesto dall'utente.

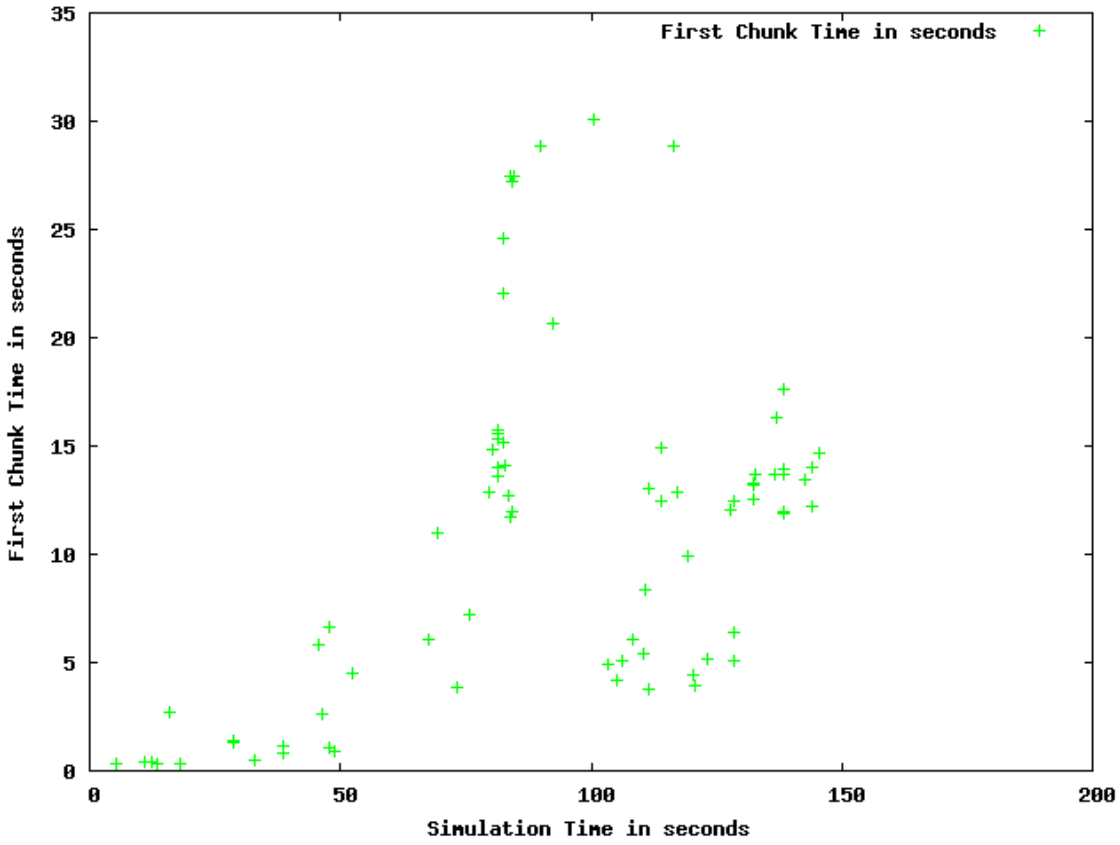

Figura 12.10: Primo chunk: singoli stream

Il grafico 12.11 a pagina 117 presenta le medie dei valori riportati nei grafici 12.9 e 12.10; in ascissa si ha il router cui uno o più utenti sono collegati e in ordinata i valori medi del tempo di risposta e del tempo di transcoding del primo chunk durante l'intera simulazione. Da una lettura del grafico si evince che in media il tempo di transcoding del primo chunk differisce di poco dal tempo di risposta anche in assenza dell'utilizzo dell'euristica di prossimità e questo è spiegabile con l'utilizzo e riutilizzo di agenti per la sottomissione di richieste di transcoding alla Grid.

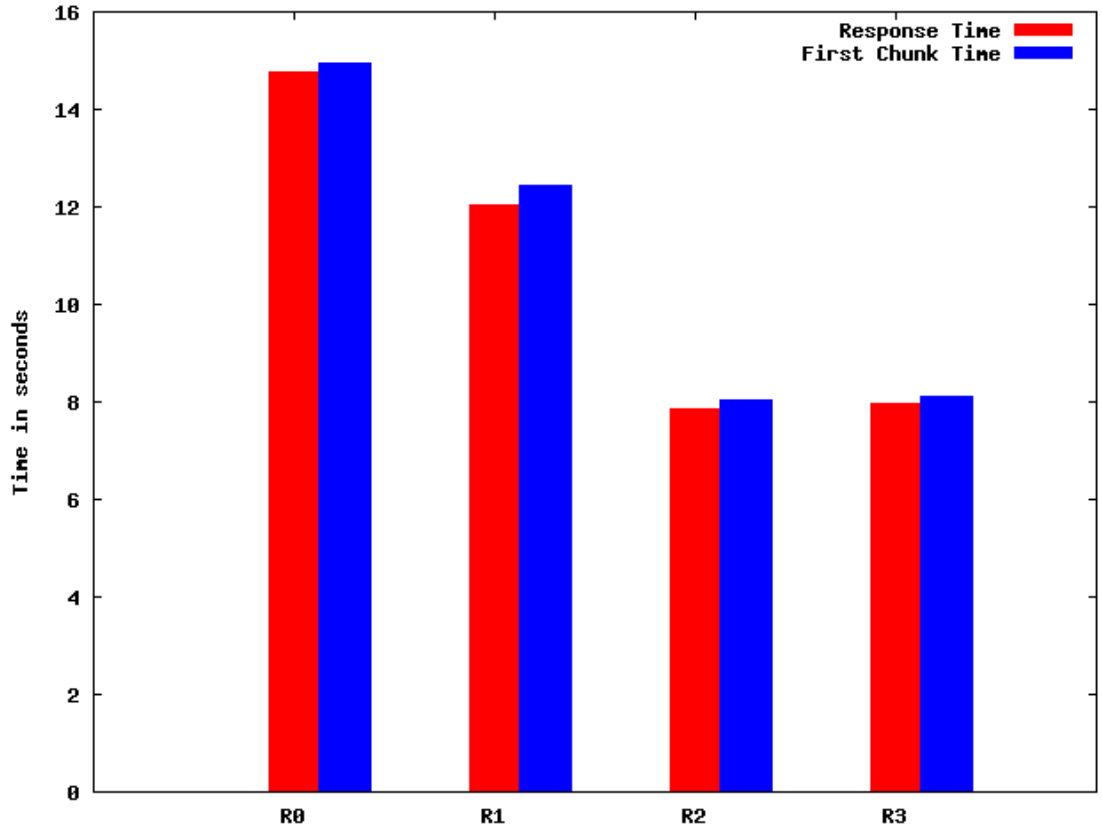

Figura 12.11: Tempi di risposta e transcoding del primo chunk: valori medi

12.3 La soluzione proposta inclusiva dell'euristica di prossimità

In questo scenario l'utente invia al servizio TraS (vedasi sezione 5.2.1 a pagina 36) la richiesta di un contenuto multimediale specificando un livello minimo di QoS. Il servizio TraS utilizza l'euristica di prossimità nel selezionare il Computing Element e lo Storage Element che intervengono nell'espletamento della richiesta.

Il grafico 12.12 a pagina 119 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il rapporto tra il tempo di ricezione dell'intero stream e la durata dello stream (*normalized streaming time*). Al fine di ottenere una ricezione fluida degli stream, condizione necessaria è che il valore dell'ordinata si mantenga minore o al più uguale ad una soglia accettabile pari ad 1; dalla lettura del grafico si evince che tale valore si mantiene sempre sotto tale soglia. Tali valori accettabili sono ottenuti grazie ai seguenti fattori concomitanti:

1. le latenze di rete tra utente e Computing Element e tra Computing Element e Storage Element, sono ottimizzate grazie al ricorso all'euristica di prossimità

2. l'overhead introdotto dall'attività di online transcoding è ridotto essendo effettuata sulle risorse più appropriate in quanto l'euristica di prossimità tiene conto, oltre che delle latenze di rete, del carico di Computing Element e Storage Element

3. i job di online transcoding e streaming sono gestiti dall'interno della Grid dai servizi e dagli agenti ponendosi come delegati dell'utente per la sottomissione di tali job alla Grid

La riduzione di oltre il 400% dei tempi di streaming rispetto allo scenario considerato in sezione 12.2 sono dovuti al ricorso all'euristica di prossimità che fornisce un beneficio cumulativo per un utilizzo ottimale delle risorse della Grid e il soddisfacimento delle richieste utente con relative specifiche di qualità del servizio.

Il grafico 12.13 a pagina 120 presenta le medie dei valori riportati nel grafico 12.12; in ascissa si ha il router cui uno o più utenti sono collegati e in ordinata il valore medio durante l'intera simulazione del rapporto tra il tempo di ricezione dell'intero stream e la durata dello stream (*normalized streaming time*). In questo caso si ha una riduzione

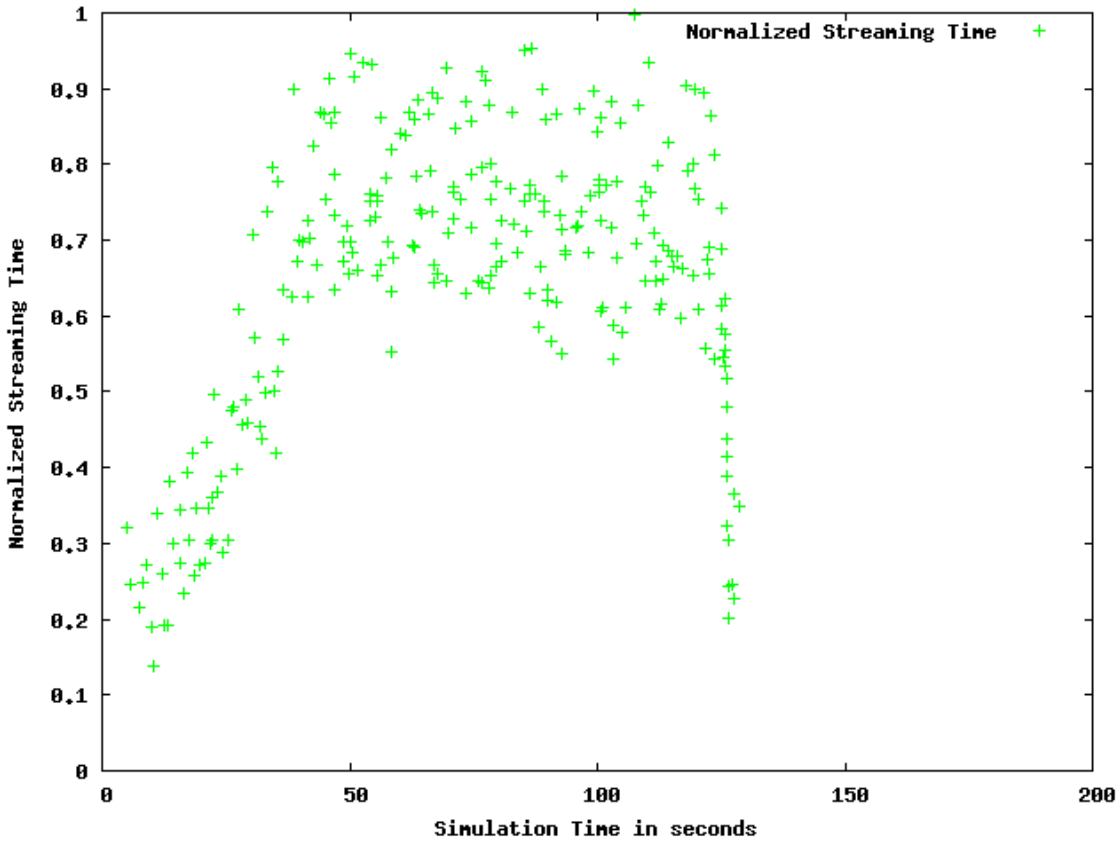

Figura 12.12: Streaming: singoli stream

tra il 400% e l'800% dei tempi medi di streaming rispetto allo scenario considerato in sezione 12.2. Da una lettura combinata dei grafici 12.13,12.12,12.8,12.7 si evince inoltre che la soluzione proposta, anche in assenza di euristica di prossimità ma a maggior ragione con il ricorso ad essa, introduce un miglioramento che si rende più palpabile durante il corso della simulazione e spiegabile in virtù del riutilizzo di agenti già presenti sulla Grid (a tal proposito vedasi anche il grafico 12.16 a pagina 123).

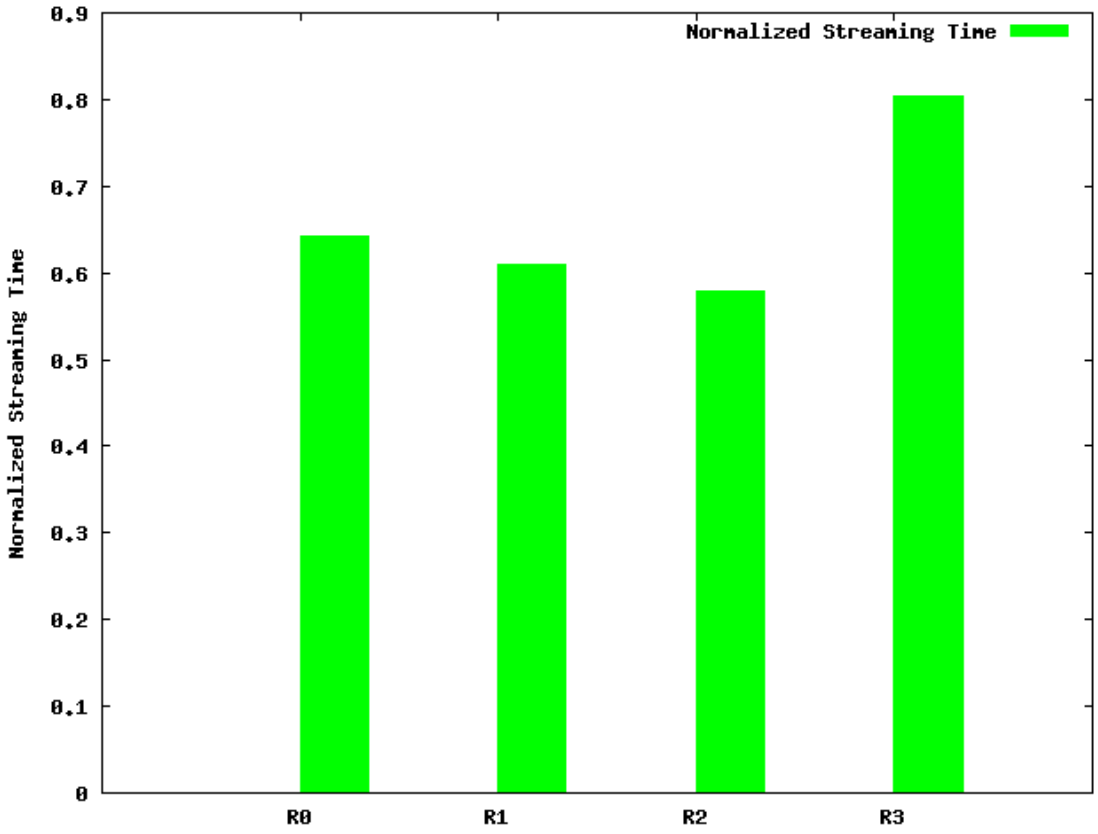

Figura 12.13: Streaming: valori medi

Il grafico 12.14 a pagina 121 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il tempo di risposta espresso in secondi (*response time in seconds*). Il tempo di risposta è calcolato rispetto all'intervallo tra il momento in cui l'utente invia la richiesta ed il momento in cui l'agente inizia ad espletare tale richiesta utilizzando le risorse della Grid. Da una lettura del grafico si evince

che il tempo di risposta tende ad aumentare al crescere delle richieste e a stabilizzarsi nel corso della simulazione.

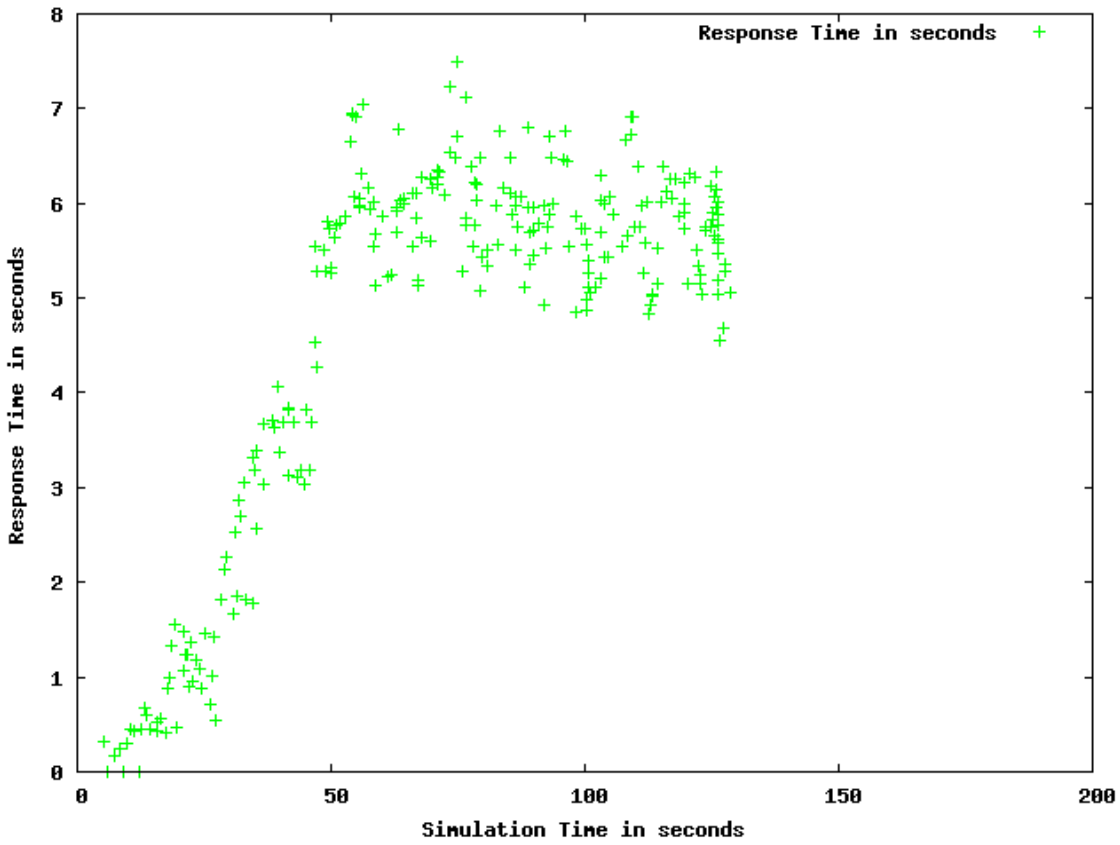

Figura 12.14: Tempo di risposta: singoli stream

Il grafico 12.15 a pagina 122 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il tempo per di transcoding del primo chunk espresso in secondi (*first chunk time in seconds*). Il tempo di transcoding del primo chunk è comprensivo del tempo di risposta ed è calcolato rispetto all'intervallo tra il momento in cui l'utente invia il job e il momento in cui il job completa il transcoding del primo chunk del contenuto multimediale richiesto dall'utente.

Il grafico 12.16 a pagina 123 presenta le medie dei valori riportati nei grafici 12.14 e 12.15; in ascissa si ha il router cui uno o più utenti sono collegati e in ordinata i valori medi del tempo di risposta e del tempo di transcoding del primo chunk durante l'intera

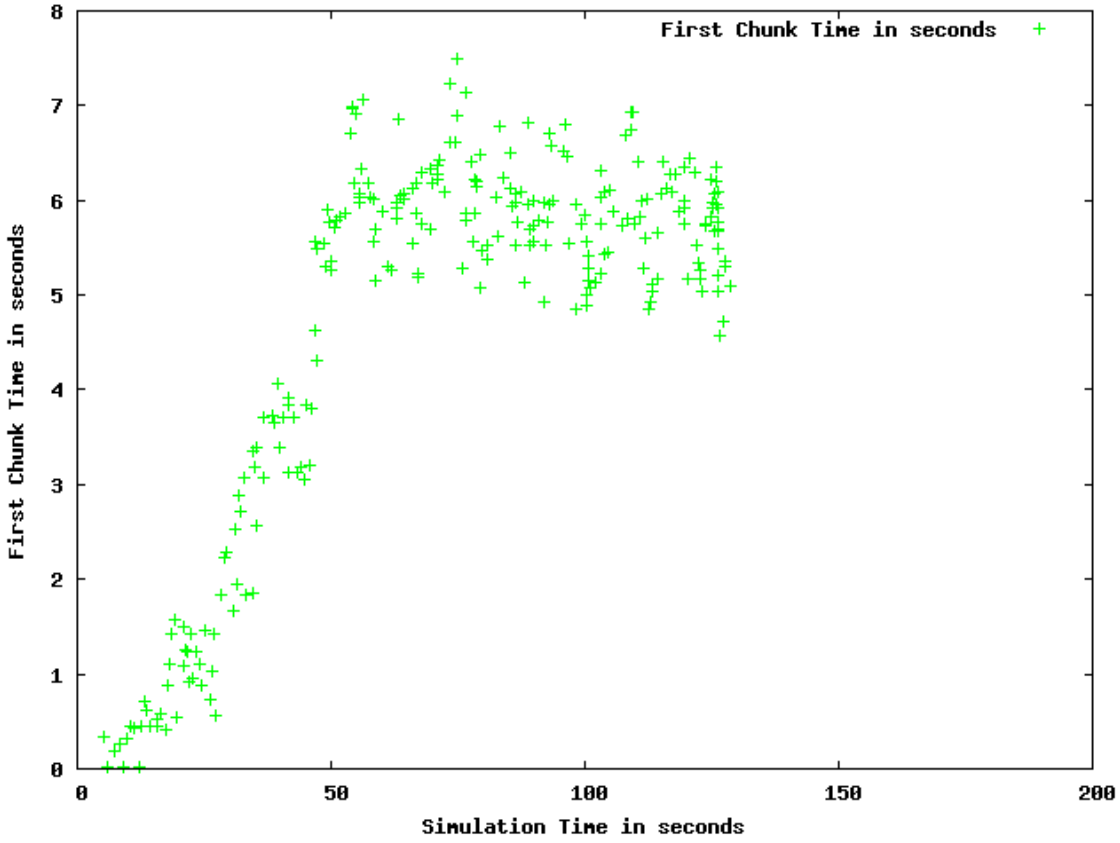

Figura 12.15: Primo chunk: singoli stream

simulazione. Da una lettura del grafico si evince che in media il tempo di transcoding del primo chunk differisce di poco dal tempo di risposta e ciò è spiegabile con l'utilizzo e riutilizzo di agenti per la sottomissione di richieste di transcoding alla Grid.

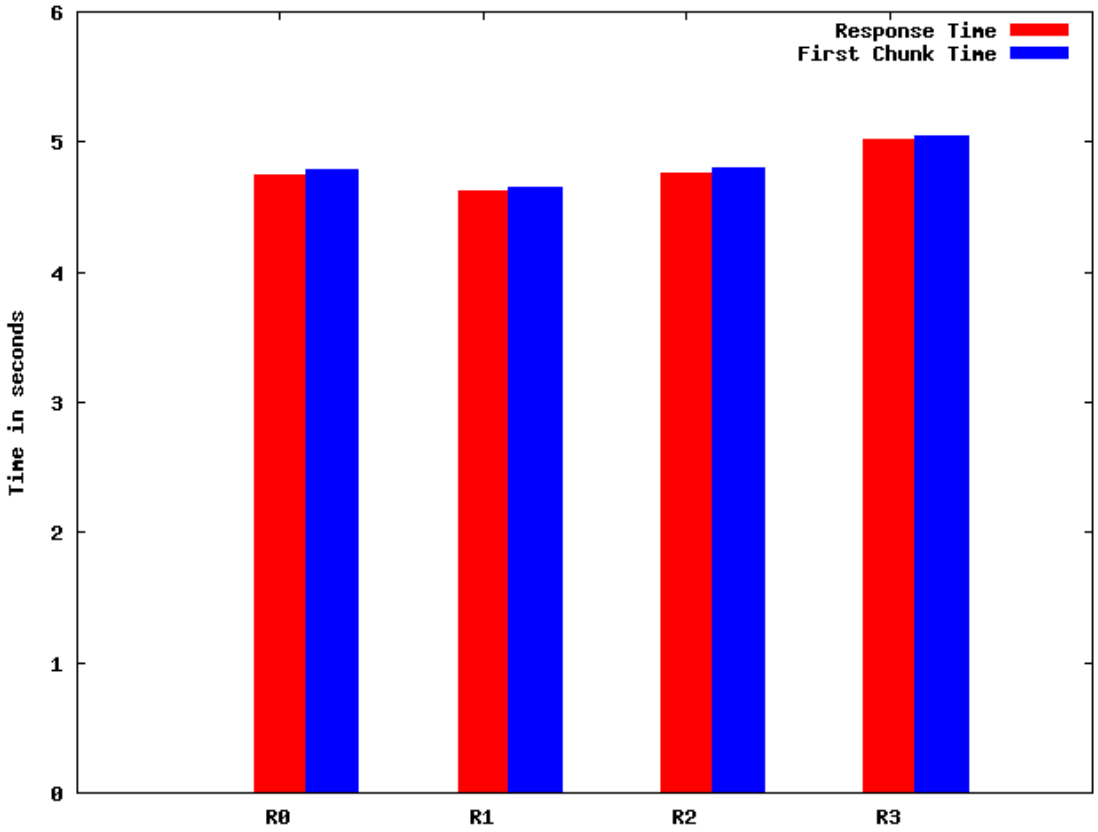

Figura 12.16: Tempi di risposta e transcoding del primo chunk: valori medi

12.4 Confronto tra i tre scenari

In questa sezione viene presentato un confronto dei risultati ottenuti dalla simulazione dei tre scenari:

1. sottomissione diretta dei jobs di online transcoding e streaming da parte degli utenti (vedasi sezione 12.1 a pagina 105) individuato nei grafici che seguono con il colore rosso e legenda *Direct Submission*

2. la soluzione proposta in assenza dell'euristica di prossimità (vedasi sezione 12.2 a pagina 111) individuato nei grafici che seguono con il colore blu e legenda *Random (CE, SE)*

3. la soluzione proposta inclusiva dell'euristica di prossimità (vedasi sezione 12.3 a pagina 118) individuato nei grafici che seguono con il colore verde e legenda *Proximity Heuristic*

Il grafico 12.17 a pagina 125 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il rapporto tra il tempo di ricezione dell'intero stream e la durata dello stream (*normalized streaming time*) per i tre scenari considerati. Al fine di ottenere una ricezione fluida degli stream, condizione necessaria è che il valore dell'ordinata si mantenga minore o al più uguale ad una soglia accettabile pari ad 1 indicata nel grafico con la linea arancione denominata in legenda come *streaming time threshold*.

Da una lettura del grafico si evince che la soluzione proposta, i cui valori di tempo di streaming sono rappresentati con delle crocette verdi, rispetti sempre nella specifica simulazione la soglia accettabile sul tempo di streaming mentre gli altri due scenari simulati non la rispettino se non per singoli stream isolati.

Il grafico 12.18 a pagina 126 presenta le medie dei valori riportati nel grafico 12.17; in ascissa si ha il router cui uno o più utenti sono collegati e in ordinata il valore medio durante l'intera simulazione del rapporto tra il tempo di ricezione dell'intero stream e la durata dello stream (*normalized streaming time*).

Il grafico 12.19 a pagina 127 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il tempo di risposta espresso in secondi (*response time in seconds*). Il tempo di risposta è calcolato rispetto all'intervallo tra il momento in cui l'utente invia la richiesta ed il momento in cui l'agente inizia ad espletare tale richiesta utilizzando le risorse della Grid. Il grafico 12.20 a pagina 128 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata il tempo di transcoding del primo chunk espresso in secondi (*first chunk time in seconds*). Il tempo di transcoding del primo chunk è comprensivo del tempo di risposta ed è calcolato rispetto all'intervallo tra il momento in cui l'utente invia il job e il momento in cui il job completa il transcoding del primo chunk del contenuto

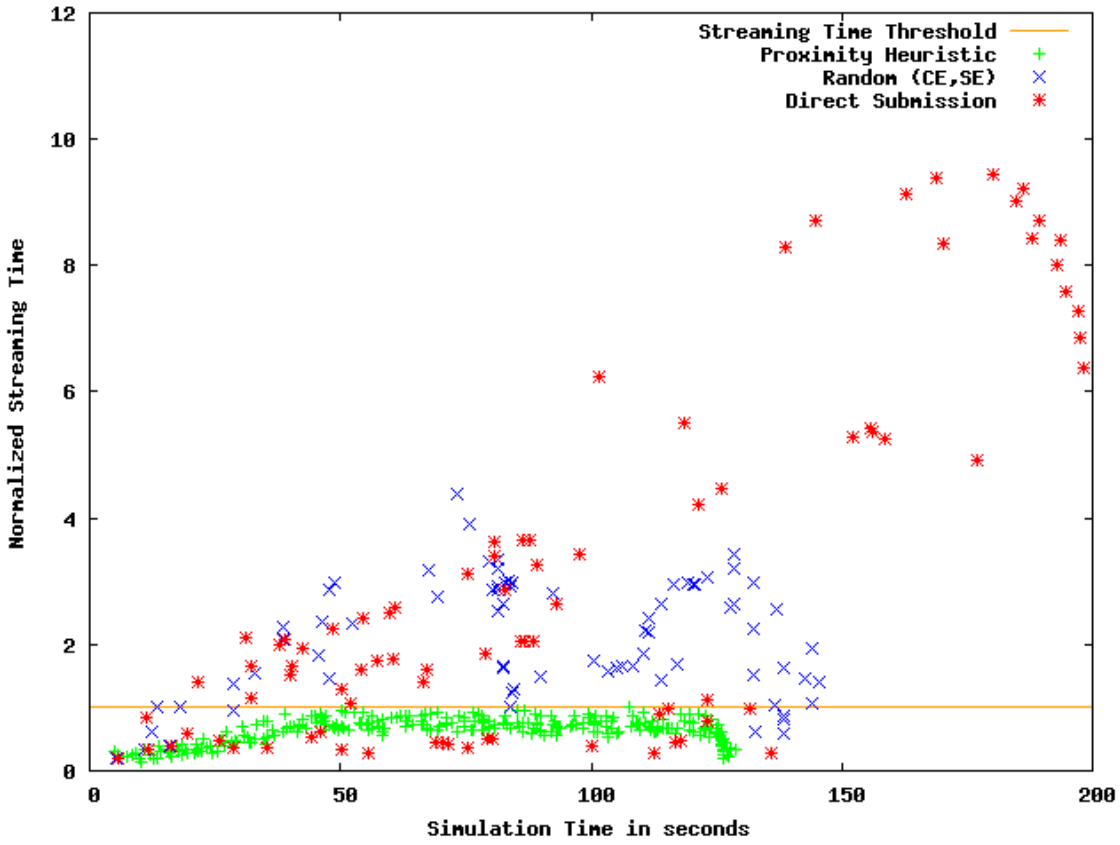

Figura 12.17: Streaming: singoli stream

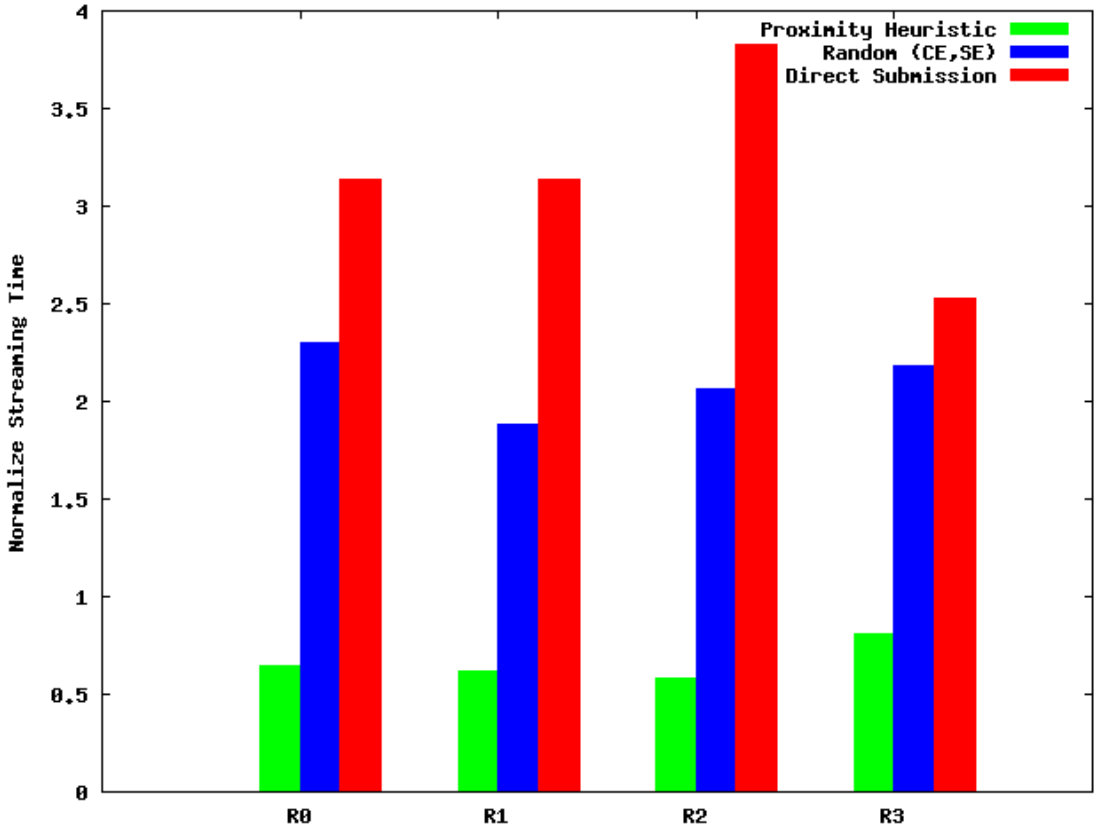

Figura 12.18: Streaming: valori medi

multimediale richiesto dall'utente. Da una lettura di questi due grafici si evince che solo con la soluzione proposta il tempo di risposta e il tempo di transcoding del primo chunk tendano a stabilizzarsi nel corso della simulazione.

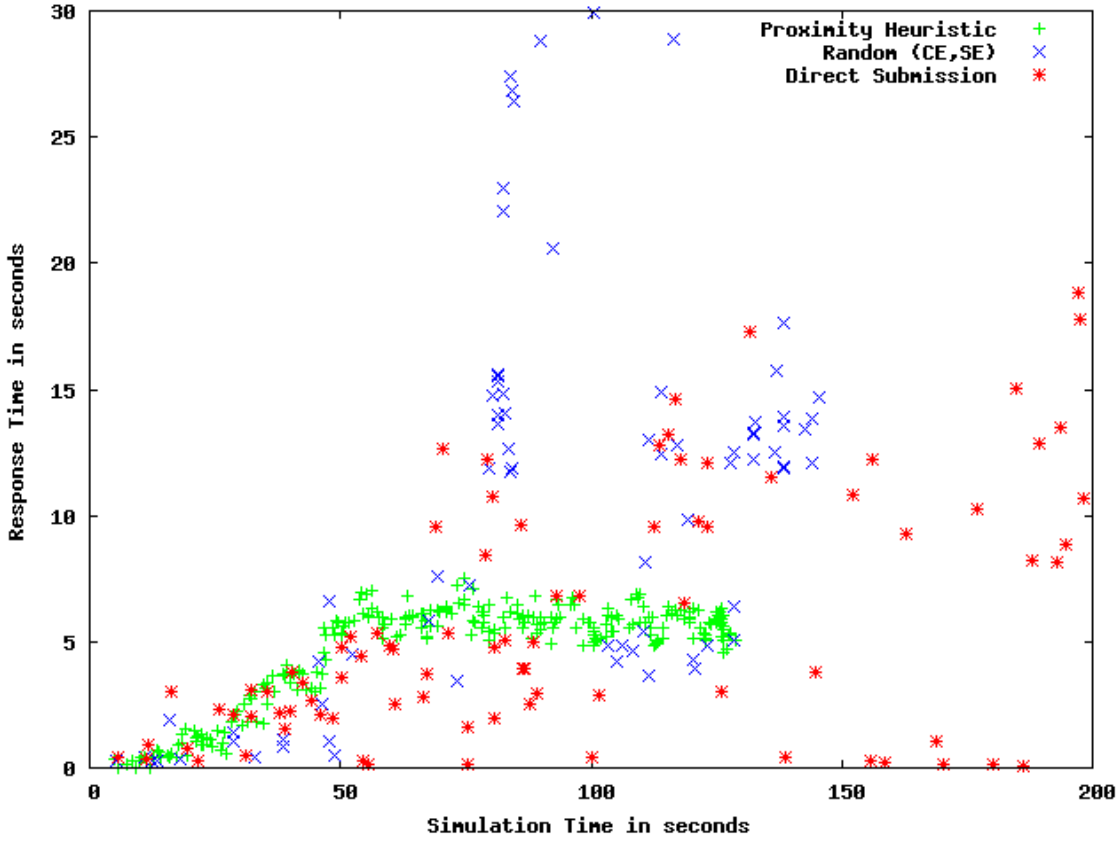

Figura 12.19: Tempi di risposta

Il grafico 12.21 a pagina 130 riporta in ascissa il variare del tempo di simulazione espresso in secondi (*simulation time in seconds*) e in ordinata la perdita di qualità degli stream (*quality loss*). Nel grafico è presente una linea arancione (in legenda *Quality Loss Threshold*) che rappresenta la soglia QoS_{vmin} di perdita di qualità ritenuta accettabile dagli utenti per tutte le richieste. Si è scelto di utilizzare un unico valore di QoS_{vmin} al fine di rendere comparabili le simulazioni dei tre scenari anche in relazione alle modalità ed entità di adattamento della Quality of Service per tutti gli stream serviti. Da una prima lettura del grafico si nota subito che per la soluzione proposta non è presente

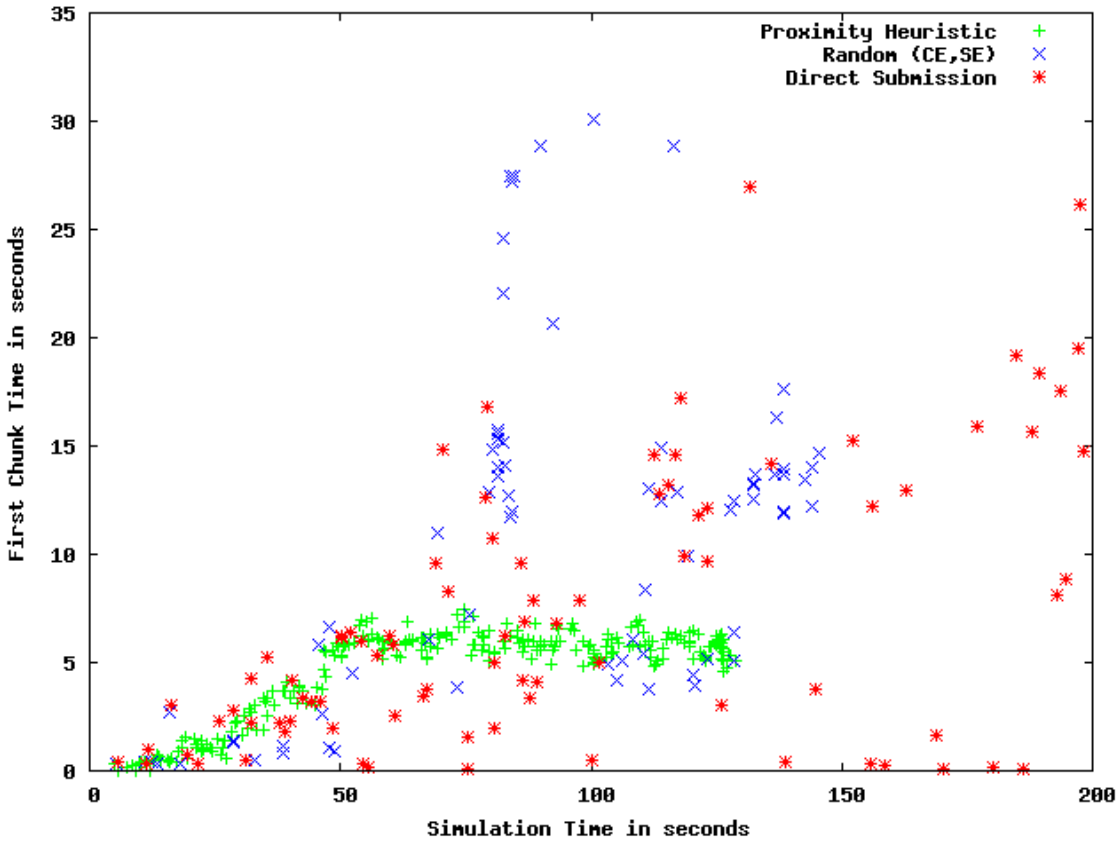

Figura 12.20: Tempi di transcoding del primo chunk

alcuna perdita di qualità; questo è dovuto all'orchestrazione intervenuta nella gestione delle richieste utente in modo tale che anche l'attività di adattamento della Quality of Service non si è resa necessaria nel corso dell'intera simulazione. Inoltre si nota come nello scenario di sottomissione diretta rappresentato con la curva rossa (in legenda *Direct Submission*) si ha un'iniziale crescita della perdita di qualità e poi una caduta; ciò è dovuto al fatto che il degrado della qualità degli stream non è sufficiente ad ottenere stream fluidi lato utente anche perché tale attività, se da un lato riduce la dimensione dei frame ricevuti dall'utente, dall'altro introduce un overhead di online transcoding effettuato su risorse inappropriate. Nello scenario visto in sezione 12.2, ove si considera la soluzione proposta in assenza di euristica di prossimità, la curva relativa alla perdita di qualità mostra un andamento instabile sebbene non presenti indicazioni di una caduta dovuta a saturazione del carico degli host; ciò è spiegabile col bilanciamento tra il ritardo introdotto da host di transcoding non ottimali e l'utilizzo e riutilizzo degli agenti nella gestione delle richieste di transcoding alla Grid e alle proprietà di mobilità proprie degli agenti.

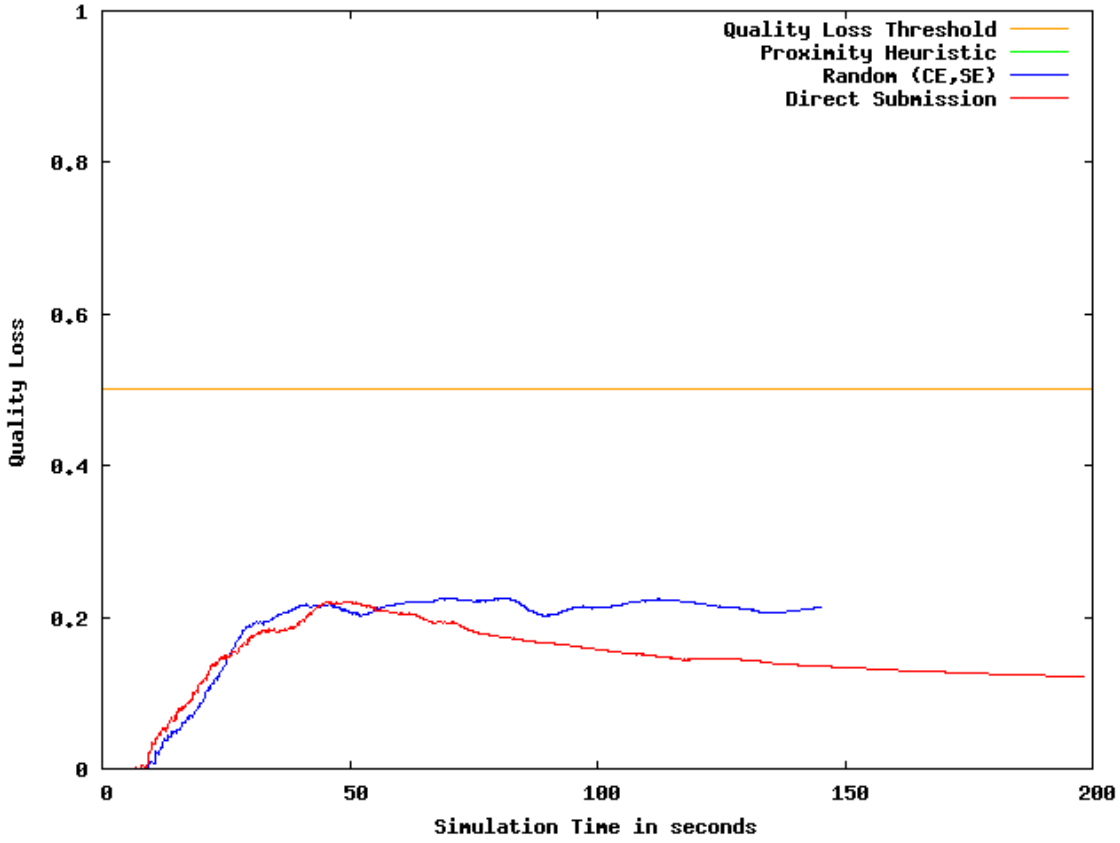

Figura 12.21: QoS loss

Parte V

Conclusioni

In questa tesi è stata presentata l'attività di ricerca da me condotta nell'ambito dell'integrazione tra tecnologie multi-agente e ambienti di Grid Computing per la streaming di contenuti multimediali e adattamento degli stessi con attività CPU-intensive di online transcoding. La soluzione proposta è stata descritta, dopo aver motivato la scelta di tale sinergia in tale ambito, elencandone servizi, agenti, le loro principali interazioni, presentando l'euristica di prossimità che informa l'allocazione delle risorse e degli agenti necessari all'espletamento delle richieste utente secondo dei requisiti di QoS e descrivendo i meccanismi di adattamento della QoS informati da decisioni basate sul controllo fuzzy.

Inoltre è stata affrontata la problematica del deployment di tale soluzione implementata prototipalmente con il middleware multi-agente JADE su ambienti di Grid Computing basati su gLite, un'estensione al middleware Globus.

La soluzione proposta è stata sottoposta a sperimentazione tramite simulazione con la libreria GAP per ottenere feedback qualitativi e quantitativi sulle prestazioni di tale sinergia rispetto allo scenario considerato, mostrando come la sinergia tra tali tecnologie permetta di sfruttare le ingenti quantità di risorse offerte dagli ambienti di Grid Computing non solo per memorizzare e distribuire i contenuti multimediali ma anche per realizzarne l'online transcoding quando le caratteristiche e le condizioni della rete non supportino la trasmissione del contenuto originale o il client utente non disponga di un player adatto alla ricezione e presentazione del contenuto multimediale nel suo formato originario.

Inoltre la libreria GAP, inizialmente sviluppata per soddisfare le esigenze di simulazione relative all'ambito applicativo esaminato, è stata presentata come tool di simulazione utilizzabile per lo studio di sinergie in senso lato tra tecnologie multi-agente e di Grid Computing.

L'attività svolta offre diversi spunti di lavoro. In primo luogo sarebbe interessante giungere ad un'implementazione di produzione della soluzione proposta.

Inoltre si potrebbe estendere la soluzione proposta ampliando le capacità di collaborazione e di adattamento degli agenti inserendo meccanismi di comunicazione basati sul paradigma P2P.

Un ulteriore spunto di lavoro potrebbe essere quello di rivisitare la libreria GAP al fine di rendere la simulazione parallelizzabile su più computer riducendo le limitazioni del paradigma di simulazione sequenziale adottato dalla sottostante libreria GridSim.

Bibliografia

[1] Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.

[2] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Jounral of Supercomputer Applications*, 15(3), 2001.

[3] Ian T. Foster. The anatomy of the grid: Enabling scalable virtual organizations. In *CCGRID* [149], pages 6–7.

[4] Thomas A. DeFanti, Ian Foster, Michael E. Papka, Rick Stevens, and Tim Kuhfuss. Overview of the I-WAY: Wide-area visual supercomputing. *The International Journal of Supercomputer Applications and High Performance Computing*, 10(2/3):123–131, Summer/Fall 1996.

[5] Rick Stevens, Paul R. Woodward, Thomas A. DeFanti, and Charles E. Catlett. From the i-way to the national technology grid. *Commun. ACM*, 40(11):50–60, 1997.

[6] Ian T. Foster, Jonathan Geisler, Bill Nickless, Warren Smith, and Steven Tuecke. Software infrastructure for the i-way metacomputing experiment. *Concurrency - Practice and Experience*, 10(7):567–581, 1998.

[7] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[8] Ian T. Foster and Carl Kesselman. The globus project: a status report. *Future Generation Comp. Syst.*, 15(5-6):607–621, 1999.

[9] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Softw., Pract. Exper.*, 32(2):135–164, 2002.

[10] Omer F. Rana, Ali Shaikhali, and Gregor von Laszewski. *Grid Computing: A Practical Guide to Technology and Applications*, chapter Business Value of Grid Computing, pages 31–41. Charles River Media, Hingham, MA, 2003.

[11] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[12] Global Grid Forum Open Grid Services Architecture Working Group. *Defining the Grid: A Roadmap for OGSA Standards v 1.0*, September 2005. INFO.

[13] F. Gagliardi, B. Jones, and E. Laure. The EU DataGrid Project: Building and Operating a large scale Grid Infrastructure. In B. Di Martino, J. Dongarra, A. Hoisie, L.Y. Yang, and H. Zima, editors, *Engineering the Grid: Status and Perspective.* American Scientific Publishers, January 2006.

[14] Michael Wooldridge and Nicholas R. Jennings. Agent theories, architectures, and languages: A survey. In Michael Wooldridge and Nicholas R. Jennings, editors, *ECAI Workshop on Agent Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in Computer Science*, pages 1–39. Springer, 1994.

[15] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with jade. In Cristiano Castelfranchi and Yves Lespérance, editors, *ATAL*, volume 1986 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2000.

[16] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE.* Wiley, 2007.

[17] E. Kasutani and T. Ebrahimi. Investigation Report on Universal Multimedia Access. Technical report, Ecublens, 2004.

[18] Anthony Vetro. Mpeg-21 digital item adaptation: Enabling universal multimedia access. *IEEE MultiMedia*, 11(1):84–87, 2004.

[19] Katashi Nagao. *Digital Content Annotation and Transcoding.* Artech House, Inc., Norwood, MA, USA, 2003.

[20] Yonghyun Hwang, Jihong Kim, and Eunkyong Seo. Structure-aware web transcoding for mobile devices. *IEEE Internet Computing*, 7(5):14–21, 2003.

[21] Hoi Ka Kit and Dik Lun Lee. Document visualization on small displays. In *WWW Posters*, 2001.

[22] Vetro Anthony and Huifang Sun. Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, 20(2):18–29, March 2003.

[23] Shih-Fu Chang and Anthony Vetro. Video adaptation: Concepts, technologies, and open issues. *Special Issue on Advances in Video Coding and Delivery, Proceedings of IEEE*, 2005.

[24] Huifang Sun, Tihao Chiang, and Xuemin Chen. *Digital Video Transcoding for Transmission and Storage.* CRC Press, Inc., Boca Raton, FL, USA, 2004.

[25] David Salomon and David Salomon. *Data Compression: The Complete Reference.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[26] Gilbert Strang. The discrete cosine transform. *SIAM Review*, 41(1):135–147, 1999.

[27] J. Yeh, M. Vetterli, and M. Khansari. Motion compensation of motion vectors. In *ICIP '95: Proceedings of the 1995 International Conference on Image Processing (Vol. 1)-Volume 1*, page 574, Washington, DC, USA, 1995. IEEE Computer Society.

[28] Huahui Wu, Mark Claypool, and Robert Kinicki. Guidelines for selecting practical mpeg group of pictures. In *IMSA'06: Proceedings of the 24th IASTED international conference on Internet and multimedia systems and applications*, pages 61–66, Anaheim, CA, USA, 2006. ACTA Press.

[29] J.-R. Ohm. Advances in scalable video coding. 93:42–56, 2005.

[30] Weiping Li. Overview of fine granularity scalability in mpeg-4 video standard. *IEEE Trans. Circuits Syst. Video Techn.*, 11(3):301–317, 2001.

[31] Thomas Wiegand, Gary J. Sullivan, Jens-Rainer Ohm, and Ajay Luthra. Introduction to the special issue on scalable video coding-standardization and beyond. *IEEE Trans. Circuits Syst. Video Techn.*, 17(9):1099–1102, 2007.

[32] Gang Peng. Cdn: Content distribution network. Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, 2003.

[33] George Pallis and Athena Vakali. Insight and perspectives for content delivery networks. *Commun. ACM*, 49(1):101–106, 2006.

[34] Athena Vakali and George Pallis. Content delivery networks: Status and trends. *IEEE Internet Computing*, 7(6):68–74, 2003.

[35] A. Pathan and R. Buyya. A taxonomy and survey of content delivery networks. Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia., Feb. 2007.

[36] M. Day, B. Cain, G. Tomlinson, and P. Rzewski. A model for content internetworking (cdi). Technical report, Internet Engineering Task Force RFC 3466, February 2003.

[37] A. Barbir, B. Cain, R. Nair, and O. Spatscheck. Known Content Network (CN) Request-Routing Mechanisms. RFC 3568 (Informational), July 2003.

[38] Swaminathan Sivasubramanian, Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Replication for web hosting systems. *ACM Computing Surveys*, to appear. `http://www.globule.org/publi/RWHS_cs.html`.

[39] Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Netairt: A DNS-based redirection system for Apache. In *Proceedings of the IADIS International Conference on WWW/Internet*, Algarve, Portugal, November 2003. `http://www.globule.org/publi/NDBRSA_icwi2003.html`.

[40] Giovanni Novelli, Giuseppe Pappalardo, Corrado Santoro, and Emiliano Tramontana. Transcoding agents for multimedia content delivery in a grid. *ITSSA*, 2(3):281–288, 2006.

[41] Fabrizio Messina, Giovanni Novelli, Giuseppe Pappalardo, Corrado Santoro, and Emiliano Tramontana. A qos-aware architecture for multimedia content provisioning in a grid environment. In Flavio De Paoli, Antonella Di Stefano, Andrea Omicini, and Corrado Santoro, editors, *WOA*, volume 204 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[42] Giovanni Novelli, Giuseppe Pappalardo, Corrado Santoro, and Emiliano Tramontana. A grid-based infrastructure to support multimedia content distribution. In Giancarlo Fortino, Carlo Mastroianni, and Guillaume Pierre, editors, *UPGRADE-CN*, pages 57–64. ACM, 2007.

[43] Anthony Vetro. Transcoding, scalable coding, and standardized metadata. In *VLBV*, pages 15–16, 2003.

[44] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *Jade Programmer's Guide*, June 2007.

[45] Nuno Santos and Birger Koblitz. Distributed metadata with the amga metadata catalog. *CoRR*, abs/cs/0604071, 2006.

[46] Birger Koblitz, Nuno Santos, and V. Pose. The amga metadata service. *J. Grid Comput.*, 6(1):61–76, 2008.

[47] Yunyoung Nam and Eenjun Hwang. Real-time transcoding of mpeg videos in a distributed environment. In Liew et al. [150], pages 280–283.

[48] Gerassimos D. Barlas. A taxonomy and dlt-based analysis of cluster-based video trans/en-coding. In *PDP*, pages 388–395. IEEE Computer Society, 2006.

[49] Abmar Abbas and Amhar Abbas. *Grid Computing: A Practical Guide to Technology and Applications*, chapter 12: Grid-Enabling Software Applications.

[50] no F. J. Gonzalez-Casta R. Asorey-Cacheda, R. P. Martinez-Alvarez, na-Seijo F. Comesa and J. Vales-Alonso. Dvd transcoding via linux metacomputing. *Linux J.*, 2003(116):8, 2003.

[51] Yi-Cheng Tu, Jingfeng Yan, Gang Shen, and Sunil Prabhakar. Multiquality data replication in multimedia databases. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):679–694, 2007.

[52] Felix Hartanto, Jussi Kangasharju, Martin Reisslein, and Keith W. Ross. Caching video objects: Layers vs versions?

[53] Cristian Koliver, Klara Nahrstedt, Jean-Marie Farines, Joni da Silva Fraga, and Sandra Aparecida Sandri. Specification, mapping and control for qos adaptation. *Real-Time Systems*, 23(1-2):143–174, 2002.

[54] Rudinei Goularte, Rodrigo F. De Mello, Evgueni Dodonov, and Laurence Tianruo Yang. A model to estimate transcoding costs applied in live-video transmissions. *ipc*, 0:341–348, 2007.

[55] Vidyut Samanta, Ricardo V. Oliveira, Advait Dixit, Parixit Aghera, Petros Zerfos, and Songwu Lu. Impact of video encoding parameters on dynamic video transcoding. In *COMSWARE*. IEEE, 2006.

[56] Nick G. Duffield, K. K. Ramakrishnan, and Amy R. Reibman. Save: an algorithm for smoothed adaptive video over explicit rate networks. *IEEE/ACM Trans. Netw.*, 6(6):717–728, 1998.

[57] B. Li and K. Nahrstedt. A control-based middleware framework for quality of service adaptations, 1999.

[58] D. Wu, Y. Hou, W. Zhu, H. Lee, T. Chiang, Y. Zhang, and H. Chao. On end-to-end architecture for transporting mpeg-4 video over the internet, 2000.

[59] Panagiotis Papadimitriou Sofia. Multimedia streaming over the internet.

[60] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An introduction to fuzzy control.* Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[61] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[62] Stephen Wolf and Margaret Pinson. Ntia report 02-392: Video quality measurement techniques. Technical report, Institute for Telecommunication Sciences.

[63] Charles Krasic and Jonathan Walpole. QoS scalability for streamed media delivery. Technical Report CSE-99-011, 17, 1999.

[64] G. Ghinea and J. P. Thomas. Qos impact on user perception and understanding of multimedia video clips. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 49–54, New York, NY, USA, 1998. ACM.

[65] Giuseppe Andronico, Valeria Ardizzone, Roberto Barbera, Rosanna Catania, Antonio Carrieri, Alberto Falzone, Emidio Giorgio, Giuseppe La Rocca, Salvatore Monforte, Marco Pappalardo, Gianluca Passaro, and Giusepe Platania. Gilda: The grid infn virtual laboratory for dissemination activities. In *TRIDENTCOM*, pages 304–305, 2005.

[66] G Andronico. Overview of the glite middleware. (TEST-OVERVIEW-2007-002), 2006.

[67] Roberto Alfieri, Roberto Barbera, Patrizia Belluomo, Alessandro Cavalli, Roberto Cecchini, Andrea Chierici, Vincenzo Ciaschini, Luca dell'Agnello, Flavia Donno, Enrico Ferro, Antonio Forte, Luciano Gaido, Antonia Ghiselli, Alberto Gianoli, Alessandro Italiano, Stefano Lusso, Marisa Luvisetto, Paolo Mastroserio, Mirco Mazzucato, Daniele Mura, Mario Reale, Livio Salconi, Giuseppe Sava, Marco Serra, Fabio Spataro, Francesco Taurino, Gennaro Tortone, Luca Vaccarossa, Marco Verlato, and Giulia Vita Finzi. The infn-grid testbed. *Future Generation Comp. Syst.*, 21(2):249–258, 2005.

[68] Francesca Lo Piccolo, Giuseppe Bianchi, and Stefano Salsano. Measurement study of the mobile agent jade platform. In *WOWMOM*, pages 638–646. IEEE Computer Society, 2006.

[69] Fabio Bellifemine. Fipa: a standard for agent interoperability. In *WOA*, page 121, 2000.

[70] R. Bruno. glite user interface installation and configuration. (TEST-GRIDSERVICESINSTALLATION-2006-004), 2006.

[71] Jean-Philippe B. Baud, James Caey, Sophie Lemaitre, Caitriana Nicholson, David Smith, and Graeme Stewart. Lcg data management: From edg to egee. In *UK e-Science All Hands Meeting, Nottingham, UK*, 2005.

[72] Graeme A Stewart, David Cameron, Greig A Cowan, and Gavin McCance. Storage and data management in egee. In *ACSW '07: Proceedings of the fifth Australasian symposium on ACSW frontiers*, pages 69–77, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[73] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke. GridFTP Protocol Specification. Ggf gridftp working group document, September 2002.

[74] William Allcock, John Bresnahan, Rajkumar Kettimuthu, and Michael Link. The globus striped gridftp framework and server. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 54, Washington, DC, USA, 2005. IEEE Computer Society.

[75] Rashedur M. Rahman, Ken Barker, and Reda Alhajj. Predicting the performance of gridftp transfers. In *IPDPS* [151].

[76] Gregor von Laszewski, Ian Foster, Jarek Gawor, and Peter Lane. A Java Commodity Grid Kit. *Concurrency and Computation: Practice and Experience*, 13(8-9):643–662, 2001.

[77] Kaizar Amin, Mihael Hategan, Gregor von Laszewski, and Nestor J. Zaluzec. Abstracting the Grid. In *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2004)*, pages 250–257, A Coruña, Spain, 11-13 February 2004.

[78] Linden deCarmo. *Core Java Media Framework*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[79] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003.

[80] E. Sazonov, P. Klinkhachorn, H. GangaRao, and U. Halabe. Fuzzy logic expert system for automated damage detection from changes in strain energy mode shapes, 2002.

[81] Sven Graupner, Winfried Kalfa, and Carsten Reimann. Modeling and simulation of media-on-demand.

[82] S. Basu, S. Adhikari, R. Kumar, Y. Yan, R. Hochmuth, and B. Blaho. mmgrid: Distributed resource management infrastructure for multimedia applications, 2003.

[83] Angelo Zaia, Dario Bruneo, and Antonio Puliafito. A scalable grid-based multimedia server. In *WETICE*, pages 337–342. IEEE Computer Society, 2004.

[84] Software Engineering Grid Computing Focus Group and Tecnology Labs. Distributed video-on-demand - a grid based vod solution, 2005.

[85] Zhuoying Luo and Huadong Ma. The design method of a video delivery grid. In *APWeb Workshops*, pages 653–658, 2006.

[86] Angelo Zaia, Dario Bruneo, and Antonio Puliafito. Using the grid paradigm for multimedia applications. *Concurrency and Computation: Practice and Experience*, 18(8):899–910, 2006.

[87] Guillaume Pierre. Grid services for adaptive content delivery. In *Proceedings of the Workshop on the Use of P2P, GRID and Agents for the Development of Content Distribution Networks (UPGRADE-CDN)*, June 2006. `http://www.globule.org/publi/GSACD_upgradecdn2006.html`.

[88] Giancarlo Fortino and Wilma Russo. Using p2p, grid and agent technologies for the development of content distribution networks. *Future Gener. Comput. Syst.*, 24(3):180–190, 2008.

[89] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for grid services, 2003.

[90] Domenico Talia. The open grid services architecture: Where the grid meets the web. *IEEE Internet Computing*, 6(6):67–71, 2002.

[91] Valeria Cardellini, Michele Colajanni, and Philip S. Yu. Request redirection algorithms for distributed web systems. *IEEE Trans. Parallel Distrib. Syst.*, 14(4):355–368, 2003.

[92] D. Brookshier, D. Govoni, and N. Krishnan. *JXTA: Java P2P Programming*. Sams, 2002.

[93] Junwei Cao, Stephen A. Jarvis, Subhash Saini, Darren J. Kerbyson, and Graham R. Nudd. Arms: An agent-based resource management system for grid computing. *Sci. Program.*, 10(2):135–148, 2002.

[94] P. Muthuchelv and V. Ramachandran. Abrmas: Agent based resource management with alternate solution. In *GCC '07: Proceedings of the Sixth International Conference on Grid and Cooperative Computing*, pages 147–153, Washington, DC, USA, 2007. IEEE Computer Society.

[95] Kwang-Won Koh, Hiecheol Kim, Kyung-Lang Park, Hwang-Jik Lee, and Shin-Dug Kim. A smart agent-based grid computing platform. In *ICCSA (2)*, pages 20–29, 2004.

[96] A. Lenica, Frederic Ogel, Frédéric Peschanski, and Jean-Pierre Briot. Toward agent-based cooperative resource management in a telecommunication operator grid platform. In *WETICE*, pages 214–219, 2006.

[97] Jia Tang and Minjie Zhang. An agent-based grid computing infrastructure. In *ISPA*, pages 630–644, 2005.

[98] Gang Chen, Zhonghua Yang, Simon See, and Jie Song. Agent-mediated genetic super-scheduling in grid environments. In Liew et al. [150], pages 367–371.

[99] D. Ouelhadj, J. Garibaldi, J. MacLaren, R. Sakellariou, K. Krishnakumar, and Amnon Meisels. A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing.

[100] Qingkui Chen and Lichun Na. Multiagent model for grid computing. In *PRIMA*, pages 571–577, 2006.

[101] Jia Chen, Yue Wu, Ming Li, and Bei Hui. Multi-agent system-based hierarchy grid middleware. In *COMPSAC '07: Proceedings of the 31st Annual International Computer Software and Applications Conference - Vol. 2- (COMPSAC 2007)*, pages 141–146, Washington, DC, USA, 2007. IEEE Computer Society.

[102] Sakamoto Kentaro and Sato Hiroyuki. A resource-oriented grid meta-scheduler based on agents. In *PDCN'07: Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference*, pages 109–114, Anaheim, CA, USA, 2007. ACTA Press.

[103] Montree Bunruangses, Weerawat Poompattanapong, Pratheep Banyatnoparat, and Banjong Piyatamrong. Qos multi-agent applied for grid service management. In *ISICT '04: Proceedings of the 2004 international symposium on Information and communication technologies*, pages 74–79. Trinity College Dublin, 2004.

[104] Jigar Patel, W. T. Luke Teacy, Nicholas R. Jennings, Michael Luck, Stuart Chalmers, Nir Oren, Timothy J. Norman, Alun D. Preece, Peter M. D. Gray, Gareth Shercliff, Patrick J. Stockreisser, Jianhua Shao, W. Alex Gray, N. J. Fiddian, and Simon G. Thompson. Conoise-g: agent-based virtual organisations. In *AAMAS*, pages 1459–1460, 2006.

[105] Xun Gong, Tao Li, Ji Lu, Tiefang Wang, Gang Liang, Jin Yang, and Feixian Sun. Immunity and mobile agent based intrusion detection for grid. In *PRIMA*, pages 187–198, 2006.

[106] H. Tianfield. Towards agent based grid resource management. In *CCGRID '05: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 1*, pages 590–597, Washington, DC, USA, 2005. IEEE Computer Society.

[107] Li Chunlin and Li Layuan. Multi economic agent interaction for optimizing the aggregate utility of grid users in computational grid. *Appl. Intell.*, 25(2):147–158, 2006.

[108] Zhongzhi Shi, He Huang, Yuncheng Jiang, Jiewen Luo, Zheng Zheng, and Fen Lin.

Agrip - agent grid intelligence platform. In *E-Service Intelligence*, pages 627–646. 2007.

[109] Xiong Li, Yujin Wu, Kai Wang, and Zongchang Xu. Concurrent negotiations for agent-based grid computing. In *IEEE ICCI*, pages 31–36, 2006.

[110] Munehiro Fukuda and Duncan Smith. Uwagents: A mobile agent system optimized for grid computing. In *GCA*, pages 107–113, 2006.

[111] Munehiro Fukuda, Koichi Kashiwagi, and Shin ya Kobayashi. Agentteamwork: Coordinating grid-computing jobs with mobile agents. *Appl. Intell.*, 25(2):181–198, 2006.

[112] Badr Jabari, Vincent Englebert, Jean-Pol Vigneron, and El Mostafa Oualim. Towards an agent-based grid platform 1st episode: the meta-grid. In *UPGRADE '07: Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, pages 73–80, New York, NY, USA, 2007. ACM.

[113] Yanxiang He, Weidong Wen, Hui Jin, and Haowen Liu. Agent-based mobile service discovery in grid computing. In *CIT '05: Proceedings of the The Fifth International Conference on Computer and Information Technology*, pages 351–355, Washington, DC, USA, 2005. IEEE Computer Society.

[114] Martha L. Kahn and Cynthia Della Torre Cicalese. The coabs grid. In *WRAC*, pages 125–134, 2002.

[115] Autonomic computing: IBM's perspective on the state of information technology. Internet: http://researchweb.watson.ibm.com/autonomic/.

[116] 1 Zhao, 1 Belloum, 1 De Laat, 1 Adriaans, and 1 Hertzberger. Using jade agent framework to prototype an e-science workflow bus. *ccgrid*, 00:655–660, 2007.

[117] Jiewen Luo and Zhongzhi Shi. Agent-based integration platform on grid infrastructure. In *IEEE ICCI*, pages 170–175, 2006.

[118] Zhongzhi Shi, Haijun Zhang, Yong Cheng, Yuncheng Jiang, Qiujian Sheng, and Zhikung Zhao. Mage: An agent-oriented programming environment. In *IEEE ICCI*, pages 250–257, 2004.

[119] M. Baker, B. Carpenter, G. Fox, and Sung Hoon Koo. mpiJava: An object-oriented Java interface to MPI. 1586, 1999.

[120] Li Chunlin and Li Layuan. Apply agent to build grid service management. *J. Netw. Comput. Appl.*, 26(4):323–340, 2003.

[121] Mohammad Tanvir Huda, Heinz W. Schmidt, and Ian D. Peake. An agent oriented proactive fault-tolerant framework for grid computing. In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, pages 304–311, Washington, DC, USA, 2005. IEEE Computer Society.

[122] Rajkumar Buyya and M. Manzur Murshed. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1175–1220, 2002.

[123] Wolfgang Kreutzer, Jane Hopkins, and Marcel van Mierlo. Simjava - a framework for modeling queueing networks in java. In *Winter Simulation Conference*, pages 483–488, 1997.

[124] R. McNab and F. Howell. Using java for discrete event simulation, 1996.

[125] F. Howell and R. McNab. Simjava: A discrete event simulation package for java with applications in computer systems, 1998.

[126] Costas Simatos. Making simjava count, 2002.

[127] Anthony Sulistio, Chee Shin Yeo, and Rajkumar Buyya. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Softw., Pract. Exper.*, 34(7):653–673, 2004.

[128] Hyo Jung Song, Xianan Liu, Dennis Jakobsen, Ranjita Bhagwan, Xingbin Zhang, Kenjiro Taura, and Andrew A. Chien. The microgrid: a scientific tool for modeling computational grids. In *SC*, 2000.

[129] Andrew A. Chien. The microgrid: Enabling scientific study of dynamic grid behavior. In Minglu Li, Xian-He Sun, Qianni Deng, and Jun Ni, editors, *GCC (1)*, volume 3032 of *Lecture Notes in Computer Science*, page 9. Springer, 2003.

[130] Xin Liu, Huaxia Xia, and Andrew A. Chien. Validating and scaling the microgrid: A scientific instrument for grid dynamics. *J. Grid Comput.*, 2(2):141–161, 2004.

[131] Henri Casanova. Simgrid: A toolkit for the simulation of application scheduling. In *CCGRID* [149], pages 430–441.

[132] Arnaud Legrand, Loris Marchal, and Henri Casanova. Scheduling distributed applications: the simgrid simulation framework. In *CCGRID*, pages 138–145. IEEE Computer Society, 2003.

[133] Henri Casanova. Modeling large-scale platforms for the analysis and the simulation of scheduling strategies. In *IPDPS* [151].

[134] Henri Casanova, Arnaud Legrand, and Martin Quinson. Simgrid: A generic framework for large-scale distributed experiments. *uksim*, 0:126–131, 2008.

[135] Kayo Fujiwara and Henri Casanova. Speed and accuracy of network simulation in the simgrid framework. In *ValueTools '07: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[136] Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1):3, 2006.

[137] Anthony Sulistio, Chee Shin Yeo, and Rajkumar Buyya. Visual modeler for grid modeling and simulation (gridsim) toolkit. In Peter M. A. Sloot, David Abramson, Alexander V. Bogdanov, Jack Dongarra, Albert Y. Zomaya, and Yuri E. Gorbachev, editors, *International Conference on Computational Science*, volume 2659 of *Lecture Notes in Computer Science*, pages 1123–1132. Springer, 2003.

[138] Rajkumar Buyya and M. Manzur Murshed. A deadline and budget constrained cost-time optimisation algorithm for scheduling task farming applications on global grids. *CoRR*, cs.DC/0203020, 2002.

[139] Rajkumar Buyya. Economic-based distributed resource management and scheduling for grid computing. *CoRR*, cs.DC/0204048, 2002.

[140] Anthony Sulistio, Gokul Poduval, Rajkumar Buyya, and Chen-Khong Tham. Constructing a grid simulation with differentiated network service using gridsim. In Hamid R. Arabnia and Rose Joshua, editors, *International Conference on Internet Computing*, pages 437–444. CSREA Press, 2005.

[141] Anthony Sulistio, Gokul Poduval, Rajkumar Buyya, and Chen-Khong Tham. On incorporating differentiated levels of network service into gridsim. *Future Generation Comp. Syst.*, 23(4):606–615, 2007.

[142] W. Bell, D. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini. Optorsim - a grid simulator for studying dynamic data replication strategies, 2003.

[143] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml) 1.0 (fourth edition). World Wide Web Consortium, Recommendation REC-xml-20060816, August 2006.

[144] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. Extensible markup language (xml) 1.1 (second edition). World Wide Web Consortium, Recommendation REC-xml11-20060816, August 2006.

[145] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. Xml schema part 1: Structures second edition. World Wide Web Consortium, Recommendation REC-xmlschema-1-20041028, October 2004.

[146] Paul V. Biron and Ashok Malhotra. Xml schema part 2: Datatypes second edition. World Wide Web Consortium, Recommendation REC-xmlschema-2-20041028, October 2004.

[147] Dave Berton. Advanced video coding on linux. *Linux J.*, 2006(150):5, 2006.

[148] Suramya Tomar. Converting video formats with ffmpeg. *Linux J.*, 2006(146):10, 2006.

[149] *First IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2001), May 15-18, 2001, Brisbane, Australia*. IEEE Computer Society, 2001.

[150] Kim-Meow Liew, Hong Shen, Simon See, Wentong Cai, Pingzhi Fan, and Susumu Horiguchi, editors. *Parallel and Distributed Computing: Applications and Technologies, 5th International Conference, PDCAT 2004, Singapore, December 8-10, 2004, Proceedings*, volume 3320 of *Lecture Notes in Computer Science*. Springer, 2004.

[151] *18th International Parallel and Distributed Processing Symposium (IPDPS 2004), CD-ROM / Abstracts Proceedings, 26-30 April 2004, Santa Fe, New Mexico, USA*. IEEE Computer Society, 2004.