Indice

Introduzione

In questi ultimi anni, il campo dei sistemi multimediali distribuiti ha mostrato una straordinaria crescita grazie al lavoro svolto in molte Università e Centri di Ricerca e alla diffusione di applicazioni multimediali commerciali che rapidamente stanno emergendo e stanno modificando il modo di creare il software.

Una tipica applicazione multimediale trasporta delle informazioni all'utente finale in differenti modi, utilizzando audio e video, immagini, grafici e animazioni, in aggiunta al tradizionale testo. Il denominatore comune di tutte le applicazioni multimediali è l'elevata richiesta di risorse e di garanzie di *Qualità di Servizio*: riuscire a garantire un livello elevato di Qualità di Servizio è dunque uno degli obiettivi primari nello sviluppo delle applicazioni multimediali.

In questo lavoro di tesi l'attenzione sarà posta sullo sviluppo di un componente che andrà ad inserirsi all'interno di un'applicazione multimediale distribuita per la fruizione di presentazioni multimediali. Verranno affrontati due aspetti fondamentali che contribuiscono al miglioramento dell'applicazione: l'incremento della Qualità di Servizio e l'apertura verso altre applicazioni che utilizzano informazioni dello stesso tipo.

Mediante la realizzazione di un sistema di caching distribuito per la descrizione dei contenuti multimediali si cercherà di aumentare il livello di Qualità di Servizio che l'applicazione offre all'utente finale. L'utilizzo del caching permette una drastica riduzione dei tempi di attesa a fronte della richiesta di un dato, specialmente se la frequenza di richiesta del dato è molto elevata. Il sistema di caching sarà caratterizzato da un meccanismo di gestione delle politiche di accesso, che permetterà di ottimizzare

la scrittura dei dati all'interno delle cache, in base alla particolare politica adottata.

Il secondo aspetto che verrà preso in considerazione nello sviluppo del componente è l'apertura dell'applicazione, ossia lo scambio di informazioni con altre applicazioni dello stesso tipo. Per far fronte a questa esigenza, in questi ultimi anni, sono stati realizzati degli standard per la descrizione di informazioni tramite un linguaggio universale. Si cercherà, dunque, di creare uno schema per la rappresentazione di informazioni di tipo multimediale che sia il più possibile aderente ai moderni standard in merito.

La tesi è stata organizzata nel seguente modo.

Nel Capitolo 1 vengono presentati ed analizzati alcuni concetti base sulla Qualità di Servizio. Dopo una breve introduzione sulle applicazioni multimediali e sulle problematiche legate al loro sviluppo, si passa alla discussione delle caratteristiche e dei principi che regolano la Qualità di Servizio all'interno di un'applicazione multimediale.

L'argomento del Capitolo 2 è il caching: dopo aver dato alcune definizioni sul caching e sul funzionamento di un sistema di caching, viene messa in evidenza l'importanza che riveste il caching come strumento per garantire Qualità di Servizio. Successivamente, vengono analizzate in dettaglio le tipologie dei sistemi di caching: il caching di tipo gerarchico, di tipo distribuito e di tipo ibrido, evidenziando per ciascun tipo i pro e contro.

Per facilitare al lettore la comprensione dei capitoli successivi, nel Capitolo 3, vengono presentate le due infrastrutture in cui sarà sviluppato il modulo. Si parte dalla descrizione dei paradigmi basati sulla mobilità di codice, per poi passare alla descrizione dell'infrastruttura per agenti mobili *SOMA* e, infine, alla presentazione di *MUM*, l'applicazione multimediale distribuita per la fruizione di presentazioni multimediali all'interno della quale andrà ad inserirsi il nuovo componente.

Nel Capitolo 4, si cercherà di identificare uno schema per la rappresentazione delle informazioni di tipo multimediale che sia il più possibile aderente alle specifiche degli standard correnti. Inizialmente vengono perciò

analizzati due standard internazionali per la descrizione di presentazioni multimediali, mettendone in evidenza le caratteristiche fondamentali e le mancanze. Si passerà successivamente alla realizzazione di uno schema per la descrizione che racchiuda al suo interno le caratteristiche migliori dei due standard precedentemente analizzati.

Nel Capitolo 5, si entra nello sviluppo vero e proprio del modulo. In questo capitolo, infatti, viene condotta l'analisi del sistema di gestione dei contenuti multimediali. Verranno dapprima messi in evidenza i requisiti del componente. Si passerà successivamente alla loro analisi e si riuscirà a ricavare un primo schema che rappresenta le classi fondamentali del componente.

Nel Capitolo 6, si passa alla fase di progettazione del sistema di gestione dei contenuti multimediali. Partendo dalla rappresentazione ottenuta dall'analisi svolta nel precedente capitolo, il componente verrà suddiviso in più parti, e per ognuna verranno descritte le classi che la compongono. Si passerà, poi, alla descrizione delle interazioni fra le classi che compongono ogni singola parte del componente.

Nella prima parte del Capitolo 7 vengono delineate le principali scelte implementative e sono messe in evidenza le motivazioni che hanno portato a queste scelte. Nella seconda parte del capitolo, l'attenzione è infine dedicata al test del sistema: si parte dalla verifica del corretto funzionamento del componente e, infine, si passa all'analisi delle prestazioni che l'applicazione raggiunge utilizzando il nuovo componente.

Capitolo 1

Qualità di Servizio nei sistemi multimediali distribuiti

In questo primo capitolo, verranno dapprima forniti i concetti essenziali propri delle applicazioni multimediali distribuite: il concetto di presentazione multimediale, la tipologia delle applicazioni multimediali e le problematiche legate alla loro realizzazione.

La definizione dei requisiti delle applicazioni multimediali distribuite e di una buona base per il loro supporto può essere racchiusa in un unico concetto che prende il nome di *Qualità di Servizio*. Essa rappresenta l'insieme delle caratteristiche quantitative e qualitative che un sistema multimediale distribuito deve avere per fornire in maniera ottimale le funzionalità richieste.

Il concetto di Qualità di Servizio rappresenta un aspetto molto importante di tutte le moderne applicazioni multimediali distribuite: verranno, perciò, nel seguito del capitolo, analizzati gli aspetti salienti che caratterizzano il concetto di Qualità di Servizio.

1.1 Applicazioni multimediali

Lo scopo principale di un'applicazione multimediale è quello di manipolare informazioni di varia natura e di fornirle all'utente. Le informazioni vanno dal tradizionale e semplice testo, a più moderne e comples-

se tipologie, come audio, video, immagini, grafici e animazioni. Esempi di applicazioni multimediali possono essere le applicazioni per il *video on demand*, i sistemi per le video-conferenze o per la *tele-educazione*.

L'informazione viene molto spesso indicata con il termine *presentazione multimediale*. Le presentazioni multimediali maggiormente utilizzate nelle applicazioni multimediali consistono di una sequenza continua di campioni finiti aventi una stretta dipendenza temporale fra loro: tali presentazioni vengono definite *continue* e il termine *stream* è utilizzato per indicare una sequenza di campioni costituenti una presentazione continua.

Largamente parlando, le applicazioni multimediali possono essere classificate nel seguente modo:

- **applicazioni per la presentazione**: sono caratterizzate dall'utilizzo di informazioni memorizzate in uno o più dispositivi con capacità elevata. Gli utenti possono ottenere le informazioni che desiderano in tempo reale attraverso una rete a larga banda direttamente sul display dei propri dispositivi. Ad esempio, in un sistema di video on demand, essi hanno la possibilità di selezionare fra i differenti video presenti e di richiede la presentazione del video che hanno richiesto.

  La difficoltà nello sviluppo di questo tipo di applicazioni è legata all'alta mole di dati e ai severi vincoli temporali che caratterizzano le presentazioni di tipo continuo: lo sviluppo di queste applicazioni dipende dunque dai progressi nei sistemi di comunicazione, nei dispositivi di memorizzazione e nella capacità di elaborazione.

- **applicazioni per la conversazione**: sono utilizzate per veicolare comunicazioni in tempo reale tra persone situate in luoghi differenti, ad esempio, per la video-conferenza. Queste applicazioni sono molto più sensibili al ritardo delle applicazioni per le presentazioni, poiché l'informazione è creata in tempo reale e dovrebbe essere ricevuta in tempo reale. I dati prodotti sono validi per un limitato periodo e una volta che questo lasso di tempo è passato i dati diventano inutilizzabili.

- **applicazioni ibride**: sono le applicazioni multimediali che non possono essere inserite in nessuna delle due precedenti categorie. Esse vengono utilizzate sia per la presentazione che per la conversazione: ad esempio, un sistema per la video conferenza che permetta non soltanto la comunicazione tra i partecipanti, ma anche la visione di presentazioni memorizzate in un dispositivo.

1.1.1 Problematiche e requisiti di un'applicazione multimediale

Il denominatore comune di tutte le applicazioni multimediali è l'elevata richiesta di risorse e di garanzie di Qualità di Servizio, richiesta che aumenta all'aumentare della criticità dell'applicazione.

Nello sviluppo di un'applicazione multimediale i progettisti devono tenere in considerazione determinati limiti entro i quali muoversi: i limiti possono essere di natura fisica (ad esempio, la carenza di risorse) oppure possono dipendere dal software che l'applicazione utilizza (ad esempio, le limitazioni di un particolare protocollo di rete utilizzato).

Per comprendere le caratteristiche di un'applicazione multimediale, verranno ora prese in considerazione le principali problematiche legate al loro sviluppo e alla loro diffusione, le quali obbligano i progettisti a seguire determinati principi per poter garantire Qualità di Servizio nelle loro applicazioni.

Carenza di risorse fisiche

Uno dei fattori che maggiormente limita la diffusione delle applicazioni multimediali è l'alta richiesta di risorse fisiche: in particolare, fra tutte le risorse fisiche, quelle più interessate sono:

- **CPU**: è la risorsa più contesa in quanto è coinvolta in diverse fasi della fruizione della presentazione multimediale. Nel caso in cui i dati devono essere inviati essa ha il compito di codificarli e di impacchettarli; se i dati devono essere ricevuti essa deve spacchettarli,

decodificarli e poi effettuare il *rendering*. Queste sono solo alcune delle più importanti operazioni che una CPU deve effettuare sui dati multimediali;

- **Banda trasmissiva**: l'alta mole di dati che caratterizza una presentazione multimediale fa sì che un'applicazione multimediale abbia bisogno di un'elevata banda trasmissiva per garantire la fruizione corretta della presentazione multimediale. Sono stati sviluppati, per questo motivo, degli algoritmi di compressione dei dati, il cui utilizzo riduce notevolmente la richiesta di banda trasmissiva ma va ad influire sulle prestazioni legate alla CPU, in quanto essa è costretta ad un carico maggiore di lavoro;
- **Memoria**: la trasmissione e la visualizzazione delle presentazioni multimediali richiedono un'elevata quantità di memoria. Essa viene utilizzata per effettuare operazioni di *buffering* dei dati, in maniera da continuare la visualizzazione in caso di momentanei ritardi nella trasmissione, o per memorizzare i dati durante le fasi di codifica/decodifica.

Mancanza di espressività nelle piattaforme di supporto

Un secondo problema nello sviluppo delle applicazioni multimediali è la carenza di un supporto che garantisca la loro corretta realizzazione.

Sia a livello delle infrastrutture di rete sia a livello dei sistemi operativi, viene infatti prevalentemente fornito il supporto per lo sviluppo di applicazioni con architetture di tipo *best-effort*, che non offrono garanzie di Qualità di Servizio.

Ci sono due modi per far fronte a questa carenza: agire a livello del sistema operativo, intervenendo sul kernel e inserendo opportune funzionalità, oppure sviluppare uno strato intermedio che realizzi il supporto necessario per la gestione delle risorse. Questo strato intermedio, che viene comunemente indicato con il termine *middleware*, è posto tra il sistema operativo e l'applicazione ed ha il compito di fornire a quest'ultima le funzionalità che il sistema operativo non è in grado di offrirle.

Eterogeneità delle piattaforme e dei dispositivi

Un altro problema è dovuto all'eterogeneità delle piattaforme e dei dispositivi con cui si accede al contenuto del sistema.

L'adozione di un middleware riesce a facilitare il superamento di questo problema, nascondendo agli sviluppatori dell'applicazione multimediale i dettagli della piattaforma sottostante.

Quando, però, il dispositivo dispone di una capacità limitata per riuscire a ospitare il middleware, occorre studiare delle soluzioni particolari per ogni singolo dispositivo.

1.2 Qualità di Servizio

1.2.1 Definizioni generali

Il concetto di Qualità di Servizio (*Quality of Service, QoS*) può essere così definito: *La Qualità di Servizio è l'effetto complessivo della prestazione del servizio, che determina il grado di soddisfazione di un utente del servizio* [ITU89].

Una definizione di Qualità di Servizio, più specifica della precedente, può essere: *La Qualità di Servizio è descritta in termini di un insieme di caratteristiche, rilevate dall'utente, che riguardano le prestazioni di un sistema. Essa è espressa in un linguaggio comprensibile all'utente e ed è rappresentata da un numero di parametri, ognuno dei quali ha un valore soggettivo o oggettivo* [RAC91].

La Qualità di Servizio include parametri che esprimono il comportamento delle prestazioni del sistema e parametri che esprimono le caratteristiche di altri servizi, quali, ad esempio, la protezione o la priorità. Esistono due tipologie di parametri:

- I **parametri oggettivi**, che possono essere direttamente osservati e misurati nei punti in cui l'utente accede al servizio (ad esempio, ritardo, banda trasmissiva, etc.);

- I **parametri soggettivi**, che dipendono dalle apparecchiature dell'u-

tente (ad esempio, la qualità degli altoparlanti) e dalle sue impressioni, e che non possono essere misurati direttamente.

I giudizi sulla Qualità di Servizio provengono da due parti (come è mostrato nella figura 1.1): il fornitore (*provider*) dei servizi e il cliente (*customer*). Nella vita reale, le due opinioni spesso sono differenti, dato che ciascuna parte si basa sui propri termini di giudizio. I fornitori creano un servizio implementando specifiche funzioni e procedure, utilizzando in molti casi, servizi addizionali con proprie prestazioni individuali.

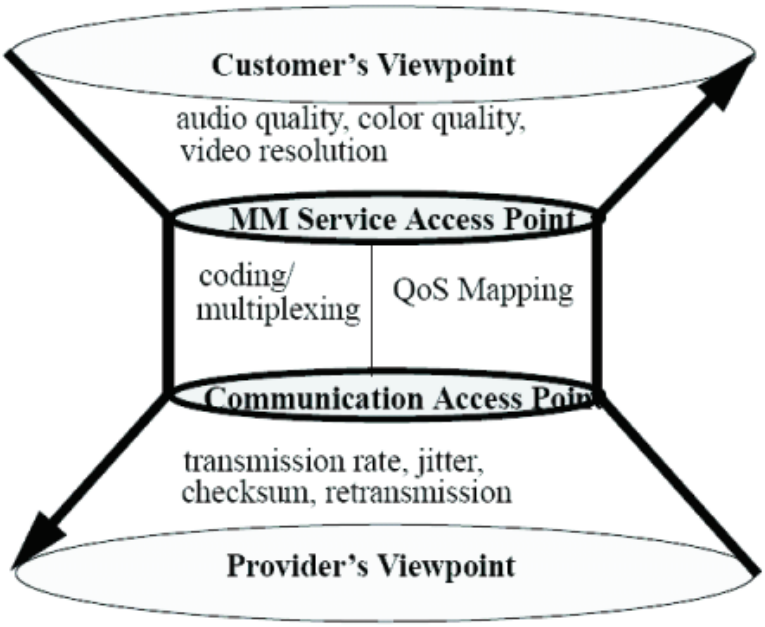

Figura 1.1: Differenti viste della Qualità di Servizio.

Il servizio è fornito nel *Multimedia (MM) Service Access Point*: questa interfaccia rappresenta un filtro che permette di separare i parametri che misurano le prestazioni visibili dall'utente da ciò che invece riguarda le caratteristiche interne del fornitore.

Ciò che il fornitore sa sui requisiti dell'utente è ristretto alle opinioni dell'utente. L'utente, infatti, richiede parametri individuali di Qualità di Servizio, ad esempio la risoluzione del video o la qualità del colore, per le sue applicazioni.

1.2.2 Qualità di Servizio nelle applicazioni multimediali distribuite

Per soddisfare i requisiti delle applicazioni multimediali, tutti i componenti del sistema devono riuscire ad ottenere un certo livello di Qualità di Servizio. Ciascun componente, sia esso hardware o software, realizza una particolare funzione, che può essere l'elaborazione degli stream di dati oppure la loro trasmissione.

I componenti fondamentali sono i sistemi di comunicazione (come, ad esempio, l'insieme dei protocolli oppure le reti) e i sistemi finali (cioè quelli utilizzati dall'utente).

In un sistema per il video on demand, per fornire una presentazione con una desiderata Qualità di Servizio, il server che contiene la presentazione, la rete, i protocolli di trasporto, i *codec* e l'applicazione utente, devono garantire un certo livello di Qualità di Servizio (Figura 1.2). Ciascuno di questi componenti è responsabile di sé stesso, attraverso dei meccanismi di prenotazione delle risorse, per raggiungere determinati livelli di ritardo, o altro, in maniera da fornire lo stream in accordo con i requisiti dell'utente.

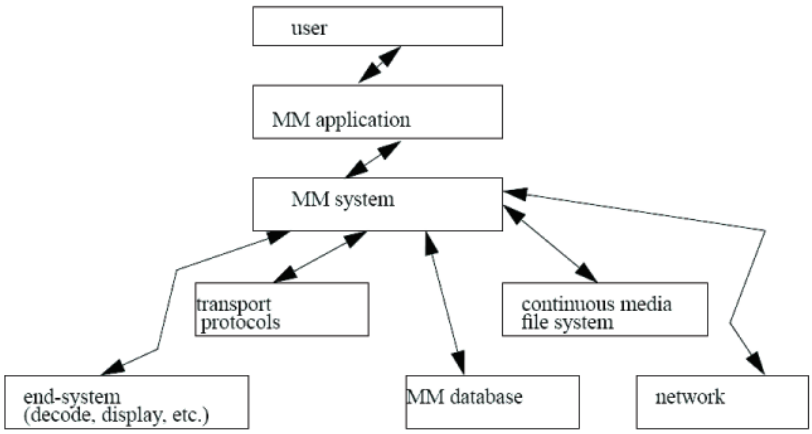

Figura 1.2: Componenti di un sistema multimediale.

1.2.3 Gestione della Qualità di Servizio

Un tipico scenario per l'utilizzo della Qualità di Servizio in un applicazione multimediale distribuita coinvolge i seguenti passi:

1. L'utente richiede un servizio multimediale, ad esempio una presentazione, con una desiderata Qualità di Servizio. In questo modo, l'utente può specificare il valore di un insieme di parametri, ad esempio la risoluzione di un video.

2. In base alla richiesta dell'utente il sistema si configura, prenotando le risorse necessarie a garantire la Qualità di Servizio richiesta.

3. Per assicurare che tutti i componenti possano garantire la Qualità di Servizio richiesta, avviene una negoziazione della Qualità di Servizio.

4. Una volta che il sistema è stato configurato correttamente, l'utente può finalmente ricevere quello che ha richiesto.

In ogni caso, durante la fase di fruizione del materiale multimediale da parte dell'utente, il sistema può monitorare la Qualità di Servizio correntemente fornita e prendere le appropriate decisioni quando non sia più possibile garantire quella richiesta dall'utente (ad esempio, a causa di una congestione della rete). In base alle differenti applicazioni multimediali, tre azioni possono essere intraprese: la sessione applicativa è abortita, viene rinegoziata la Qualità di Servizio con l'utente, oppure l'applicazione cerca di adattarsi come meglio può.

Per assicurare che i requisiti richiesti dall'utente siano soddisfatti è essenziale la gestione della Qualità di Servizio. Essa consiste in un insieme di attività che permettono al fornitore dei servizi di garantire la Qualità di Servizio desiderata. Le sue funzioni possono essere così descritte:

- **Specifica**: l'utente deve essere in grado di specificare i requisiti desiderati tramite un linguaggio a lui familiare. In generale, l'utente specifica i valori per una lista di parametri rilevanti: i valori richiesti

possono essere espressi in termini di limiti superiori e inferiori, possono essere dei valori esatti o possono essere di altro tipo, ad esempio una lista di valori accettabili. Un altro approccio per la specifica della Qualità di Servizio è quello di usare delle classi, dividendo i requisiti dell'applicazione in: realizzabili, irrealizzabili, dipendenti dal tempo e indipendenti dal tempo. Questo approccio, però, è incompleto e poco flessibile, e perciò tende a non essere utilizzato.

- **Mapping**: si tratta della traslazione delle rappresentazioni della Qualità di Servizio nei differenti livelli del sistema: la traslazione dei requisiti dell'utente nei parametri rilevanti per il fornitore del servizio è un aspetto di fondamentale importanza per riuscire a supportare ogni possibile richiesta.

- **Negoziazione**: il ruolo di questa fase è quello di trovare un accordo tra i valori richiesti dall'utente per la Qualità di Servizio e il sistema. Quando l'utente richiede una certa Qualità di Servizio, il sistema verifica se può garantirla: in caso affermativo, alloca le risorse necessarie e fa partire la comunicazione; in caso negativo, fa partire una fase di negoziazione per riuscire ad accordarsi con l'utente. Se anche questa fase non ha esito positivo, la richiesta dell'utente viene respinta.

- **Monitoraggio**: questa fase permette una continua misurazione della Qualità di Servizio attualmente fornita. Essa fondamentalmente si suddivide in due parti:

  - Individuare e notificare ogni violazione della Qualità di Servizio e possibilmente anche la causa.
  - Memorizzare le informazioni acquisite.

  Non bisogna tuttavia abusare di questa fase perché richiede un utilizzo di risorse che vengono sottratte alle risorse complessive del sistema.

- **Controllo**: permette di prendere dei provvedimenti quando l'utente viola la Qualità di Servizio inizialmente accordata col sistema. Essa permette di proteggere il sistema da utenti scorretti (siano essi consapevoli o no) che possano influire sulle garanzie dell'esistente Qualità di Servizio.

- **Adattamento**: il compito di questa fase è quello di mantenere, nel modo migliore possibile, la Qualità di Servizio accordata durante la fase di negoziazione. Un protocollo di adattamento viene eseguito ogni qual volta che viene rilevata una violazione alla Qualità di Servizio. Quando il mantenimento della Qualità di Servizio accordata non è possibile, essa deve essere in grado di contenere i danni della degradazione delle prestazioni.

- **Rinegoziazione**: questa fase può essere iniziata o da parte dell'utente o da parte del sistema. L'utente, ad esempio, potrebbe voler richiede una qualità migliore, mentre il sistema potrebbe iniziare la fase di rinegoziazione a causa di una riduzione delle risorse e quindi ridurre la qualità fornita per evitare l'interruzione del servizio.

- **Stima dei costi**: implica il calcolo del costo relativo ad un servizio richiesto dall'utente. Poiché le applicazioni multimediali distribuite forniscono una vasta gamma di classi di Qualità di Servizio, la stima dei costi è un'attività molto complessa, ma di fondamentale importanza: senza questa fase, infatti, il sistema potrebbe rischiare di bloccarsi, non riuscendo ad eseguire più nessuna operazione, a causa dei provvedimenti non presi per limitare le richieste da parte degli utenti, nel caso in cui queste stiano sottraendo tutte le risorse a disposizione del sistema.

- **Terminazione**: Quando il servizio è terminato, le risorse riservate possono essere rilasciate. Ogni componente viene informato della terminazione dell'applicazione e del rilascio delle risorse: il sistema, quindi, può procedere all'aggiornamento di tutte le informazioni.

1.3 Conclusioni

In questo primo capitolo sono stati messi in evidenza i meccanismi che permettono ad un'applicazione multimediale distribuita di garantire la Qualità di Servizio.

Nel prossimo capitolo, l'attenzione verrà posta su una modalità che può essere utilizzata per garantire i principi di Qualità di Servizio presentati in questo capitolo: il caching.

Verranno analizzati gli aspetti del caching e le diverse tipologie dei sistemi di caching, per riuscire ad identificare un buon modello per il modulo che è oggetto di sviluppo del lavoro di tesi.

Capitolo 2

Soluzioni di caching per garantire Qualità di Servizio

Come è emerso dall'analisi fatta nel precedente capitolo, per riuscire a garantire Qualità di Servizio in un'applicazione multimediale distribuita occorre rispettare dei principi e mettere in pratica delle determinate politiche. In questo capitolo verrà preso in considerazione l'utilizzo del *caching* all'interno di un'applicazione distribuita come strumento atto a garantire Qualità di Servizio.

Viene dapprima fornita una definizione di caching e una descrizione dei meccanismi utilizzati per la sua realizzazione, per poi successivamente passare all'analisi dei vari livelli e delle differenti tipologie di caching, evidenziando di ciascuna i pro e i contro. Si riuscirà, in questo modo, a determinare un modello che sarà successivamente utilizzato nello sviluppo del progetto di tesi.

2.1 Livelli di caching

Negli anni '60, per colmare il divario in termini di velocità tra la CPU e la memoria principale, Maurice V. Wilkes introdusse il seguente concetto di cache: *La cache è una memoria piccola ma veloce, con lo scopo di mantenere alcuni dati per un efficiente prossimo riuso* [WIL65].

Col passare del tempo, tale concetto è stato adottato ad ogni livello

della gerarchia dei sistemi informatici: nonostante ogni livello presenti problematiche ed esigenze particolari, i principi che derivano da tale definizione restano ugualmente validi.

Negli attuali sistemi informatici, viene fatto largo uso del caching. Si possono distinguere vari livelli di caching:

- CPU: Sistemi monoprocessore;
- SMP: Sistemi multiprocessore con memoria condivisa;
- DSM: Sistemi con memoria condivisa distribuita;
- DFS: File System Distribuiti;
- WWW: World Wide Web;
- IA: Applicazioni Internet.

Non si vuole, in questa sede, entrare nei particolari dei primi quattro livelli, in quanto ciò sarebbe superfluo ai fini del progetto di tesi. Verranno perciò presi in considerazione solamente gli ultimi due livelli, a causa della loro similarità con le infrastrutture su cui andrà a svilupparsi il progetto di tesi.

2.1.1 Livello WWW (World Wide Web)

Con la crescita esponenziale, che in questi ultimi ha caratterizzato il World Wide Web, il traffico sulla rete Internet rischia di diventare un problema veramente critico. Diverse direzioni vengono percorse per far fronte in maniera efficiente ed economica a questo problema.

La soluzione più ovvia sarebbe quella di aumentare la larghezza di banda della rete, ma tale soluzione non è per il momento né economicamente né tecnicamente affrontabile. Anche se sono stati ottenuti dei miglioramenti a livello di protocolli, questo non sono del tutto sufficienti.

Per questo motivo, l'uso massiccio ed ottimizzato del caching può determinare forti riduzioni di traffico sulla rete, miglioramenti a livello di carico sui server e riduzione dei tempi di attesa per gli utenti del Web.

All'interno di Internet, si fa utilizzo del caching in tre differenti entità: nei *Web Browser*, nei nodi della rete, nei *Web Server*. Il caching ha una caratteristica comune in tutte e tre queste entità: si tratta della modalità di funzionamento. Quando, in ciascuna delle tre entità, perviene la richiesta di un documento, se è presente nella cache una copia, il richiedente riceve una copia di tale documento; se il documento non è presente in cache, la richiesta viene inoltrata verso un'altra entità (che potrebbe essere un'altra cache o il server di partenza).

2.1.2 Livello IA (Applicazioni Internet)

Questo livello racchiude applicazioni distribuite più generiche rispetto ai sistemi Web e comprende una molteplicità di casi, con caratteristiche molto differenti. Anche se le implementazioni concrete possono differire enormemente, la modalità di gestione della cache risulta un aspetto che le accomuna.

2.2 Caratteristiche dei sistemi di caching

La funzionalità base di una cache è molto semplice; tuttavia, una sua efficace realizzazione può dipendere da più fattori. Il progetto di un sistema di caching efficiente, infatti, risulta essere tutt'altro che banale e implica la necessità di dover fare numerose scelte: quante unità di cache utilizzare, dove sistemarle geograficamente, come strutturarle, quali dati ogni cache dovrà mantenere, per quanto tempo, e così via.

Ogni problema che si presenta può essere affrontato seguendo diverse politiche. Per poter essere analizzate in maniera più organica possono essere distinte le seguenti classi (non si vuole entrare tuttavia nei particolari di ogni classe perché ciò risulterebbe superfluo ai fini del progetto di tesi. Una trattazione più esaustiva si trova in [ROU98]):

- **prefetching**: si propone di anticipare e prevedere le esigenze future degli utenti, attraverso l'analisi di particolari elementi, come, ad

esempio, le richieste passate degli utenti oppure i documenti correlati con quelli richiesti. In questo modo, possono essere memorizzate nella cache le copie dei documenti che si ritiene saranno richiesti in futuro, per averli già a disposizione;

- **routing**: poiché i sistemi di caching sono organizzati su più nodi, esiste il problema del percorso che eventuali richieste devono seguire. Il problema è simile al *network routing*, anche se non può essere affrontato con gli stessi strumenti, a causa delle differenze strutturali tra i due sistemi;

- **placement**: in un sistema formato da più cache che cooperano tra loro, è di fondamentale importanza bilanciare il carico dei dati all'interno delle singole cache, per non sovraccaricare l'intero sistema. Questa politica si occupa proprio del modo in cui tutto ciò viene garantito. Essa riveste particolare importanza nel presente progetto di tesi in quanto, come si vedrà nei successivi capitoli, esso è focalizzato sulla realizzazione di politiche di placement all'interno di un'applicazione multimediale distribuita.

- **coherency**: ogni singola cache deve mantenere una copia aggiornata dei documenti che contiene. Può infatti succedere che il documento originale venga modificato, mentre la copia presente nella cache non venga modificata: questa politica riguarda i modi in cui è possibile garantire la massima coerenza tra i documenti e le copie presenti all'interno de sistema;

- **removal**: quando in una cache si esaurisce lo spazio per memorizzare i dati, si dovrà provvedere ad eliminarne alcuni. Questa politica si occupa delle modalità in base alle quali eliminare dei dati, ad esempio, rimuovendo i documenti più vecchi o quelli che occupano più risorse, a altro. Questa politica influisce più di ogni altra sulle prestazioni della cache in quanto determina quali oggetti devono essere mantenuti in cache e quali invece devono essere rimpiazzati.

2.3 Il caching come strumento per garantire Qualità di Servizio

Andremo, in questo paragrafo, ad analizzare per quali motivi la realizzazione di un sistema di caching all'interno di un'applicazione multimediale distribuita contribuisca al raggiungimento dei principi di Qualità di Servizio introdotti nel precedente capitolo.

L'utilizzo di un sistema di caching influisce sia sui parametri oggettivi che misurano la Qualità di Servizio sia sulle modalità con cui il sistema cerca di garantire un determinata Qualità di Servizio.

I parametri oggettivi maggiormente influenzati risultano essere il tempo di trasmissione, l'utilizzo di banda trasmissiva e la quantità di spazio utilizzato nei dispositivi di memorizzazione. I primi due ottengono dei miglioramenti dall'utilizzo del caching, mentre il terzo subisce un peggioramento.

- Il **tempo di trasmissione** dei documenti diminuisce notevolmente in quanto la memorizzazione di copie nei nodi in cui sono presenti delle cache accorcia il percorso che i documenti avrebbero dovuto affrontare per giungere al client se si fossero trovati esclusivamente nel server centrale. Più è grande la popolarità di un documento, più è probabile che una copia di quel documento si trovi in una cache più vicina a chi ne fa richiesta, riducendo così il tempo di trasmissione di quel documento;

- L'utilizzo di **banda trasmissiva** si riduce per il fatto che i documenti, o le copie di essi, trovandosi in nodi più vicini a chi ne fa richiesta, compiono dei percorsi più brevi. Tuttavia bisogna mettere in evidenza che la cooperazione tra le entità che effettuano il caching introduce nel sistema delle informazioni aggiuntive, informazioni che, però, essendo di piccole dimensioni rispetto ai documenti, non influiscono in maniera rilevante sul consumo di banda trasmissiva all'interno del sistema;

- Lo **spazio utilizzato nei dispositivi** per la memorizzazione dei documenti o delle copie di essi, al contrario dei due precedenti parametri, aumenta. Ogni entità incaricata di effettuare il caching ha bisogno di una determinata quantità di memoria per memorizzare i propri dati: tuttavia, il peggioramento di questo parametro può essere considerato irrilevante sia perché il miglioramento degli altri è molto più importante ai fini del raggiungimento di una determinata Qualità di Servizio sia perché le moderne tecnologie assicurano dispositivi di memorizzazione sempre più capienti.

Le modalità utilizzate per garantire una determinata Qualità di Servizio che sono influenzate dall'utilizzo di un sistema di caching sono la negoziazione, l'adattamento e la rinegoziazione.

Durante la fase di negoziazione, il sistema cerca di accordarsi con il cliente, sulla Qualità di Servizio che deve essere fornita: nel far ciò, il sistema deve tener conto di eventuali copie del documento richiesto che siano migliori per l'utente, ad esempio, in termini di vicinanza o di banda trasmissiva disponibile.

Nella fase di adattamento, cercando di mantenere la Qualità di Servizio accordata, può far fronte ad eventuali malfunzionamenti della rete, come ad esempio la perdita del nodo dal quale si stava richiedendo il documento, fornendo come alternativa degli altri nodi sui quali è presente il documento.

Anche nella fase di rinegoziazione, la presenza di copie di un documento in nodi più o meno vicini all'entità che richiede il documento, fa si che il sistema possa offrire differenti scenari di Qualità di Servizio.

Risulta perciò evidente che un utilizzo massiccio dei sistemi di caching riesca a migliorare le prestazioni di un'applicazione multimediale distribuita, che può così offrire livelli più alti di Qualità di Servizio. Nel progettare un sistema di caching occorre naturalmente analizzare le caratteristiche e le esigenze dell'applicazione, per poter identificare l'architettura più consona all'applicazione.

2.4 Architettura dei sistemi di caching

Molti dei moderni sistemi di caching sono formati da più entità che utilizzano una o più cache. Queste entità gestiscono le proprie cache cooperando fra di loro, migliorando in questo modo le prestazioni delle cache isolate.

Per far in modo che più entità cooperino su larga scala, migliorando effettivamente in tal modo le prestazioni dell'intero sistema, esse vengono organizzate mediante particolari architetture, che definiscono la struttura e i protocolli per garantire la cooperazione.

Negli attuali sistemi di caching, si possono distinguere due tipologie di struttura: centralizzata e distribuita. La prima è caratterizzata dalla presenza di un'unica cache in un server centrale, che ha il compito di soddisfare tutte le richieste; la seconda è invece caratterizzata dalla presenza di più cache, posizionate all'interno dei vari nodi del sistema.

Nei sistemi che utilizzano una struttura distribuita, esistono fondamentalmente due tipologie di protocolli per la cooperazione fra le cache: gerarchico e distribuito.

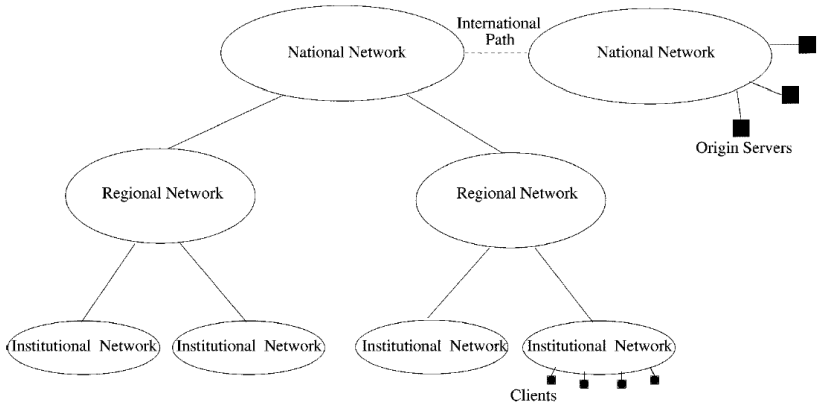

Figura 2.1: Topologia della rete Internet.

Prendendo in considerazione la struttura della rete Internet, la quale non è altro che il più grande sistema distribuito, possono essere messe in evidenza le caratteristiche dei due protocolli: in figura 2.1 è mostrata la classica topologia della rete mondiale. Tutte le reti sono interconnesse

fra loro formando una struttura ad albero: tutti i client sono collegati alle reti istituzionali; le reti istituzionali sono collegate alle reti regionali; le reti regionali sono collegate alle reti nazionali; infine, le reti nazionali sono interconnesse tra loro tramite collegamenti transoceanici o via satellite.

La struttura appena presentata può essere schematizzata, come in figura 2.2, mettendo in evidenza le cache presenti all'interno di ogni rete e di ogni nodo.

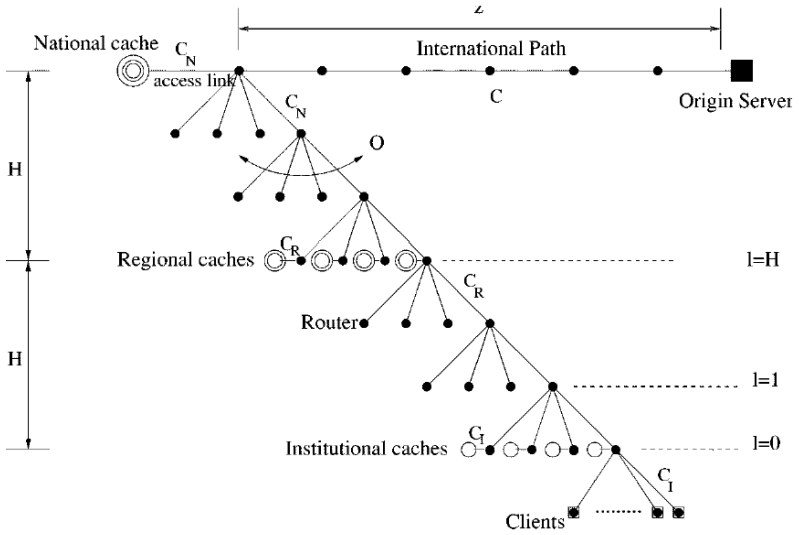

Figura 2.2: Modello ad albero.

Questa struttura resta invariata sia che si utilizzi un protocollo di caching gerarchico sia che se ne utilizzi uno distribuito: a cambiare è solamente il modo in cui le cache cooperano fra loro.

Esiste, tuttavia, una terza tipologia di protocollo, caratterizzata dall'utilizzo di funzionalità presenti sia nel caching gerarchico sia in quello distribuito.

2.4.1 Caching gerarchico

La cooperazione tra le differenti cache si basa fondamentalmente sul fatto che ogni cache può comunicare solamente con quella del livello superiore (ad eccezione della cache presente nella radice della gerarchia) e

con quelle degli eventuali figli. Il protocollo stabilisce come devono essere fornite le funzionalità di inserimento, aggiornamento, eliminazione e richiesta dei documenti.

Consideriamo, ad esempio, il caso di richiesta di un documento da parte di un'entità nel livello client: nel caso in cui la richiesta non potesse essere soddisfatta dalla cache del client, essa verrebbe inoltrata alla cache della rete istituzionale. Se anche nella cache della rete istituzionale la richiesta non potesse essere soddisfatta, essa verrebbe inoltrata alla cache del livello superiore, in questo caso alla cache della rete regionale. Si procederebbe così, giungendo fino alla cache del livello più elevato della gerarchia, nel caso in cui la richiesta non potesse essere soddisfatta da nessuna cache intermedia. Se il documento non dovesse essere presente in nessuna cache, la richiesta verrebbe inviata direttamente al server di origine. Una volta che il documento richiesto venisse trovato, esso percorrerebbe la gerarchia al contrario e, in base al protocollo, potrebbe essere memorizzato da ogni cache che esso attraversa, in maniera da averlo disponibile a fronte di una successiva richiesta.

Le funzionalità di inserimento, aggiornamento ed eliminazione devono essere realizzate in maniera tale da garantire la consistenza dei dati presenti all'interno di ogni cache: ad esempio, quando si inserisce un nuovo dato all'interno della cache, bisogna far in modo che almeno la cache presente nella radice della gerarchia (e possibilmente una o più cache intermedie) sia a conoscenza del dato, per garantire la conoscenza dello stesso all'interno di tutto il sistema.

Nel caso in cui un documento venisse modificato o eliminato, tutte le cache in cui fosse presente quel documento dovrebbero venirne a conoscenza, in maniera tale da mantenere nella cache una copia aggiornata del documento e garantire, in questo modo, la consistenza dei dati all'interno del sistema.

Ci sono molteplici problemi associati a questo tipo di architettura:

- ogni gerarchia introduce dei ritardi addizionali, che fanno aumentare il ritardo complessivo delle comunicazioni, riducendo le prestazioni del sistema;

- le cache dei livelli più elevati possono diventare dei colli di bottiglia, a causa dei ritardi introdotti dalle code di attesa;

- più copie dello stesso documento sono memorizzate in differenti livelli, con la possibilità che sussistano delle inconsistenze, nel caso in cui il protocollo non fosse stato realizzato in maniera corretta.

2.4.2 Caching distribuito

Utilizzando un protocollo di tipo distribuito, ogni cache presente all'interno del sistema coopera con le altre cache: deve esistere, perciò, un meccanismo che permetta alle varie cache di scambiarsi le informazioni necessarie per fornire le funzionalità di inserimento, aggiornamento, eliminazione e richiesta dei documenti. Alcuni di questi meccanismi sono:

- le cache possono interrogarsi a vicenda quando non riescono a realizzare una determinata funzionalità attraverso un particolare linguaggio (un esempio è *Inter Cache Protocol* [WES97]). Tuttavia, utilizzando questo sistema, c'è il rischio di aumentare in maniera significativa l'utilizzo della banda trasmissiva e di introdurre dei ritardi indesiderati all'interno del sistema;

- le cache possono mantenere dei sommari di ciò che è contenuto nelle altre cache, in maniera da evitare di effettuare delle richieste ogni volta che non riescono a realizzare una funzionalità. Questi sommari vengono poi periodicamente aggiornati tramite lo scambio di informazioni tra le varie cache: per far ciò, può essere utilizzata un protocollo di tipo gerarchico, che tuttavia è utilizzata solamente per lo scambio di informazioni sulle cache e non dei documenti veri e propri;

- le cache possono basarsi su un meccanismo centralizzato di memorizzazione dei documenti, ottenuto mediante un identificatore univoco all'interno del sistema. Le cache non memorizzano delle copie

dei documenti, bensì mantengono in memoria una lista degli identificatori dei documenti con l'indicazione della posizione. All'atto di richiesta di un documento, ad esempio, verrà fornita l'indicazione del nodo in cui è memorizzato il documento.

2.4.3 Caching ibrido

Il caching gerarchico e quello distribuito vengono già in larga parte utilizzati all'interno di Internet: negli USA, NLANR fornisce un'architettura per il caching gerarchico per distribuire informazioni popolari; in Europa, molti paesi hanno sviluppato dei sistemi di caching gerarchico per ridurre il numero di richieste che attraversano il già congestionato collegamento transoceanico.

Tuttavia, sono allo studio degli schemi ibridi, che racchiudono caratteristiche presenti sia nei protocolli di tipo gerarchico sia in quelli di tipo distribuito: essi si basano fondamentalmente su una suddivisione in livelli. Le entità presenti all'interno di un livello comunicano con le entità presenti all'interno dei livelli adiacenti tramite un protocollo di tipo gerarchico. All'interno di ogni livello, vengono invece utilizzati dei protocolli di comunicazione di tipo distribuito.

In [ROD01] si trova una trattazione esaustiva in cui vengono messi a confronto i differenti protocolli di caching: ciò che emerge da questa trattazione è che l'utilizzo dei protocolli ibridi di caching riduce i ritardi, l'utilizzo della banda trasmissiva e il carico all'interno delle singole cache. Non si vuole entrare nei dettagli di tutto ciò, tuttavia, visto l'importanza che riveste il risultato della trattazione, verranno ora discusse le prestazioni dei protocolli di caching in base ai principali parametri che li caratterizzano.

2.4.4 Confronto tra le differenti tipologie di caching

In questo paragrafo, verrà preso in considerazione il comportamento dei differenti protocolli di caching descritti nei precedenti paragrafi, sulla base dei parametri quantitativi che li caratterizzano: il tempo di connessione, il tempo di trasmissione, il tempo totale e la banda trasmissiva.

Tempo di connessione

Il tempo di connessione è la prima parte del ritardo percepito da un generico utente, dopo una richiesta di un documento.

In figura 2.3, è mostrato il tempo di connessione per il caching gerarchico, distribuito e ibrido, per un generico documento, la cui popolarità è indicata in figura con λ_{tot}, in un arco di tempo di 24 ore.

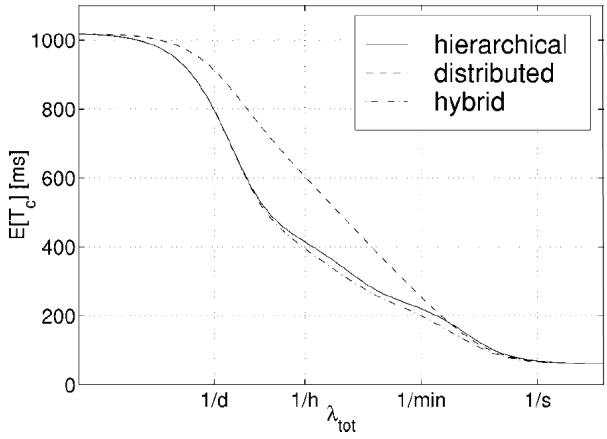

Figura 2.3: Tempo di connessione per il caching gerarchico, distribuito e ibrido.

Come si può vedere dalla figura, le prestazioni del caching gerarchico, distribuito e ibrido sono pressoché uguali: per i documenti poco popolari (con λ_{tot} piccola) il tempo di connessione è molto alto per il fatto che la richiesta di un documento viaggia sempre verso il server di origine.

Quando il numero delle richieste di un documento aumenta in maniera significativa, c'è aumenta la probabilità di trovarlo in una cache; se la richiesta di un documento varia tra una al giorno e una al minuto, il caching gerarchico e quello ibrido sono di una volta e mezzo più veloci di quello distribuito; in particolare, se la richiesta di un documento varia tra una l giorno e una al secondo, il caching ibrido è leggermente più efficiente di quello gerarchico.

Quando, infine, un documento è molto popolare, i tempi di connessione del caching gerarchico e distribuito sono veramente similari, poiché la probabilità di trovare un documento in una cache vicina è molto alta.

Tempo di trasmissione

La seconda parte del ritardo percepito da un generico utente, dopo una richiesta di un documento, è il tempo di trasmissione.

In figura 2.4 è mostrato il tempo di trasmissione di un documento: in questo caso, il caching distribuito e quello ibrido hanno delle prestazioni migliori rispetto al caching gerarchico quando la richiesta di un documento varia da una volta al giorno a una volta al minuto.

Nel caso in cui, invece, un documento sia poco popolare il tempo di trasmissione è molto alto, perché esso deve essere prelevato direttamente dal server di origine, mentre nel caso abbia una popolarità molto alta, la probabilità di trovarne una copia in una cache vicina è molto alta.

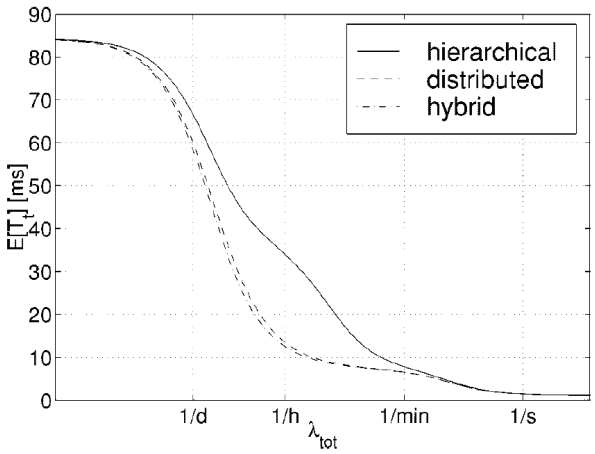

Figura 2.4: Tempo di trasmissione per il caching gerarchico, distribuito e ibrido.

Tempo totale

Il tempo totale è la somma del tempo di connessione e del tempo di trasmissione. Per i documenti più grandi è molto rilevante il tempo di trasmissione che il tempo di connessione, mentre, per i documenti più piccoli, predomina il tempo di connessione.

In figura 2.5 è mostrato il tempo totale per un documento di grandi dimensioni, utilizzando le tre tipologie di caching: in questo caso, la situa-

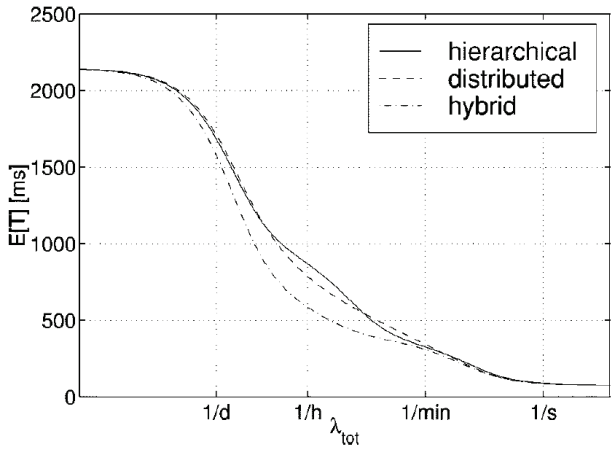

Figura 2.5: Tempo totale per il caching gerarchico, distribuito e ibrido.

zione è simile a quella del tempo di trasmissione, con un leggero calo delle prestazioni per il caching distribuito e per quello ibrido.

Banda trasmissiva

L'utilizzo di banda trasmissiva è uno dei parametri più importanti che caratterizzano un sistema di caching.

In figura 2.6 è mostrato l'utilizzo della banda trasmissiva per caching di tipo gerarchico, distribuito e ibrido. In quest'ultimo caso, sono presenti due differenti configurazioni in cui varia il numero di cache cooperanti in ogni livello (k).

Come si può notare dalla figura, il caching gerarchico e quello ibrido hanno all'incirca le stesse prestazioni, al contrario del caching distribuito, il quale mostra un peggioramento quando la richiesta di un documento è compresa tra una volta al giorno e una al minuto. Questo fatto è dovuto all'alta mole di informazioni che le cache devono scambiarsi per cooperare. Nel caso del caching ibrido, la presenza di cache intermedie riesce a diminuire il traffico tra le cache dello stesso livello, diminuendo così i ritardi e aumentando le prestazioni del sistema.

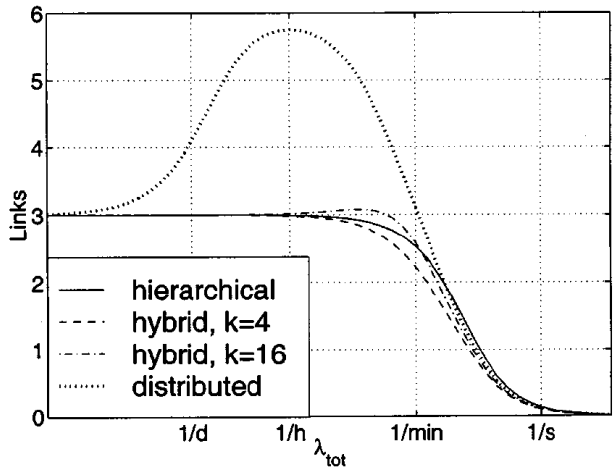

Figura 2.6: Utilizzo di banda trasmissiva per il caching gerarchico, distribuito e ibrido (con k=4 e k=16).

2.5 Utilizzo delle metodologie di caching

I concetti e gli strumenti descritti in questo capitolo sono alla base delle scelte effettuate nella fase di sviluppo del presente lavoro di tesi. La scelta dell'architettura di caching da utilizzare, con relativa struttura e relativi protocolli, è stata fondata sulle considerazioni sulle differenti architetture di caching presentate in questo capitolo.

Come verrà chiarito nei prossimi capitoli, il presente lavoro di tesi si prefigge lo sviluppo di un componente all'interno di un'applicazione multimediale distribuita, che realizzi dei meccanismi di caching. In particolare, l'attenzione si concentrerà sulla creazione di un sistema di caching di tipo distribuito e sulla gestione di una politica introdotta in questo capitolo, il placement.

La prerogativa fondamentale di un sistema di caching di tipo distribuito e dell'applicazione della politica di placement è la cooperazione tra le varie entità che gestiscono il sistema: per questo motivo, è stato necessario basarsi fortemente sulle considerazioni effettuate in questo capitolo al fine di scegliere una struttura e un protocollo per lo scambio di informazioni fra queste entità.

Dalle considerazioni fatte nel paragrafo 3.3, è emerso che la soluzione

ideale è la realizzazione di un sistema di caching con una struttura distribuita che fa uso di un protocollo di tipo gerarchico per lo scambio di informazioni tra le varie entità, dal momento che questo sistema di caching dovrà trattare dati di piccole dimensioni.

2.6 Conclusioni

In questo capitolo è stato preso in considerazione uno dei fattori con cui è possibile garantire la Qualità di Servizio in un sistema multimediale distribuito: il caching.

L'analisi effettuata permetterà di scegliere un'opportuna architettura da utilizzare nello sviluppo di un sistema di caching per il progetto di tesi.

Nel prossimo capitolo verranno esaminate le infrastrutture sulle quali verrà sviluppato il lavoro, in maniera tale da permettere al lettore la comprensione dei successivi capitoli.

Capitolo 3

Infrastrutture ad agenti mobili per sistemi distribuiti

In questo capitolo, dopo aver brevemente introdotto i paradigmi basati su mobilità di codice, verranno analizzati i due sistemi all'interno dei quali è stato sviluppato il progetto di tesi. Si tratta di SOMA [SOMA] e di MUM [MUM], rispettivamente un'infrastruttura per agenti mobili e un middleware, basato sulla precedente, per la fruizione di servizi multimediali.

La trattazione di questi due sistemi non sarà tuttavia esauriente ed esaustiva: verranno infatti delineate solamente le caratteristiche principali, con particolare riferimento agli aspetti che maggiormente interessano il progetto di tesi, per permettere al lettore la comprensione della trattazione successiva.

3.1 Paradigmi basati su mobilità di codice

I paradigmi basati su mobilità di codice aggiungono flessibilità e adattabilità nelle applicazioni distribuite. Essi prevedono, infatti, che lo scambio di informazioni, nella comunicazione tra le entità che compongono il sistema, non sia più di soli dati, ma anche di codice, in maniera tale da ridurre la trasmissione di ingenti moli di dati.

Il codice, che in questi casi viene detto *mobile*, risiede all'interno di un

ambiente di computazione che viene detto *Component*. I Component sono contenuto all'interno del *Computational Environment (CE)*, che fornisce loro un ambiente di esecuzione uniforme. Ogni Component Environment utilizza le funzionalità offerte dal Sistema Operativo.

I Component possono essere distinti in risorse e *execution unit (EU)*. Quest'ultima rappresenta un singolo thread con un proprio stato di esecuzione e uno spazio dei dati e può condividere delle risorse con altre EU. A seconda che lo stato di esecuzione della EU venga mosso o meno, si distinguono due tipi di mobilità: la mobilità forte, che supporta la migrazione sia del codice della EU sia del suo stato di esecuzione, e la mobilità debole, che supporta la migrazione del codice ed eventualmente di uno stato che permetta di continuare la computazione una volta ripresa l'esecuzione.

In relazione alle modalità di esecuzione e alla posizione delle EU e delle risorse una volta completata la computazione, si distinguono diversi paradigmi di mobilità di codice. In figura 3.1, per ogni paradigma è indicata la situazione delle risorse (*resource*), delle EU (*know-how*) e delle entità computazionali (*A o B*), in due differenti siti (S_A *e* S_B), prima e dopo l'esecuzione.

Paradigm	Before		After	
	S_A	S_B	S_A	S_B
Client-Server	A	know-how resource B	A	know-how resource **B**
Remote Evaluation	know-how A	resource B	A	*know-how* resource **B**
Code on Demand	resource A	know-how B	resource *know-how* **A**	B
Mobile Agent	know-how A	resource	—	*know-how* resource ***A***

Figura 3.1: Paradigmi per la mobilità di codice.

Nel classico modello *client/server* non c'è ovviamente mobilità di codice: la conoscenza per effettuare la computazione e le risorse necessarie

sono presenti sul sito S_B, che svolge il ruolo di server e ha il compito di soddisfare le richieste provenienti dal sito S_A, il client.

3.1.1 Remote Evaluation (REV)

Nel caso della Remote Evaluation, il componente A possiede la conoscenza necessaria all'esecuzione del servizio, ma non dispone delle risorse necessarie, presenti in questo caso sul sito S_B. Il componente A passa perciò al componente B il necessario know-how, che viene utilizzato da B per portare a termine la computazione e restituire un risultato ad A.

Esempi dell'applicazione di questo paradigma sono l'esecuzione di operazioni che richiedano un intenso uso di risorse computazionali, come complessi calcoli scientifici.

L'utilizzo del paradigma REV evita la trasmissione di ingenti quantità di dati all'interno della rete.

3.1.2 Code on Demand (CoD)

Nel caso del paradigma Code on Demand, sul sito S_A sono presenti le risorse necessarie alla computazione, ma non la conoscenza necessaria alla computazione. Per questo motivo, il componente A richiede il know-how al componente B, e, non appena ce l'ha disponibile, porta a termine il proprio compito.

Esempi dell'applicazione di questo paradigma sono gli aggiornamenti automatici di codice.

3.1.3 Paradigma ad agenti mobili

Il termine *agente* non trova in letteratura una definizione completa e definitiva. Esso, infatti, viene spesso utilizzato per descrivere cose differenti e in ambiti molto diversi dell'*Information Technology* (i due campi di ricerca in cui trova maggiormente spazio sono l'intelligenza artificiale e le reti di calcolatori).

Una possibile definizione di agente è la seguente: *L'agente è un'entità che possiede una o più delle seguenti proprietà: autonomia, socialità, reattività, pro-attività, mobilità, adattabilità e continuità temporale* (The Foundation for Intelligent Physical Agents, [FIPA]).

Nei sistemi attuali basati sul paradigma ad agenti, esistono diverse tipologie di agenti, dipendentemente dal ruolo che rivestono e dalle funzioni che devono assolvere: esistono, infatti, agenti intelligenti, agenti autonomi, agenti cooperanti, agenti mobili.

L'aggettivo *mobile*, che accompagna il termine agente, serve a delineare una sua particolare proprietà: l'agente è in grado di spostarsi tra i nodi di un sistema distribuito.

Gli agenti mobili costituiscono un mezzo, particolarmente efficace, per strutturare applicazioni in ambienti distribuiti. La loro efficacia è dovuta alle seguenti carattteristiche:

- **Indipendenza**: gli agenti possono risiedere in differenti nodi della rete e svolgere autonomamente le proprie attività;
- **Interazione**: gli agenti possono comunicare fra di loro all'interno del sistema, per portare a termine i loro compiti;
- **Mobilità**: gli agenti possono migrare autonomamente, secondo necessità, da un nodo all'altro della rete.

Affinché sia possibile creare ed eseguire agenti mobili è necessaria un infrastruttura che implementi il paradigma appena introdotto.

3.2 Un'infrastruttura per agenti mobili: SOMA

SOMA (Secure and Open Mobile Agent) è un infrastruttura per agenti mobili realizzata presso il DEIS (Dipartimento di Elettronica Informatica e Sistemistica) della Facoltà di Ingegneria Informatica dell'Università di Bologna.

Il sistema è realizzato interamente utilizzando il linguaggio Java [JAVA], il quale risulta particolarmente adatto per la realizzazione di applicazioni ad agenti mobili per molteplici motivi:

- È un linguaggio interpretato e per questo motivo può eseguire su una qualunque macchina che possegga una Java Virtual Machine (JVM) in grado di trasformare il bytecode in istruzioni macchina. Risulta perciò portabile su diverse architetture software/hardware;
- È un linguaggio object-oriented che permette uno sviluppo modulare del codice per estensione ed ereditarietà, così che l'utente possa scrivere i propri agenti con un sforzo limitato a partire dalle classi messe a disposizione dalla piattaforma;
- Prevede un meccanismo di caricamento dinamico delle classi anche da sorgenti remote e di collegamento dinamico all'applicazione corrente;
- Fornisce la possibilità di serializzare gli oggetti, cioè di rappresentarli come stream di byte e di trasferirli in rete;
- Ha caratteristiche di sicurezza built-in, la più nota delle quali è quella associata alle applet: queste ultime, originariamente, potevano essere scaricate da un Web server, ma non avevano diritto di accedere alle risorse del sistema locale. Dalla versione 1.2 del linguaggio si ha un'architettura di sicurezza rivisitata, molto più flessibile ed espressiva, fatta di domini di protezione, controlli d'accesso e permessi da associare sia a codice remoto che a codice locale.

3.2.1 Caratteristiche principali di SOMA

Le caratteristiche principali di SOMA sono le seguenti:

- **astrazioni di località**: esistono due diversi livelli di astrazione, il *place* ed il *dominio*, che rappresentano, tramite un'interfaccia semplice ed uniforme, le tipiche località di Internet: il nodo e la rete locale (LAN);

- **scalabilità**: è disponibile, a vari livelli, dalla configurazione del sistema, la cui topologia può essere estesa a piacimento, alla gestione degli utenti, al numero di agenti in esecuzione;

- **sicurezza**: è realizzata la protezione degli ambienti di esecuzione, mentre gli agenti sono protetti solo in parte (tramite la verifica di integrità del codice e la riservatezza nella comunicazione dello stato). La protezione completa degli agenti, tuttavia, richiederebbe l'utilizzo di un apposito supporto hardware. L'interazione degli utenti, per quanto riguarda l'introduzione di nuovi agenti e l'amministrazione del sistema, è controllata tramite l'utilizzo di password;

- **apertura**: il sistema intende aderisce alla standardizzazione proposta da OMG [OMG], basata su CORBA [CORBA], che permette agli agenti di interagire con applicazioni *CORBA-compliant* e con gli agenti di altri sistemi che aderiscono allo stesso standard;

- **dinamicità**: la configurazione del sistema, per quanto riguarda la gestione degli ambienti di esecuzione e gli utenti che hanno diritto ad interagire con il sistema, è di tipo dinamico. È possibile cioè aggiungere o rimuovere ambienti di esecuzione ed utenti in qualsiasi momento;

- **prestazioni**: attraverso l'uso del modello ad agenti mobili si intende superare i limiti del modello *cliente/servitore*, ottenendo migliori prestazioni globali dalle applicazioni. In particolare, la gestione della sicurezza, punto determinante nel raggiungere buone prestazioni, è configurabile in maniera flessibile, in maniera tale da raggiungere il compromesso desiderato tra prestazioni e sicurezza;

3.2.2 Architettura di SOMA

La topologia introdotta da SOMA nasce dall'imitazione della struttura di un sistema complesso come Internet: esso è caratterizzato da una serie di località organizzate in gerarchia, ognuna con le proprie caratteristiche,

dal punto di vista dell'amministrazione, della gestione e della sicurezza; le varie località sono connesse tra loro in maniera dinamica, realizzando in questo modo una topologia variabile nel tempo.

SOMA cerca di modellare una simile struttura, introducendo diverse astrazioni di località, che descrivono le tipiche località di Internet e forniscono nel contempo un ambiente in cui sia possibile sviluppare applicazioni secondo il modello ad agenti mobili.

Le astrazioni di località introdotte sono:

- **Place**: rappresenta l'ambiente di esecuzione degli agenti, come astrazione del concetto di nodo (Figura 3.2). Al suo interno gli agenti possono interagire con le risorse locali e cooperare con altri agenti residenti sul place. Ogni place è caratterizzato da un nome univoco. È inoltre possibile che ogni nodo contenga più place, mentre ogni place deve essere confinato all'interno di un nodo.

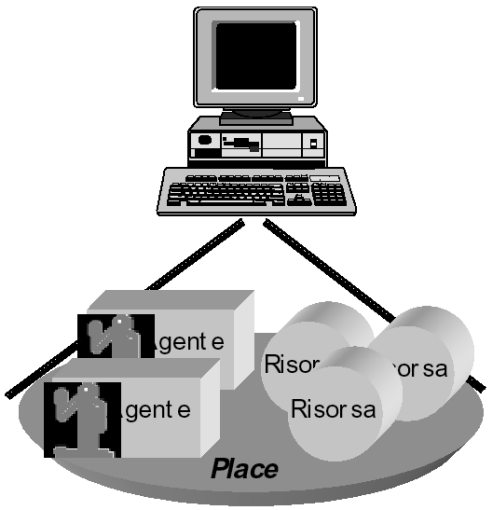

Figura 3.2: Astrazione di place in SOMA.

- **Dominio**: rappresenta un'aggregazione di place, come astrazione del concetto di rete locale (Figura 3.3), che presentano caratteristiche comuni di tipo fisico (appartenenza alla medesima LAN) o logico (amministratore e politiche comuni). Ogni dominio è caratterizzato da un nome univoco.

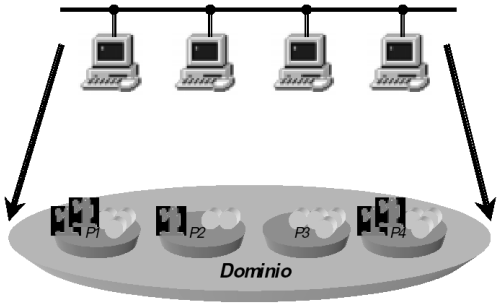

Figura 3.3: Astrazione di dominio in SOMA.

I place di un dominio sono strettamente correlati tra loro, si conoscono a vicenda e sono in grado di comunicare direttamente.

Le interazioni che coinvolgono due diversi dominii sono invece mediate da un particolare place, detto *Gateway* o *Default Place*. I Gateway hanno conoscenza uno dell'altro e sono in grado di comunicare direttamente: essi rappresentano quindi gli unici punti di accesso verso l'esterno di un dominio (Figura 3.4).

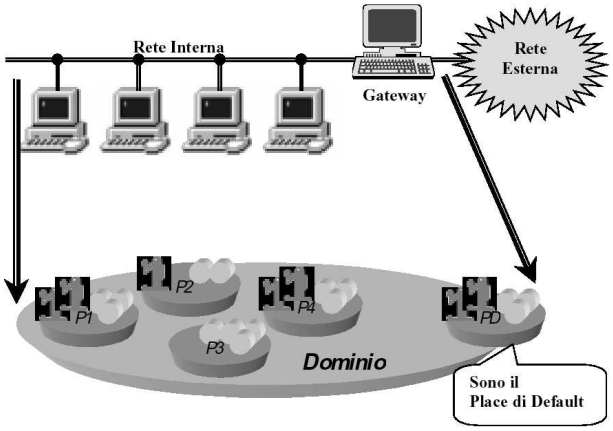

Figura 3.4: Astrazione di gateway (default place) in SOMA.

I domini sono organizzati gerarchicamente mediante una topologia ad albero, che permette l'identificazione univoca di un certo place/default place, tramite il percorso che conduce dal default place *radice* al place/default place stesso.

In figura 3.5 è riportata una gerarchia di facile comprensione.

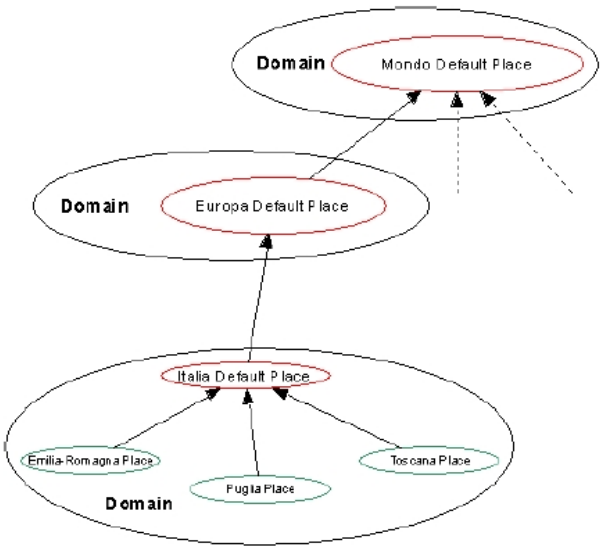

Figura 3.5: Organizzazione gerarchica dei domini in SOMA.

Il place è l'ambiente di esecuzione dell'agente, realizzato mediante moduli di supporto che forniscono i servizi fondamentali. Questi moduli gestiscono sia gli agenti, permettendone l'esecuzione, l'ingresso e l'uscita da un place, che le interazioni tra i vari place, mantenendo i necessari canali di comunicazione, ed infine anche le informazioni sui place appartenenti ad un dominio.

Tutti gli agenti vengono associati ad un identificatore unico, AgentID, ricavato dalla combinazione dell'identificatore del place sul quale nasce e di un intero progressivo, in modo da conoscere sempre l'origine di un agente. Per identificare Place e Domini si utilizza un sistema di naming che si basa su un identificatore detto PlaceID. Tale identificatore è il risultato della concatenazione del nome dominio e del nome del place. Nel caso di un place di default quest'ultimo sarà semplicemente nullo.

In SOMA ogni agente possiede la capacità di comunicare con gli altri agenti, dal momento che questa è una ulteriore caratteristica di base. Questo avviene tramite scambio di messaggi che risulta essere una soluzione molto flessibile. Vi è inoltre un servizio di localizzazione degli agenti, attivabile su necessità, che permette di avere grande trasparenza.

Infine si deve sottolineare come l'ambiente SOMA garantisca il rispetto di una politica di autorizzazione così che agenti maliziosi non interagiscano in modo incontrollato con le risorse ed i servizi messi a disposizione dall'ambiente. La definizione di diverse astrazioni di località, inoltre, consente di introdurre politiche di sicurezza nelle quali le azioni siano controllate sia a livello di dominio, sia a livello di place.

3.3 Middleware di supporto per applicazioni multimediali: MUM

MUM (Multimedia agent based Ubiquitous Multimedia middleware) è un sistema sviluppato, come SOMA, presso il DEIS (Dipartimento di Elettronica Informatica e Sistemistica) della Facoltà di Ingegneria Informatica dell'Università di Bologna [FOS03].

MUM fornisce una piattaforma per la gestione dei flussi multimediali, offrendo un valido supporto all'utilizzo delle risorse e facilitando lo sviluppo di applicazioni di nuova generazione, ispirate a modelli di *ubiquitous computing*. In particolare, fornisce il supporto per la pubblicazione e la fruizione di presentazioni multimediali di varia natura e qualità diverse su rete fissa e mobile.

Il sistema fornisce agli utilizzatori la possibilità di richiedere il delivery dei titoli presenti, comandare la presentazione e richiedere lo spostamento della sessione verso un terminale diverso da quello che stanno attualmente utilizzando. Se il sistema viene utilizzato da rete mobile, ossia da un device di tipo wireless, il sistema fornisce un supporto per seguire i movimenti dell'utente. Inoltre l'infrastruttura si occupa anche della gestione delle risorse supportando prenotazione e monitoraggio delle stesse, nonché della scelta della presentazione più adatta alla piattaforma computazionale dalla quale un certo utente sta accedendo al sistema. Tale specifica si realizza tramite lo studio delle caratteristiche delle diverse piattaforme, le cui caratteristiche sono salvate in dei descrittori che sono riferiti nei pro-

fili degli utenti. Infine, MUM supporta il downloading e l'inizializzazione di software a runtime.

3.3.1 Fruizione del materiale multimediale

MUM è realizzato in ambiente distribuito ed adotta, come modello computazionale, una integrazione tra il tipico modello client/server ed il modello ad agenti mobili, cercando di sfruttare il meglio di questi due approcci.

Si introducono le seguenti cinque entità: *Client*, *Server*, *Proxy*, *ClientAgent* e *ProxyAgent*. Le prime tre sono realizzate come entità fisse, mentre le due rimanenti sono realizzate come agenti mobili. Le entità sono tra loro collegate a formare un percorso, detto *service path*, che va dal Client al Server e che coinvolge tutti quei nodi intermedi che ospitano i Proxy. Tale percorso è attivo e riconfigurabile.

- Il **Client** è l'*end-point* che riceve i flussi multimediali richiesti. Questa entità non è realizzata come agente mobile, ma come entità fissa residente sulla macchina dalla quale l'utente effettua l'accesso al sistema. Questo non implica che il software richiesto per l'esecuzione di questa entità debba necessariamente essere presente sulla macchina dalla quale verrà poi lanciato, dal momento che è stato realizzato in modo da poter scaricare questo codice a runtime, seguendo il paradigma *code on demand*.

- Il **Server** è per sua natura molto simile al Client ed i discorsi fatti in precedenza mantengono qui la loro validità. Infatti anche questa entità è realizzata come fissa ed è scaricabile secondo il paradigma code on demand. Inoltre, il server, una volta che sia stato avviato su una certa macchina, verrà utilizzato per servire tutte le richieste che da quel punto in poi arriveranno a quella macchina per quel tipo di server. Si può facilmente intuire come questa entità sia definita come l'altro end-point della comunicazione dei flussi multimediali.

In pratica è il Server a spedire i flussi multimediali ad un Client che ne ha fatto richiesta.

- Il **Proxy** è un'entità di fondamentale importanza in quanto partecipa attivamente alla consegna del servizio. Le sue principali funzioni sono:

  - gestisce le eventuali disconnessioni che si possono verificare, in particolare quelle intermittenti, potendo fare del caching dell'oggetto multimediale che è stato richiesto dall'utente;
  - incapsula la logica di trasformazione dei dati multimediali, ad esempio per rispondere alle esigenze di terminali mobili con basse capacità computazionali;
  - incapsula la logica di gestione della Qualità di Servizio: può, cioè, partecipare attivamente nel Service Path, monitorando le condizioni delle risorse e richiedendo la riconfigurazione del servizio in caso di malfunzionamenti.

In figura 3.6 è mostrato un Service Path, tra un client C e un server S, in una sotto-porzione di un ipotetico albero dei domini. Le entità proxy P vengono a trovarsi tra il client ed il server: in figura è presente un solo Proxy, ma, generalmente, possono esserci più proxy.

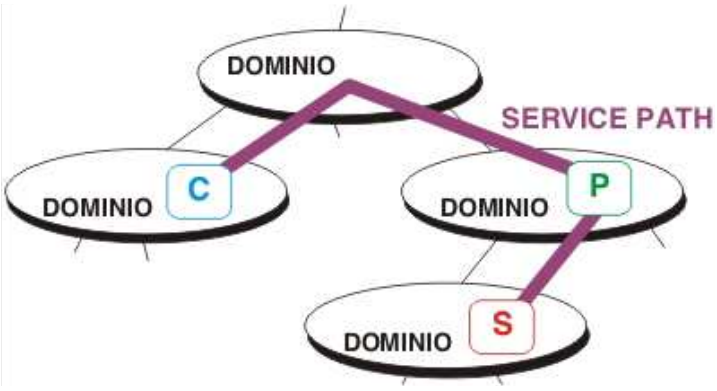

Figura 3.6: Esempio di Service Path in MUM.

- Il **ClientAgent** è l'*entry-point* del sistema dal lato client. Questa entità è realizzata come agente mobile ed il suo compito è quello di inizializzare la sessione per l'utilizzatore e gestire poi le interazioni col resto del middleware distribuito. Questo agente può essere considerato come una sorta di mediatore che accetta le richieste fatte dall'utente attraverso una opportuna GUI, per interagire col resto del sistema e richiedere l'esecuzione dell'operazione richiesta. Il ClientAgent è stato pensato come un agente per facilitare l'interazione con gli altri agenti, e per permettere il movimento di queste entità, in modo da supportare gli utenti che si muovono fra i diversi terminali (Figura 3.7).

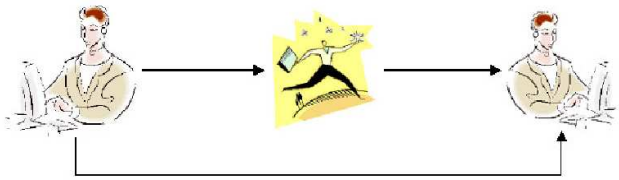

Figura 3.7: Movimento degli utenti in MUM.

- Il **ProxyAgent** è anch'esso realizzato come un agente mobile, in maniera tale da supportare il movimento dei terminali (Figura 3.8). Per la gestione vera e propria dei flussi il ProxyAgent si avvale dell'entità Proxy precedentemente introdotta.

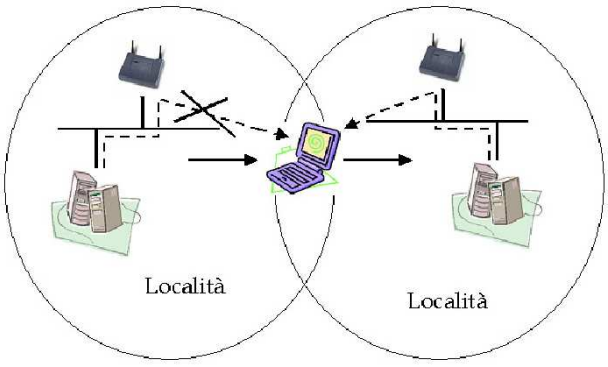

Figura 3.8: Movimento dei terminali in MUM.

3.3.2 Gestione dei contenuti multimediali

Il servizio di gestione dei contenuti multimediali si occupa di come le presentazioni multimediali all'interno del sistema vengono rappresentate e del protocollo che permette di gestirle.

La presentazione multimediale è un'aggregazione di diversi oggetti multimediali: in essa si specifica la qualità della presentazione e la sua posizione all'interno del sistema.

Ogni presentazione è poi partizionata in sottosistemi mutuamente esclusivi, detti contenuti multimediali. Tale suddivisione è introdotta per raggruppare presentazioni multimediali che portano lo stesso contenuto informativo, ossia che hanno la stessa struttura e lo stesso titolo. Naturalmente vi sono poi delle differenziazioni a partite dai livelli di qualità, ma non sono escluse presentazioni con medesima struttura e livello di qualità. L'identificatore del contenuto multimediale all'interno dell'intero sistema è detto titolo.

Il protocollo che permette l'utilizzo delle presentazioni all'interno del sistema è attualmente basato su un architettura di tipo centralizzato: esiste, cioè, un unico database ed un unico gestore di tale database, residenti sul nodo radice. La comunicazione tra il gestore e gli utilizzatori è di tipo client/server: ogni client, infatti, ha a disposizione un riferimento al gestore, con il quale comunica per effettuare le operazioni che desidera (ad esempio, ricevere informazioni su una determinata presentazione, richiedere l'inserimento di una presentazione, etc.).

La soluzione adottata non è particolarmente scalabile: questo progetto di tesi si prefigge, per l'appunto, il miglioramento del servizio di gestione dei contenuti multimediali attraverso l'introduzione di un servizio distribuito che utilizza dei meccanismi di caching.

3.4 Conclusioni

In questo capitolo sono state prese in considerazione le infrastrutture sulle quali si basa il lavoro di tesi, mettendo in luce le principali caratteri-

stiche e gli aspetti che maggiormente interessano la trattazione successiva.

Nel prossimo capitolo si parlerà di alcuni standard internazionali per la rappresentazione di contenuti multimediali. Si riuscirà, in questo modo, a scegliere un modello che verrà adottato successivamente nello sviluppo del progetto di tesi.

Capitolo 4

Standard per la descrizione di presentazioni multimediali

Con la crescita esponenziale di Internet e delle reti di computer in generale, e con l'utilizzo sempre più massiccio di applicazioni multimediali distribuite, è stato necessario creare degli standard per la rappresentazione dei contenuti multimediali che rendano più semplice ed efficiente lo scambio di informazioni.

In questo capitolo si cercherà di far fronte all'esigenza di rappresentare i contenuti multimediali all'interno di MUM attraverso l'analisi di due standard internazionali per la rappresentazione di presentazioni multimediali: MPEG-7 [MPEG7] e Dublin Core [DC].

L'obiettivo è di riuscire a definire un modello per la rappresentazione dei contenuti multimediali da utilizzare nel successivo sviluppo del sistema che sia il più possibile coerente con questi due standard internazionali.

4.1 MPEG-7

MPEG-7 (formalmente denominato *Multimedia Content Description Interface*) si pone nell'ottica di fornire strumenti e standard per la descrizione di dati di tipo multimediale, non considerando la loro particolare tecnologia di memorizzazione e trasmissione, ma ponendosi come vali-

do strumento per la gestione e interpretazione del contenuto informativo presente in essi.

Utilizzando un'espressione inglese si potrebbe dire che riguarda *bits about the bits*, ossia dati riguardanti dati, che vengono molto più comunemente indicati con il termine di *metadati*.

MPEG-7 si pone l'obiettivo di essere compatibile con le soluzioni già esistenti, al fine di garantire una meno traumatica migrazione verso quello che potrebbe essere lo standard del futuro in tale ambito.

Come ogni standard aperto, MPEG-7 non si focalizza su particolari applicazioni ma vuole invece essere utilizzabile dal più vasto numero possibile di programmi e piattaforme.

Anche se l'obiettivo di MPEG-7 non è la descrizione di documenti testuali, questi possono essere considerati come contenuti di tipo informativo e quindi descritti mediante le tecniche descritte da tale standard.

Il fine ultimo di MPEG-7 risulta essere quello di fornire un ricco insieme di strumenti standard per la generazione e la comprensione di descrittori di presentazioni multimediali in maniera tale da facilitare lo scambio e la ricerca nella applicazioni multimediali distribuite e nella rete Internet.

MPEG-7 è formato dai seguenti sette moduli:

- *MPEG-7 Systems*: strumenti di supporto;
- *MPEG-7 Description Definition Language*: linguaggio che permette la creazione di nuove caratteristiche e di nuovi schemi;
- *MPEG-7 Audio*: descrizioni di tipo Audio;
- *MPEG-7 Visual*: descrizioni di tipo Video;
- *MPEG-7 Multimedia Description Schemes*: descrizioni applicabili ad ogni informazione multimediale;
- *MPEG-7 Reference Software*: implementazione software delle parti più rilevanti dello standard;
- *MPEG-7 Conformance*: linee guida e procedure per testare la conformità allo standard delle varie implementazioni.

Non si vuole entrare nei dettagli tecnici di ogni modulo che compone lo standard, perché ciò sarebbe superfluo ai fini del progetto di tesi. Si cercherà, perciò, di mettere in evidenza gli aspetti fondamentali.

4.1.1 MPEG-7 Systems

All'interno di questo modulo vengono definiti due elementi fondamentali dello standard: i Descriptor (D) e gli Description Schemes (DS).

Descriptor

Definiscono la sintassi e la semantica delle caratteristiche di un dato multimediale. I contenuti di tipo audio-visivo che possono essere rappresentati utilizzando lo standard MPEG-7 sono: immagini, grafici, modelli tridimensionali, audio, dialoghi, video, etc. Tramite i Descriptor è possibile rappresentare le relazioni che intercorrono tra tutti gli elementi che caratterizzano uno di questi dati multimediali.

MPEG-7 permette di rappresentare i dati multimediali con differenti livelli di dettaglio e con differente granularità. Ad esempio, in una descrizione di basso livello di un dato visuale, potrebbe essere necessario rappresentare la grandezza, la texture, il colore, la posizione, e altro.

I Descriptor si dividono in due categorie: i Descriptor per la descrizione di informazioni legate al contenuto, come le informazioni di produzione, di utilizzo, fisiche, strutturali, etc., e i Descriptor per la descrizione di informazioni non legate al contenuto, come la classificazione o il contesto.

Description Schemes

Specificano la struttura e la semantica della relazioni tra i componenti che compongono la descrizione di un dato multimediale.

Un Description Scheme estende il set di Descriptor di MPEG-7, combinando tra loro elementi già esistenti, creando strutture più complesse e definendo le relazioni che intercorrono tra gli elementi che compongono queste nuove strutture.

Poiché molte volte è necessario ripetere all'interno dei DS degli elementi di utilizzo comune, come il *tempo*, un *vettore* o una *matrice*, esistono particolari schemi in cui questi elementi vengono definiti per poter essere riutilizzati all'interno di altri schemi.

Oltre a questi schemi di utilizzo generale, esistono degli schemi molto più complessi, che possono essere utilizzati quando bisogna rappresentare più di un dato multimediale, ad esempio audio e video.

4.1.2 MPEG-7 Description Definition Language

Il DDL è la parte principale dello standard. Esso fornisce, infatti, un solido linguaggio mediante il quale gli utenti creano gli schemi dei propri documenti e, se necessario, possono creare le proprie descrizioni a partire da quelle definite nello standard (sempre mediante il DDL).

Il DDL deve essere in grado di rappresentare le relazioni spaziali, temporali, strutturali e concettuali all'interno di una presentazione multimediale. Esso deve essere indipendente dalla piattaforma e dall'applicazione utilizzati, comprensibile sia dagli utenti che dai dispositivi elettronici.

Il DDL si basa su *XML Schema* [XML]: la descrizione della presentazione viene memorizzata in un file di tipo XML, utilizzando gli elementi definiti attraverso lo schema XML, e viene validata da un *parser*. Uno schema XML può essere dunque utilizzato per la creazione della descrizione di più presentazioni multimediali.

Grazie all'utilizzo degli schemi XML, che possono anche essere ampliati secondo le esigenze particolari di ogni utente, MPEG-7 riesce a fornire un approccio completo per la descrizione delle proprietà di una presentazione multimediale.

In alcuni casi, tuttavia, questo approccio rende molto complesse anche delle descrizioni per le quali basterebbe un sotto-insieme di tutto il DDL. Per far fronte a questo problema, esistono altri standard, più *leggeri*, che utilizzano un vocabolario ridotto per la descrizione delle presentazioni multimediali. Uno di questi è il Dublin Core.

4.2 Dublin Core

Il Dublin Core nasce nel marzo del 1995 da un gruppo di lavoro organizzato dall'Online Computer Library Center (OCLC), che ha sede a Dublin, in Ohio (USA), e dal National Center for Supercomputer Application (NCSA) con l'obiettivo di sviluppare un insieme di metadati per la descrizione delle informazioni elettroniche in rete.

In ambito OCLC è nata quindi la comunità dello standard di metadati Dublin Core, che raggruppa produttori, autori e detentori dei diritti sulla documentazione in rete, allo scopo di fornire strumenti per l'accesso alle risorse digitali.

4.2.1 Uso e applicabilità del Dublin Core

Lo standard di metadati Dublin Core è un insieme di elementi semplice ma efficace per descrivere un'ampia gamma di risorse.

Il Dublin Core standard comprende 15 elementi, la cui semantica è stata stabilita attraverso il consenso di un gruppo internazionale e interdisciplinare che comprende professionisti provenienti dal mondo bibliotecario, dall'ambiente museale, dalle comunità di ricerca di biblioteche digitali, ed anche esperti di informatica, di codifica di testi e di altri campi culturali.

Le caratteristiche principali di Dublin Core sono le seguenti:

- *Semplicità di creazione e utilizzo*. Il set di elementi di Dublin Core è stato mantenuto più limitato e semplice possibile per permettere a specialisti ma anche a non catalogatori di creare, facilmente ed economicamente, semplici record di descrizione di risorse utilizzabili poi per l'effettiva scoperta delle risorse stesse dalla rete.

- *Interoperabilità semantica*. Il recupero di informazioni di interesse per un utente è reso difficile dalle differenze in terminologia e metodi descrittivi che si riscontrano nei diversi campi della conoscenza. A fronte di questo problema, il Dublin Core fornisce un comune insieme di dati, il cui significato e valore è universalmente compreso e supportato.

- *Promozione di uno strumento utile per una infrastruttura a livello internazionale*. Il set di elementi di Dublin Core è stato sviluppato originariamente in lingua inglese, ma sono state create versioni in molte altre lingue. Un apposito gruppo di lavoro sul Dublin Core si sta occupando del collegamento delle diverse versioni in un registro distribuito.

- *Estendibilità* Per bilanciare il bisogno di semplicità nella descrizione di risorse digitali con l'esigenza di recupero di precise risorse, gli sviluppatori di Dublin Core hanno riconosciuto l'importanza di fornire un meccanismo per estendere il set di elementi standard per soddisfare la necessità di reperimento delle risorse. Altre comunità di esperti di metadati potranno creare e amministrare insiemi di elementi aggiuntivi, che potrebbero essere collegati al set di elementi Dublin Core per far fronte alle esigenze di estendibilità.

4.2.2 Elementi del Dublin Core

Il Dublin Core Metadata Element Set (DCMES) è formato da 15 elementi:

- **Title**: il nome dato alla risorsa dal *Creator* o dal *Publisher*;

- **Creator**: la persona o l'organizzazione che ha la responsabilità principale della creazione del contenuto intellettuale della risorsa;

- **Subject**: l'argomento di cui tratta la risorsa. In particolare il soggetto può essere espresso da parole chiave o frasi che descrivono il soggetto o il contenuto della risorsa;

- **Description**: una descrizione in forma testuale del contenuto della risorsa;

- **Publisher**: la persona o l'ente responsabile della produzione della risorsa;

- **Contributor**: una persona o un ente non compreso in alcun elemento *Creator* che ha dato significativi contributi intellettuali alla creazione della risorsa;
- **Date**: la data in cui la risorsa è stata resa disponibile nella sua forma presente;
- **Type**: la categoria della risorsa;
- **Format**: il formato della risorsa;
- **Identifier**: una sequenza di caratteri alfabetici o numerici usata per identificare univocamente la risorsa;
- **Source**: una sequenza di caratteri alfabetici o numerici, usata per identificare univocamente l'opera dalla quale è derivata la risorsa, se applicabile;
- **Language**: il linguaggio del contenuto intellettuale della risorsa;
- **Relation**: la relazione di questa risorsa con altre risorse;
- **Coverage**: le caratteristiche spaziali e/o temporali della risorsa;
- **Rights**: un collegamento con una segnalazione di copyright, con un'indicazione di gestione di diritti, o con un servizio che fornisce informazioni sui termini di accesso alla risorsa.

Poiché l'uso primario del set di elementi del Dublin Core è la descrizione di risorse presenti nei musei, nelle biblioteche, nelle agenzie governative o nelle organizzazioni commerciali, 15 elementi sono più che sufficienti.

Quando, però, bisogna descrivere una presentazione multimediale, allora è possibile arricchire il set di elementi utilizzando un linguaggio di qualificazione dei nuovi elementi.

Nel prossimo paragrafo, verrà proposta una maniera alternativa di rappresentazione delle presentazioni multimediali, attraverso l'utilizzo del Dublin Core e di alcuni elementi dello standard MPEG-7, che potrà essere utilizzata all'interno di MUM.

4.3 Descrizione di presentazioni multimediali

Dall'analisi dello standard MPEG-7 e del Dublin Core è emerso che, nonostante siano a disposizione degli sviluppatori degli strumenti per la rappresentazione di presentazioni multimediali, molte volte questi strumenti risultano essere o insufficienti o troppo complessi per ogni singola esigenza.

Per questo motivo, In questo paragrafo, si cercherà di creare uno schema per la rappresentazione di presentazioni multimediali che risulti più semplice dello standard MPEG-7 ma allo stesso tempo più espressivo del Dublin Core.

Questo nuovo schema si basa su alcuni elementi dell'insieme di elementi del Dublin Core e su alcuni elementi dello standard MPEG-7.

Fondamentalmente, le caratteristiche di un dato multimediale possono essere suddivise in due parti: le informazioni bibliografiche e le informazioni fisiche. Per la rappresentazione delle prime si è scelto di utilizzare una parte del set di elementi del Dublin Core. Per la rappresentazione delle seconde, invece, si è scelto di utilizzare quattro elementi dello standard MPEG-7.

Il nostro schema per la rappresentazione di dati multimediali risulta quindi essere formato dai seguenti cinque componenti:

1. DCMES - Si utilizza una parte del set di elementi del Dublin Core per specificare le informazioni bibliografiche associate alla risorsa. In particolare, gli elementi utilizzati sono: title, creator, subject, description, identifier, language e rights;

2. MPEG-7 MediaLocator Descriptor - Serve per definire la locazione della risorsa;

3. MPEG-7 MediaTime Descriptor - Utilizzato per definire gli attributi temporali (tempo iniziale e durata) associati alla risorsa;

4. MPEG-7 MediaFormat Descriptor - Specifica le informazioni sul formato e sulla codifica della risorsa;

5. MPEG-7 TemporalDecomposition Description Scheme - Descrivere le componenti spazio-temporali del video.

Nello schema risultante, inoltre, dovrà essere inserita l'indicazione dei *namespaces* relativi al vocabolario dei termini utilizzati per il DCMES e per gli elementi che fanno parte di MPEG-7.

In figura 4.1, è rappresentato uno schema del modello che verrà utilizzato: la descrizione del singolo dato multimediale è composta dai cinque elementi descritti precedentemente.

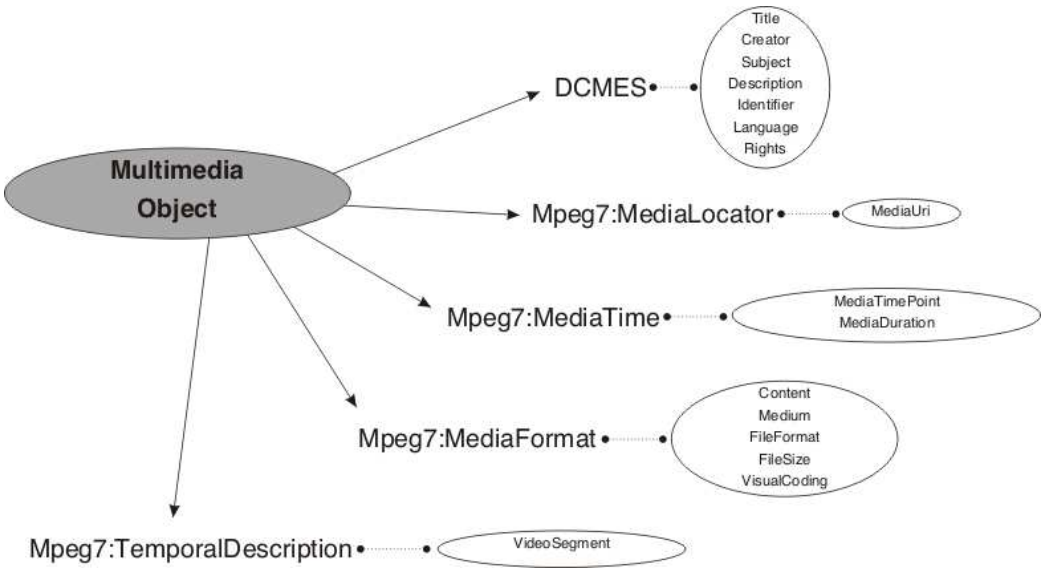

Figura 4.1: Modello per la descrizione di un generico dato multimediale.

Ecco dunque il nostro nuovo schema per la rappresentazione di dati multimediali:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:mpeg7="http://www.mpeg.org/MPEG/2000/" >

  <import namespace="http://purl.org/dc/elements/1.1/" />
  <import namespace="http://www.mpeg.org/MPEG/2000/" />

  <element name="MultimediaDescription">
    <complexType>
      <sequence>
```

```
        <element ref="dc:title" minOccurs="1" maxOccurs="1" />
        <element ref="dc:creator" minOccurs="0" maxOccurs="unbounded" />
        <element ref="dc:subject" minOccurs="0" maxOccurs="unbounded" />
        <element ref="dc:description" minOccurs="1" maxOccurs="1" />
        <element ref="dc:identifier" minOccurs="1" maxOccurs="1" />
        <element ref="dc:language" minOccurs="0" maxOccurs="1" />
        <element ref="dc:rights" minOccurs="0" maxOccurs="1" />

        <element name="MediaLocator" type="mpeg7:MediaLocatorType"
                  minOccurs="1" maxOccurs="1" />
        <element name="MediaTime" type="mpeg7:MediaTimeType"
                  minOccurs="1" maxOccurs="1" />
        <element name="MediaFormat" type="mpeg7:MediaFormatType"
                  minOccurs="1" maxOccurs="1" />
        <element name="TemporalDecomposition"
                  type="mpeg7:VideoSegmentTemporalDecompositionType"
                   minOccurs="0" maxOccurs="1" />

      </sequence>
    </complexType>
  </element>
</schema>
```

Tramite questo schema è possibile creare delle rappresentazioni di qualsiasi dato multimediale. Per mostrare ciò, proviamo ad utilizzare lo schema per la descrizione della seguente presentazione multimediale:

- Caratteristiche bibliografiche:

  titolo: World News Tonight

  creatore: Special Broadcasting Service

  soggetto: International news events

  descrizione: Comprehensive coverage of global and national events, presented by Anton Enus

  identificatore: news-2002-02-12

  lingua: EN

  diritti: all content © SBS 2000

- Caratteristiche fisiche:

  locazione della presentazione: file://disk/news-12-02-02.mpg

tempo iniziale: 00:00, durata 30 minuti

contenuto: audiovisivo, supporto: hard-disk, formato: mpeg, grandezza del file: 660 MB, codifica: MPEG-1 Video, caratteristiche video: 288x352x25

primo segmento: tempo iniziale: 0, durata 15 minuti, descrizione: Arafat intterview

secondo segmento: tempo iniziale: 15 minuti, durata 15 minuti, descrizione: Anti-War Protests

L'istanza della presentazione appena descritta, che si ottiene utilizzando il nuovo schema precedentemente descritto, è la seguente:

```
<?xml version="1.0" ? encoding="UTF-8" ?>

<MultimediaDescription xmlns:dc="http://purl.org/dc/elements/1.1/"
                       xmlns:mpeg7="http://www.mpeg.org/MPEG/2000/"
                       xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
                       xsi:schemaLocation="MultimediaDescriptionSchema
                                           http://purl.org/dc/elements/1.1/
                                           http://www.mpeg.org/MPEG/2000/" >

  <dc:title>World News Tonight</dc:title>
  <dc:creator>Special Broadcasting Service</dc:creator>
  <dc:subject>International news events</dc:subject>
  <dc:description>Comprehensive coverage of global and national events,
                  presented by Anton Enus</dc:description>
  <dc:identifier>news-2002-02-12</dc:identifier>
  <dc:language>EN</dc:language>
  <dc:rights>all content © SBS 2000</dc:rights>

  <MediaLocator>
    <mpeg7:MediaUri>file://disk/news-12-02-02.mpg</mpeg7:MediaUri>
  </MediaLocator>

  <MediaTime>
    <mpeg7:MediaTimePoint>T00:00:00</mpeg7:MediaTimePoint>
    <mpeg7:MediaDuration>PT30M00S</mpeg7:MediaDuration>
  </MediaTime>

  <MediaFormat>
    <mpeg7:Content>
      <mpeg7:Name>audiovisual</mpeg7:Name>
    </mpeg7:Content>
    <mpeg7:Medium>
      <mpeg7:Name>HD</mpeg7:Name>
    </mpeg7:Medium>
```

```
    <mpeg7:FileFormat>
      <mpeg7:Name>mpeg</mpeg7:Name>
    </mpeg7:FileFormat>
    <mpeg7:FileSize>660478608</mpeg7:FileSize>
    <mpeg7:VisualCoding>
      <mpeg7:Format>
        <mpeg7:Name>MPEG-1 Video</mpeg7:Name>
      </mpeg7:Format>
      <mpeg7:Pixel mpeg7:aspectRatio="0.75" mpeg7:bitsPer="8" />
      <mpeg7:Frame mpeg7:height="288" mpeg7:width="352" mpeg7:rate="25" 7>
    </mpeg7:VisualCoding>
  </MediaFormat>

  <TemporalDecomposition>
    <mpeg7:VideoSegment mpeg7:id="segment1" >
      <mpeg7:MediaTime>
        <mpeg7:MediaTimePoint>T00:00:00</mpeg7:MediaTimePoint>
        <mpeg7:MediaDuration>PT15M00S</mpeg7:MediaDuration>
      </mpeg7:MediaTime>
      <mpeg7:TextAnnotation>
        <mpeg7:FreeTextAnnotation>Arafat Interview</mpeg7:FreeTextAnnotation>
      </mpeg7:TextAnnotation>
    </mpeg7:VideoSegment>
    <mpeg7:VideoSegment mpeg7:id="segment2">
      <mpeg7:MediaTime>
        <mpeg7:MediaTimePoint>T00:15:00</mpeg7:MediaTimePoint>
        <mpeg7:MediaDuration>PT15M00S</mpeg7:MediaDuration>
      </mpeg7:MediaTime>
      <mpeg7:TextAnnotation>
        <mpeg7:FreeTextAnnotation>Anti-War Protests</mpeg7:FreeTextAnnotation>
      </mpeg7:TextAnnotation>
    </mpeg7:VideoSegment>
  </TemporalDecomposition>

</MultimediaDescription>
```

All'interno dell'istanza sono presenti tutti gli elementi utilizzabili per descrivere in maniera esaustiva le caratteristiche bibliografiche e fisiche della presentazione multimediale.

4.4 Conclusioni

In questo capitolo è stato individuato un modello per la rappresentazione dei contenuti multimediali all'interno di MUM che sarà utilizzato

nella fase di sviluppo del progetto di tesi.

Nel prossimo capitolo si entrerà in dettaglio nell'analisi del nuovo servizio di gestione dei contenuti multimediali, analizzandone i requisiti e ricavando una prima descrizione del sistema.

Capitolo 5

Analisi del sistema di gestione dei contenuti multimediali

In questo, e nei prossimi capitoli, si entrerà nei dettagli dell'argomento di questa tesi: lo scopo principale è lo sviluppo di un nuovo componente all'interno di MUM che realizzi la gestione dei metadati dei contenuti multimediali. Questo componente avrà il compito di:

- fornire una rappresentazione dei metadati che sia il più possibile aderente ai moderni standard internazionali;
- creare dei meccanismi di caching distribuito dei metadati;
- gestire le politiche di accesso alle cache che contengono i metadati.

Realizzando queste tre funzionalità, questo componente sarà in grado di far raggiungere all'applicazione multimediale in cui andrà ad inserirsi due obiettivi di fondamentale importanza:

- l'apertura dell'intero sistema nei confronti di altre applicazioni multimediali. Utilizzando per la rappresentazione dei contenuti multimediali le indicazioni fornite da uno o più standard internazionali, si fa in modo che l'applicazione possa scambiare delle informazioni sui contenuti multimediali con altre applicazioni che utilizzano la stessa rappresentazione.

- l'incremento della Qualità di Servizio fornita dal sistema all'utente finale. La Qualità di Servizio, come già messo in evidenza nel capitolo sulle metodologie di caching, subisce un forte incremento se vengono utilizzati dei meccanismi di caching.

- la realizzazione di differenti livelli di Qualità di Servizio. Utilizzando, infatti, delle politiche per l'accesso alle cache che contengono i metadati il sistema può offrire differenti livelli di Qualità di Servizio in base alla politica utilizzata all'interno del sistema, riuscendo a scindere la Qualità di Servizio offerta all'utilizzatore dell'applicazione dalla Qualità di Servizio offerta all'interno del sistema stesso.

Seguendo scrupolosamente i moderni principi dell'Ingegneria del Software [PRE00], che garantiscono la corretta realizzazione di applicazioni o componenti software complessi, il processo di sviluppo del componente sarà suddiviso in quattro fasi ben distinte: analisi, progettazione, implementazione e testing.

Questo capitolo tratta la fase di analisi: dopo aver descritto i requisiti del modulo, si procederà alla loro analisi; si riuscirà, in questo modo, ad ottenere una prima rappresentazione del modulo, per mezzo di diagrammi UML [UML]. Questa rappresentazione sarà successivamente ampliata e arricchita nelle fasi di progetto e di implementazione, riuscendo, in questo modo, ad ottenere la struttura finale del componente.

5.1 Requisiti

Si vuole sviluppare un nuovo componente per la gestione dei contenuti multimediali all'interno di MUM. I contenuti multimediali, detti anche **presentazioni multimediali**, sono di natura e di qualità diversa.

La presentazione multimediale, che è caratterizzata da un **titolo** e da una breve**descrizione**, che la identificano all'interno del sistema, e dal'indicazione della **qualità**, intesa come qualità complessiva della presentazione.

La presentazione multimediale è un aggregato di uno o più **oggetti multimediali**.

Il singolo oggetto multimediale è un metadato che descrive un dato multimediale presente all'interno del sistema; ogni oggetto multimediale rappresenta un solo tipo di media, quindi può essere solo di tipo audio, video, o altro.

Il singolo oggetto multimediale contiene tutte le informazioni relative al dato multimediale al quale si riferisce. Queste informazioni dovranno essere descritte attraverso lo schema per la descrizione delle presentazioni multimediali realizzato nel precedente capitolo. Vengono perciò suddivise in due categorie: le informazioni di tipo bibliografico e le informazioni di tipo fisico. Le prime rappresentano il titolo, l'autore, il soggetto, la descrizione e tutto ciò che non riguarda il dato multimediale a livello fisico. Le seconde rappresentano gli aspetti temporali, spaziali o spazio-temporali e le caratteristiche visuali o auditive del singolo oggetto multimediale.

Il componente dovrà fornire all'interno del sistema un servizio di caching dei metadati dei contenuti multimediali.

Il sistema di caching dei metadati dovrà essere realizzato tenendo in considerazione l'analisi delle differenti metodologie di caching effettuata nel capitolo sulle metodologie di caching: dall'analisi svolta precedentemente, infatti, si può ottenere un modello su cui basarsi per la realizzazione del sistema di caching che sia il più possibile rispondente alle esigenze dell'intero sistema.

In particolare, l'architettura del sistema dovrà essere di tipo distribuito, cioè in ogni default place di ogni dominio dovrà essere presente una cache con il relativo gestore. Questo perché, in base alle considerazioni effettuate nel capitolo sulle metodologie di caching, un'architettura di caching distribuita è più indicata di una centralizzata in un sistema altamente distribuito.

Il protocollo utilizzato dai gestori delle cache per comunicare con gli altri gestori dovrà essere di tipo gerarchico, dal momento che per dati di piccole dimensioni, come i metadati, questo tipo di protocollo è più efficiente di uno distribuito o ibrido.

Le tre funzionalità principali che i gestori dovranno offrire sono il reperimento, l'inserimento e l'eliminazione di un metadato. Potranno poi fornire funzionalità aggiuntive, come, ad esempio, l'interrogazione sull'esistenza o meno di un metadato all'interno della cache, o altro.

All'interno del sistema di gestione dei contenuti multimediali, oltre ai gestori delle cache dei metadati delle presentazioni multimediali, dovranno essere presenti altri gestori incaricati di regolamentare l'accesso degli agenti alla cache dei contenuti multimediali tramite l'utilizzo di particolari politiche.

Nel caso in cui un qualsiasi agente, che potrebbe essere ad esempio un Proxy in un Service Path, abbia la necessità di fare del caching di una presentazione multimediale, il gestore delle politiche dovrà farsi carico di analizzare la richiesta e, in base alla particolare politica adottata all'interno del sistema, decidere se permettere oppure negare l'inserimento della presentazione multimediale all'interno del place, nonché del relativo metadato che descrive la presentazione.

Le politiche attuate dai gestori possono essere differenti e vengono decise all'atto di inizializzazione del sistema. Dovrà, perciò, essere realizzato un meccanismo per l'inserimento di nuove politiche, per la modifica di politiche già esistenti o per la loro eliminazione, nonché per la configurazione del sistema all'atto della sua inizializzazione.

La definizione delle politiche è basata su due parametri fondamentali: lo spazio occupato dalle presentazioni multimediali e la velocità di reperimento di una presentazione multimediale all'atto della sua richiesta.

Se si volesse garantire, ad esempio, una velocità di risposta elevata all'atto di richiesta di una presentazione, per riuscire a garantire una maggiore Qualità di Servizio all'utente, il gestore potrebbe acconsentire a ogni richiesta di inserimento delle presentazioni.

Parimenti, se si volesse salvaguardare lo spazio su disco, incrementando in questo modo la Qualità di Servizio interna, potrebbe essere necessario negare l'inserimento di una presentazione se quest'ultima si trovasse, ad esempio, in un sotto-albero dei dominii adiacente a quello in cui viene effettuata la richiesta.

In generale, una generica politica di accesso dovrà basarsi su considerazioni legate alla distanza tra il place in cui viene inoltrata la richiesta di inserimento e quella in cui si trova già il dato.

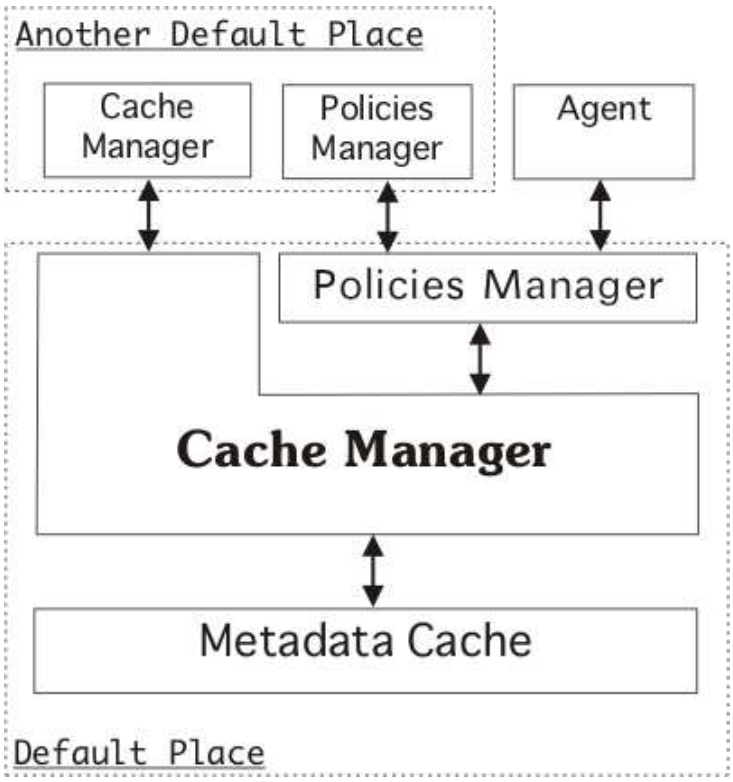

Figura 5.1: Struttura del gestore dei metadati all'interno di un place di default.

In figura 5.1, è mostrata la struttura di un gestore dei metadati dei contenuti multimediali. Il gestore della cache dei contenuti multimediali, il *Cache Manager*, ha il compito di interagire con la cache dei metadati presente all'interno del place di default, con i Cache Manager presenti negli altri place di default e con il gestore delle politiche di accesso alla cache, il *Policies Manager*. Quest'ultimo, si inserisce fra gli Agenti e il Cache Manager, regolamentando l'accesso alla cache dei metadati: deve perciò riuscire a comunicare con i Policies Manager presenti negli altri place di default per applicare le politiche di accesso.

5.2 Analisi dei requisiti

In questa sezione verrà effettuata l'analisi dei requisiti appena presentati: si riuscirà, in questo modo, ad ottenere una descrizione delle parti che compongono il sistema e delle loro reciproche interazioni.

Per realizzare ciò, verrà utilizzato UML [UML], un linguaggio di modellazione di tipo visuale utilizzato per visualizzare, costruire e documentare i sistemi orientati agli oggetti.

L'analisi dei requisiti verrà suddivisa in due parti distinte: una per la rappresentazione dei contenuti multimediali e l'altra per il sistema di gestione dei contenuti multimediali.

Dal momento che l'analisi si propone di fornire una prima semplice descrizione del sistema, nei diagrammi UML verranno messi in evidenza solamente i metodi fondamentali e tutti gli attributi saranno di tipo *String*: questo perché l'analisi fornisce una rappresentazione il più possibile indipendente dal contesto in cui andrà a inserirsi il sistema. Sarà solo in fase di progetto e di implementazione che verranno definiti tutti i dettagli del sistema.

5.2.1 Rappresentazione dei contenuti multimediali

Dai requisiti precedentemente presentati si possono ricavare le seguenti classi per la rappresentazione dei metadati dei contenuti multimediali:

- **PresentationMetadata**: rappresenta una presentazione multimediale. Essa è caratterizzata dal titolo, dalla descrizione e dalla qualità;
- **ObjectMetadata**: rappresenta un singolo oggetto multimediale. Esso contiene le indicazioni delle caratteristiche di un oggetto multimediale:
  - le caratteristiche bibliografiche: title, creator, subject, description, identifier, language, rights;
  - le caratteristiche fisiche: mediaLocator, mediaTime, mediaFormat, temporalDecomposition.

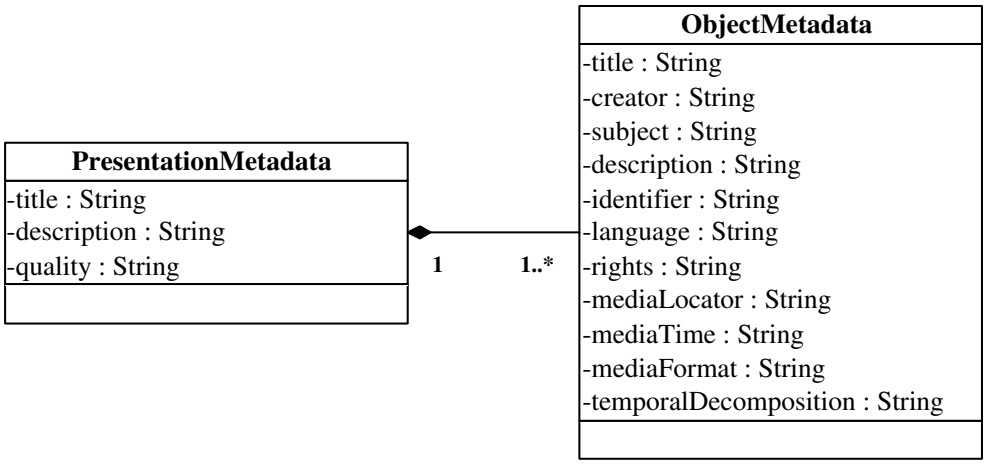

Figura 5.2: Diagramma delle classi per i metadati.

In figura 5.2, sono rappresentate le due classi appena presentate. Esse sono legate da una relazione di composizione: una presentazione multimediale è un aggregato di uno o più oggetti multimediali.

5.2.2 Sistema di caching per i metadati dei contenuti multimediali

Dai requisiti precedentemente presentati si possono ricavare le seguenti classi per il sistema di gestione dei metadati dei contenuti multimediali:

- **MetadataCache**: è l'astrazione della cache in cui vengono memorizzati i metadati. Fornisce le funzionalità di reperimento, inserimento ed eliminazione al gestore;
- **CacheManager**: rappresenta il gestore della cache dei metadati. La sua funzione è quella di interagire con la cache dei metadati e di fornire le tre funzionalità al gestore delle politiche ed ai gestori dei metadati presenti negli altri place;
- **PoliciesManager**: fornisce un'interfaccia di accesso al sistema di gestione dei metadati per un qualsiasi agente. L'inserimento di un metadato è vincolato alla determinata politica utilizzata dal gestore;

- **CacheProtocol**: rappresenta il protocollo utilizzato dai gestori della cache all'interno di tutto il sistema;

- **PoliciesProtocol**: rappresenta il protocollo utilizzato dai gestori delle politiche all'interno di tutto il sistema.

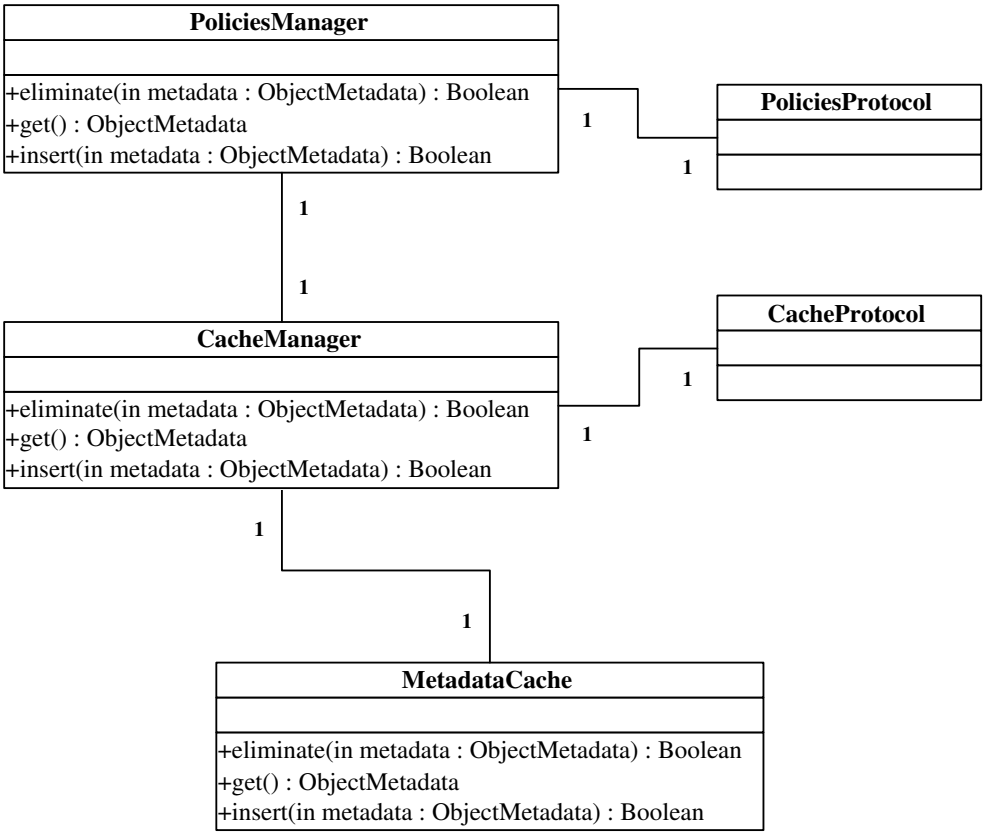

Figura 5.3: Diagramma delle classi per il sistema di caching dei metadati.

In figura 5.3, è mostrato il diagramma delle classi per il sistema di gestione dei metadati: il gestore della cache fa uso di un protocollo per lo scambio di informazioni con gli altri gestori presenti negli altri place all'interno del sistema e accede alla cache dei metadati. Il gestore delle politiche deve poter interagire con il gestore della cache e, tramite un protocollo, con gli altri gestori delle politiche presenti negli altri place all'interno del sistema.

Protocollo per il caching dei metadati dei contenuti multimediali

Il protocollo per il caching dei metadati dei contenuti multimediali si occupa del modo in cui i gestori presenti in ogni default place dell'albero dei dominii scambiano e memorizzano informazioni sui metadati.

Fondamentalmente, esso riguarda il comportamento di un gestore nei confronti delle tre funzionalità che offre: reperimento, inserimento ed eliminazione.

- **Reperimento** di un metadato. All'atto di richiesta di un metadato, il gestore controlla se esso è presente nella sua cache. Se il metadato è presente, allora ne restituisce una copia al richiedente. Se il metadato non è presente, inoltra la richiesta al gestore del place padre. Il gestore del place padre effettua le stesse operazioni del figlio, e se neanche lui trova il metadato, si rivolge a sua volta al suo place padre. La richiesta, dunque, viene inoltrata verso la radice dell'albero dei domini, finché un gestore non riesce a soddisfarla. Se il metadato non è presente neanche nella cache del nodo radice, allora significa che il metadato non è presente all'interno del sistema.

- **Inserimento** di un metadato. All'atto di richiesta dell'inserimento di un metadato, il gestore deve effettuare tre distinte operazioni:

  1. informare il nodo radice dell'inserimento del metadato, in maniera tale da renderlo immediatamente disponibile in tutto il sistema;
  2. inserire il metadato nella propria cache;
  3. inviare il metadato al gestore presente nel place padre, il quale lo memorizzerà e lo invierà al gestore dei metadati presente nel proprio padre, e così via fino ad arrivare al place radice. In questo modo il metadato verrà memorizzarlo in tutte le cache intermedie presenti tra il place in cui si richiede l'inserimento e il place radice.

- **Eliminazione** di un metadato. All'atto di richiesta di eliminazione di un metadato, il gestore inoltra la richiesta al gestore presente nel place radice della gerarchia. Quest'ultimo, propaga la richiesta di eliminazione a tutti i suoi figli, e ciascun figlio la propaga ai rispettivi figli, fino a giungere ai place che rappresentano le foglie dell'albero dei domini. In questo modo, si riesce ad eliminare il metadato da ogni cache all'interno del sistema che lo conteneva.

Protocollo di gestione delle politiche di accesso al sistema di caching dei metadati

Il protocollo di gestione delle politiche di accesso al sistema di caching dei metadati specifica le operazioni che i gestori delle politiche devono compiere per coordinarsi e per applicare le proprie politiche.

Si basa sulla suddivisione dell'albero formato dai dominii in più sotto-alberi. Ogni sotto-albero, che può essere indicato più semplicemente con il termine *località*, è definito a partire da ogni singolo dominio e comprende il dominio stesso e tutti i dominii che da esso discendono fino ai dominii che costituiscono le foglie dell'albero.

L'albero dei dominii è inoltre suddiviso orizzontalmente, in maniera tale da inserire ogni località in un livello.

In figura 5.4, è riportata la suddivisione che si creerebbe nel caso di un ipotetico albero dei dominii.

Come si può vedere in figura, la località associata ad esempio al dominio B2 è quella formata dal dominio B2 stesso, più i domini C1, C2, D1 e D2 ed è una località di livello 1; la località associata al dominio C1 è quella formata dal dominio C1 stesso più i domini D1 e D2 ed è una località di livello 3.

La singola politica di accesso si basa sulla distanza tra il place in cui si richiede l'inserimento della presentazione multimediale, e del relativo metadato, e il place più vicino in cui essa è già presente. Quando un gestore riceve una richiesta di inserimento dovrà richiedere le informazioni sul metadato agli altri gestori e poi applicare la sua politica in base alle in-

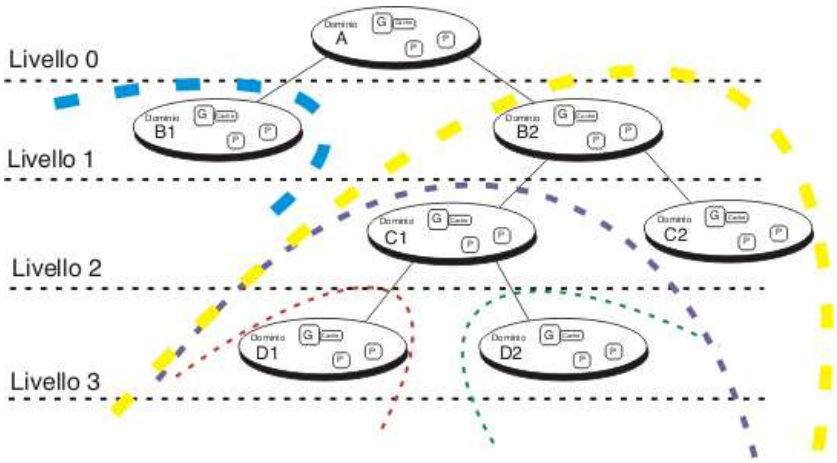

Figura 5.4: Suddivisione in località e in livelli dell'albero dei dominii.

formazioni ricevute dagli altri gestori. In particolare, esso dovrà rivolgersi al gestore presente nel livello superiore e ai gestori presenti in eventuali place dello stesso livello: anche questi gestori, se non sono in grado di fornire delle informazioni sulla presentazione, dovranno rivolgersi ad altri gestori, fino a giungere al gestore presente nel place radice della gerarchia.

5.3 Conclusioni

Dall'analisi effettuata in questo capitolo si è riusciti ad ottenere una prima rappresentazione del sistema di gestione dei contenuti multimediali e dei loro metadati.

L'argomento del prossimo capitolo sarà la progettazione: si entrerà, dunque, ancora più nel dettaglio dello sviluppo del sistema, suddividendo e arricchendo con ulteriori particolari la descrizione ottenuta in questo capitolo, per poi dare finalmente al tutto una struttura definitiva nel capitolo dedicato alle scelte implementative.

Capitolo 6

Progettazione del sistema di gestione dei contenuti multimediali

In questo capitolo si procederà nella seconda fase dello sviluppo del sistema di gestione dei contenuti multimediali: la progettazione.

La rappresentazione del sistema per la gestione dei contenuti multimediali descritta nel precedente capitolo sarà quindi arricchita con più dettagli e con più particolari dal momento che, argomenti meno rilevanti nell'analisi, saranno ora affrontati in profondità.

La fase di progettazione coinvolge l'organizzazione stratificata di package e classi, la gestione del riuso e dei componenti, la specifica delle funzionalità di ciascuna classe e i modelli di interazione tra le classi.

Cercando, come nella precedente fase di analisi, di rispettare i moderni principi dell'Ingegneria del Software, il progetto del componente sarà suddiviso in due parti fondamentali:

- il **progetto architetturale**, in cui vengono specificati i package che fanno parte del nuovo componente e vengono descritte le classi che fanno parte di ciascun package;

- il **progetto di dettaglio**, in cui vengono descritti i comportamenti e

le interazioni fra le classi che permettono al componente di raggiungere i requisiti specificati nel precedente capitolo.

Alla fine di questo capitolo, si riuscirà ad ottenere una descrizione molto dettagliata del sistema di gestione dei contenuti multimediali e si potrà passare, finalmente, alla fase di implementazione del nuovo componente.

6.1 Progetto architetturale

Il progetto architetturale fornisce una visione complessiva dell'architettura dell'intero sistema che si mantiene indipendente dal linguaggio di programmazione utilizzato nella fase di implementazione.

La rappresentazione delle parti che compongono il sistema è effettuata mediante l'utilizzo del concetto UML di **package**: un package non è altro che un contenitore di classi con uno o più obiettivi comuni. Tra i vari package che compongono il sistema possono esistere delle relazioni che mettono in evidenza di quali risorse hanno bisogno le classi di un particolare package per riuscire a svolgere le proprie funzionalità.

In figura 6.1, è mostrato il diagramma dei package che formano il sistema di gestione dei contenuti multimediali, con le relazioni che intercorrono fra essi.

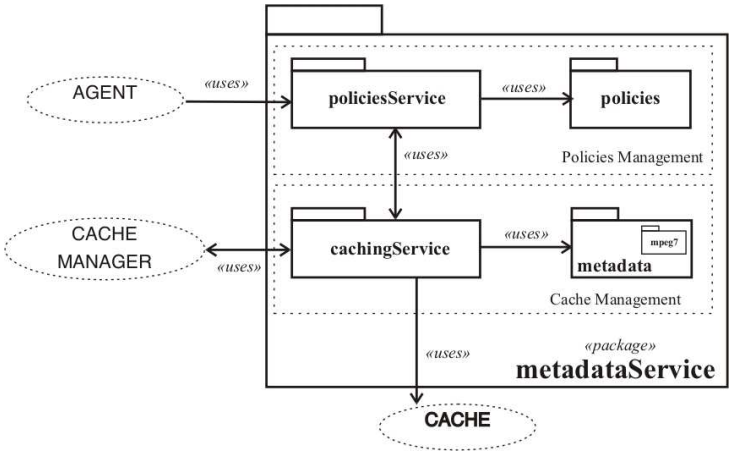

Figura 6.1: Diagramma dei package del sistema.

Nei prossimi paragrafi, verranno descritti singolarmente i package, le classi e le interfacce in essi contenute.

Si è scelto di procedere utilizzando la progettazione per interfacce, che permette agli oggetti del sistema di interagire tra di loro attraverso interfacce ben definite, nonostante non tutti i linguaggi di programmazione forniscano gli strumenti per implementarla: è sembrato tuttavia corretto utilizzarla dal momento che il componente andrà ad inserirsi in un ambiente realizzato secondo questo paradigma.

6.1.1 Package *metadataService*

Il package *metadataService* è il contenitore del sistema di gestione dei contenuti multimediali. In esso sono presenti tutti gli altri package che formano il sistema e le classi che rappresentano i gestori del sistema.

- ***IMetadataServiceManager***: è l'interfaccia che deve essere implementata da un qualsiasi gestore del sistema di gestione dei metadati. In essa sono presenti le dichiarazioni dei metodi che il gestore dovrà fornire per permettere l'utilizzo delle sue funzionalità agli altri componenti dell'applicazione e di un metodo per la verifica dell'avvenuta creazione del gestore;

- ***ClientMetadataServiceManager***: rappresenta il lato client del gestore del sistema di gestione dei metadati. Viene istanziato nei place che non hanno funzionalità di server, ossia non sono place di default, e implementa i metodi dell'interfaccia IMetadataServiceManager;

- ***ServerMetadataServiceManager***: rappresenta il lato server del gestore del sistema di gestione dei metadati. È realizzato come classe astratta, dal momento che esistono all'interno del sistema due tipologie di gestori server: i gestori del place di default radice della gerarchia, e tutti gli altri place di default all'interno della gerarchia. Contiene il riferimento ai due gestori presenti all'interno del sistema: il gestore del sistema di caching e il gestore delle politiche di accesso alla cache;

- ***RootMetadataServiceManager***: è il gestore del sistema di gestione dei metadati che viene istanziato nel place di default radice. È una sotto-classe della classe astratta ServerMetadataServicemanager;
- ***GenericMetadataServiceManager***: è il gestore del sistema di gestione dei metadati che viene istanziato in tutti i place di default presenti nella gerarchia dei dominii, ad eccezione del place di default radice.

In figura 6.2, è mostrato il diagramma delle classi per il package metadataService.

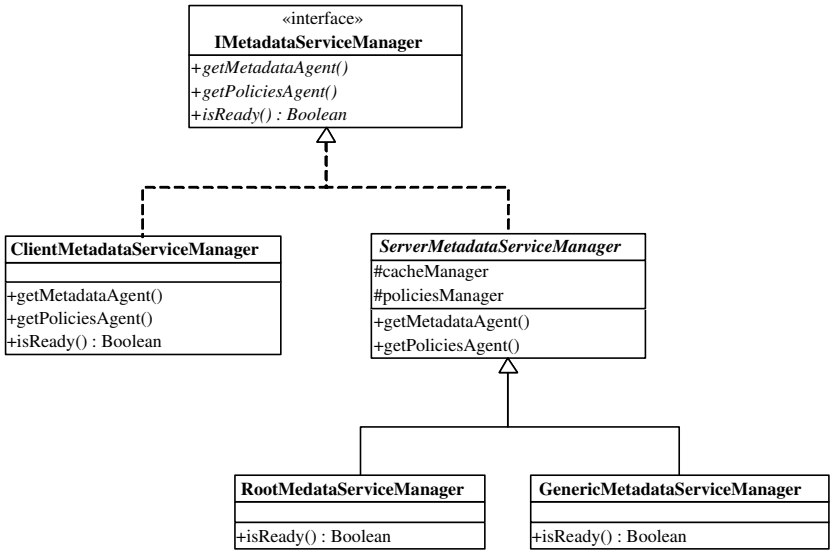

Figura 6.2: Diagramma delle classi per il package metadataService.

6.1.2 Package *metadataService.cachingService*

Il package *cachingService* contiene le classi che rappresentano il sistema di caching dei metadati.

- ***IMetadataCacheManager***: è l'interfaccia che deve essere implementata da qualsiasi gestore del caching dei metadati;
- ***ClientMetadataCacheManager***: rappresenta il lato client del sistema di caching. È utilizzata da un gestore nel caso in cui volesse comunicare con gestore presente in un altro place;

- ***ServerMetadataCacheManager***: rappresenta il lato server del sistema di caching. È realizzata come classe astratta poiché sono presenti all'interno del sistema due tipi differenti di gestori. Contiene il riferimento ad un oggetto che permette di accedere alla cache dei metadati;

- ***RootMetadataCacheManager***: è il gestore della cache dei metadati che si trova nel place default radice della gerarchia. Fornisce le funzionalità di inserimento, eliminazione e reperimento di una presentazione multimediale;

- ***GenericMetadataCacheManager***: è il gestore della cache dei metadati che si trova in tutti i place di default, tranne che in quello radice. Fornisce le funzionalità di inserimento, eliminazione e reperimento di una presentazione multimediale;

- ***IPresentationMetadataRepository***: è l'interfaccia per l'accesso alla cache dei metadati;

- ***PresentationMetadataRepositoryXML***: rappresenta la cache dei metadati, memorizzati in formato XML;

In figura 6.3, è mostrato il diagramma delle classi per il package cachingService.

6.1.3 Package *metadataService.metadata*

Il package *metadata*, all'interno del package metadataService, contiene le classi utilizzate per descrivere le presentazioni multimediali presenti all'interno del sistema. Esso si compone delle seguenti classi:

- ***IMetadata***: è l'interfaccia per il generico oggetto multimediale. Specifica i metodi per ottenere le caratteristiche fondamentali di un oggetto multimediale;

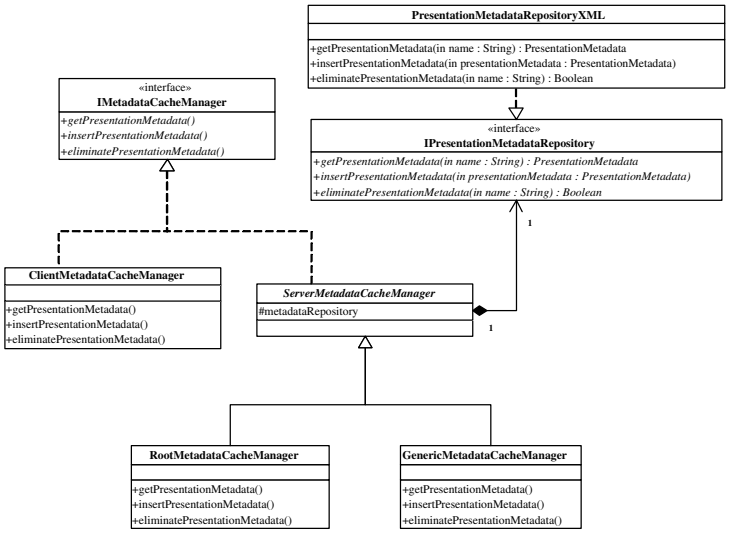

Figura 6.3: Diagramma delle classi per il package cachingService.

- ***Metadata***: rappresenta il singolo oggetto multimediale, che implementa l'interfaccia IMetadata. Partendo dalla descrizione di un oggetto multimediale ottenuta nella fase di analisi, vengono definiti i tipi degli attributi della classe e specificati i metodi per ottenere le informazioni contenute all'interno dell'oggetto multimediale;

- ***IPresentationMetadata***: è l'interfaccia per la generica presentazione multimediale. In essa vengono dichiarati i metodi per accedere agli attributi contenuti all'interno di una presentazione multimediale;

- ***PresentationMetadata***: rappresenta la singola presentazione multimediale. Essa contiene gli attributi che descrivono una presentazione multimediale e una lista di oggetti multimediali che la compongono;

- ***IMetadataList*** e ***IPresentationMetadataList***: rappresentano, rispettivamente, l'interfaccia per una generica lista di oggetti multimediali e l'interfaccia per una generica lista di presentazioni multimediali;

- ***MetadataList*** e ***PresentationMetadataList***: rappresentano, rispettivamente, un contenitore di oggetti multimediali e un contenitore

di presentazioni multimediali. Forniscono le funzionalità per l'aggiunta, la rimozione, il reperimento, la verifica dell'esistenza di un oggetto multimediale o di una presentazione multimediale;

In figura 6.4, è mostrato il diagramma delle classi per il package metadata.

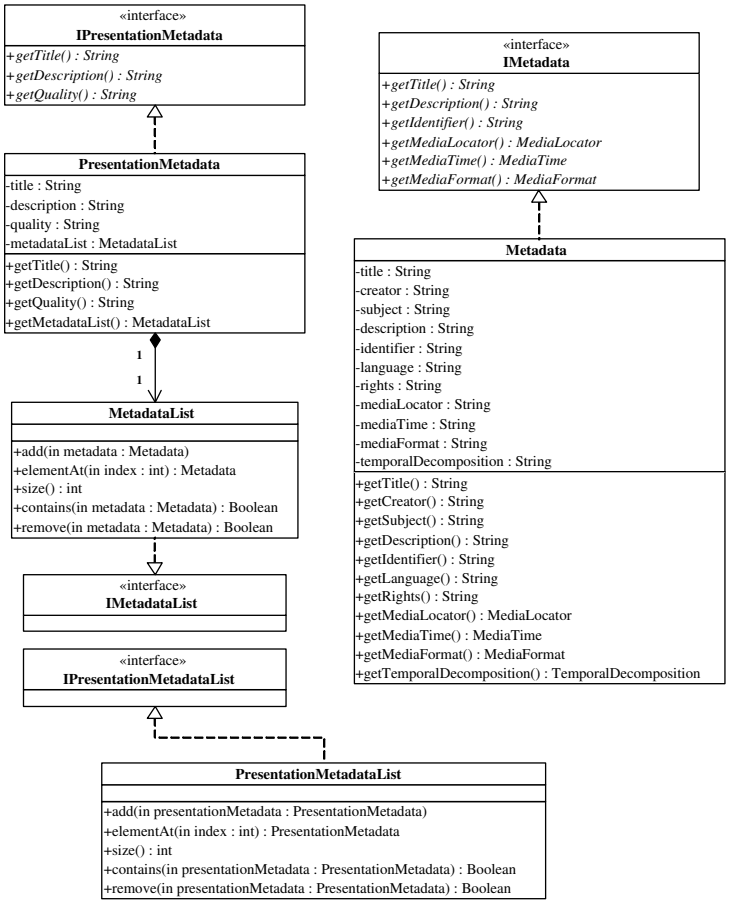

Figura 6.4: Diagramma delle classi per il package metadata.

Package *metadataService.metadata.mpeg7*

Il package *mpeg7* si trova all'interno del package metadata e contiene le classi utilizzate per la descrizione delle caratteristiche dello standard MPEG-7 utilizzate per la descrizione del singolo oggetto multimediale.

Come evidenziato nella fase di analisi, gli attributi utilizzati per la descrizione delle caratteristiche di un oggetto multimediale sono rappresentati dalle quatto classi seguenti:

- ***MediaLocator***: contiene la locazione dell'oggetto multimediale e un metodo per accedere ad essa;

- ***MediaTime***: contiene le informazioni temporali dell'oggetto multimediale, il tempo iniziale e la durata, nonché i metodi per poter accedere a questi attributi;

- ***MediaFormat***: contiene informazioni sul formato dell'oggetto multimediale. In particolare, esso è formato da: un attributo di tipo **Content** per la descrizione del contenuto, un attributo di tipo **Medium** per la descrizione del supporto fisico sul quale è memorizzato, un attributo di tipo **FileFormat** per la descrizione del formato del file, la dimensione dell'oggetto multimediale e un attributo di tipo **VisualCoding**. Quest'ultimo rappresenta le caratteristiche visuali dell'oggetto multimediale: è formato da un attributo di tipo **Pixel**, un attributo di tipo **Frame** e uno di tipo **Format**. La classe MediaFormat, specifica, inoltre, i metodi per poter accedere a tutti gli attributi che contiene;

- ***TemporalDecomposition***: contiene le informazioni sulle suddivisioni temporali di un oggetto multimediale. Esso è formato da uno o più segmenti, rappresentati dalla classe **VideoSegment**: ogni VideoSegment contiene le informazioni temporali, rappresentate da un oggetto di tipo MediaTime, e una descrizione del segmento. La classe TemporalDecomposition contiene i metodi per accedere ai suoi attributi.

In figura 6.5, è mostrato il diagramma delle classi per il package mpeg7.

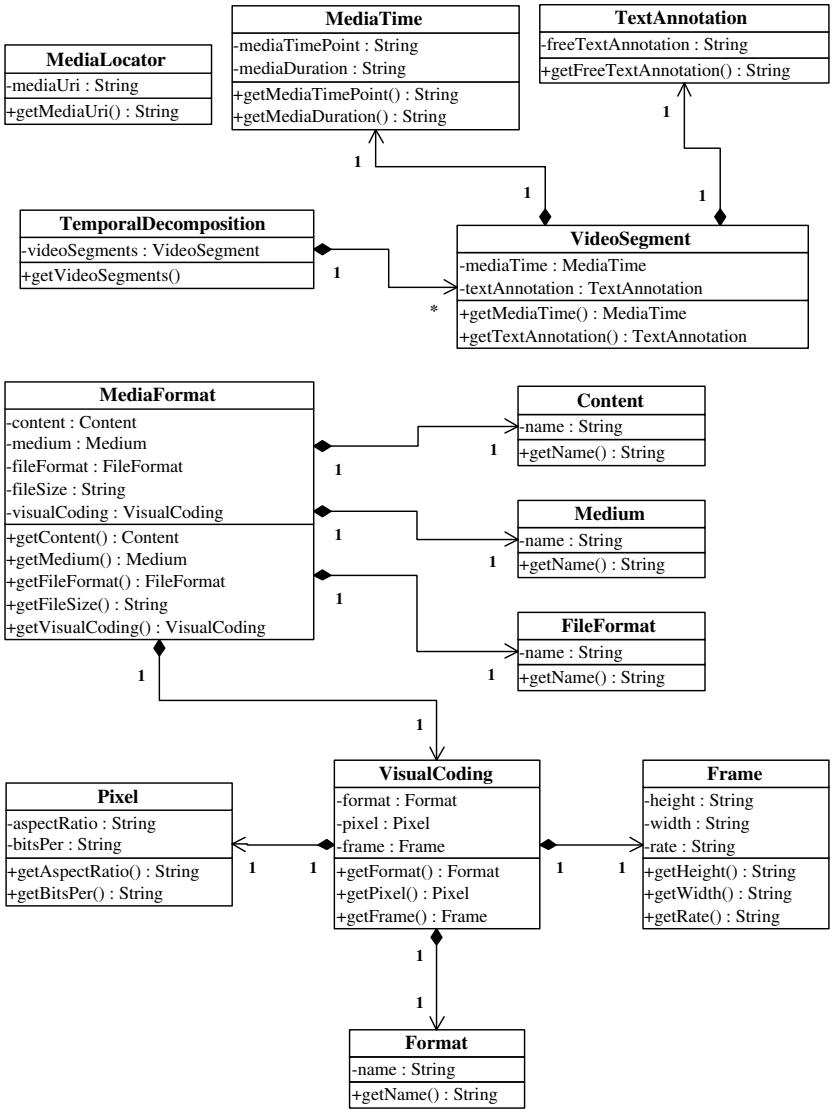

Figura 6.5: Diagramma delle classi per il package mpeg7.

6.1.4 Package *metadataService.policies*

Il package *policies* contiene le classi che rappresentano la politica di accesso alla cache dei metadati utilizzata dal gestore delle politiche.

- ***IPolicy***: è l'interfaccia di una generica politica. In essa vengono dichiarati i metodi per accedere al nome della politica e alla distanza, utilizzata nel protocollo di gestione dell'accesso alla cache;

- ***Policy***: rappresenta la singola politica. Contiene un nome e una distanza e implementa i metodi per accedere ai suoi attributi;
- ***IPoliciesList***: rappresenta l'interfaccia per una generica lista di politiche;
- ***PoliciesList***: rappresenta un contenitore di politiche. Fornisce le funzionalità per l'aggiunta, la rimozione, il reperimento, la verifica dell'esistenza di una politica;

In figura 6.6, è mostrato il diagramma delle classi per il package policies.

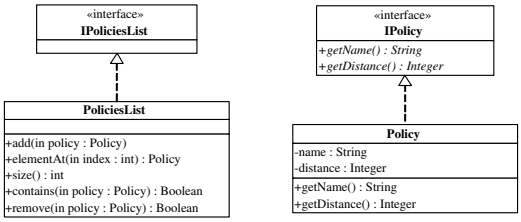

Figura 6.6: Diagramma delle classi per il package policies.

6.1.5 Package *metadataService.policiesService*

Il package *policiesService* contiene le classi che rappresentano il sistema di gestione delle politiche di accesso alla cache dei metadati.

- ***IPoliciesManager***: è l'interfaccia che deve essere implementata da qualsiasi gestore delle politiche di accesso alla cache dei metadati;
- ***ClientPoliciesManager***: rappresenta il lato client del sistema di gestione delle politiche di accesso al sistema. È utilizzata da un gestore nel caso in cui volesse comunicare con gestore presente in un altro place;
- ***ServerPoliciesManager***: rappresenta il lato server del sistema di gestione delle politiche di accesso al sistema. È realizzata come classe

astratta poiché sono presenti all'interno del sistema due tipi differenti di gestori. Contiene il riferimento ad un oggetto che permette di accedere al database delle politiche e un riferimento al gestore dei metadati presente nello stesso place;

- ***RootPoliciesManager***: è il gestore delle politiche di accesso alla cache dei metadati che si trova nel place default radice della gerarchia. Fornisce le funzionalità di inserimento, eliminazione e reperimento di una presentazione multimediale;
- ***GenericPoliciesManager***: è il gestore delle politiche di accesso alla cache dei metadati che si trova in tutti i place di default, tranne che in quello radice. Fornisce le funzionalità di inserimento, eliminazione e reperimento di una presentazione multimediale;
- ***IPoliciesRepository***: è l'interfaccia per l'accesso al database delle politiche;
- ***PoliciesRepositoryXML***: rappresenta il database delle politiche, memorizzate in formato XML;

In figura 6.7, è mostrato il diagramma delle classi per il package policiesService.

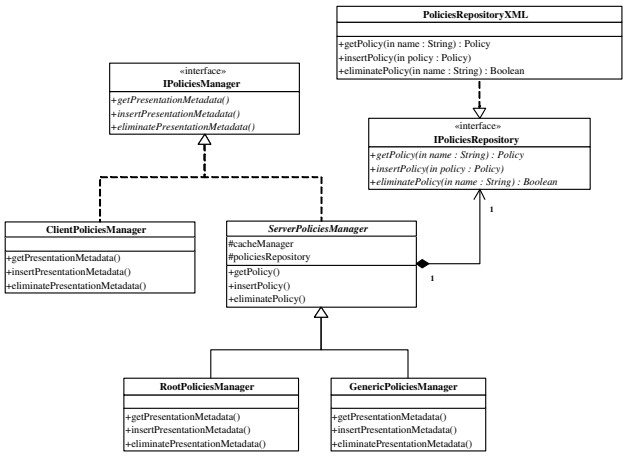

Figura 6.7: Diagramma delle classi per il package policesService.

6.2 Progetto di dettaglio

Dopo aver descritto i package e le classi che compongono il sistema di gestione dei contenuti multimediali, si passerà, in questo paragrafo, alla descrizione degli aspetti fondamentali di funzionamento di tutto il sistema.

Verrà presa in considerazione la comunicazione tra i gestori del sistema, il protocollo utilizzato per il sistema di caching e quello utilizzato per la gestione delle politiche di accesso al sistema.

Per descrivere il comportamento dei vari componenti, verranno utilizzati i diagrammi degli stati e dei diagrammi di sequenza, che forniscono una descrizione molto intuitiva e comprensibile del comportamento degli oggetti e delle loro reciproche interazioni.

In figura 6.8, sono mostrati gli elementi che costituiscono un diagramma degli stati: lo **stato** rappresenta una situazione, che si verifica durante la vita di un oggetto, nella quale l'oggetto esegue certe attività o aspetta il verificarsi di un evento; la **transazione** rappresenta il cambiamento dello stato dell'oggetto; ogni transazione può essere generata da un **evento**, il quale può verificarsi o all'interno dell'oggetto, per le operazioni effettuate dall'oggetto, oppure può provenire dall'esterno dell'oggetto.

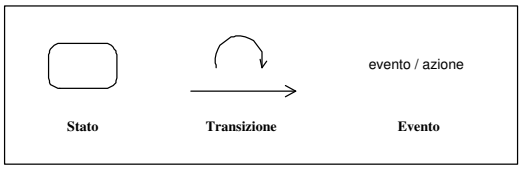

Figura 6.8: Elementi di un diagramma degli stati.

In figura 6.9, sono mostrati alcuni degli elementi che compongono un diagramma di sequenza: la **linea di vita di un oggetto** rappresenta la creazione e l'evoluzione nel tempo di un oggetto; esistono due tipi di interazioni tra gli oggetti, che vengono indicate tramite delle frecce: le frecce continue rappresentano l'**invocazione** di un metodo o di una funzione di un oggetto, le frecce tratteggiate rappresentano la **risposta** di un oggetto una volta che esso ha terminato l'esecuzione di un metodo.

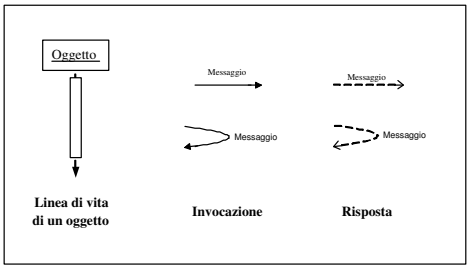

Figura 6.9: Elementi di un diagramma di sequenza.

6.2.1 Comunicazione tra i gestori

È di fondamentale importanza che i gestori del sistema di gestione dei contenuti multimediali presenti all'interno del sistema possano scambiarsi dei messaggi riguardanti le presentazioni multimediali, per riuscire ad attuare i protocolli di caching e di gestione delle politiche di accesso.

Per questo motivo, in ogni place deve essere presente un gestore della connessione, il quale ha fondamentalmente due compiti: restare in attesa di messaggi da parte degli altri gestori e, una volta ricevuti, comunicarlo al gestore del sistema; inviare i messaggi agli altri gestori.

Il gestore della connessione va ad inserirsi nel package *metadataService*, ed è rappresentato dalla classe di nome **MetadataServiceConnectionMaker**. Nel digramma degli stati di figura 6.10, è mostrato il comportamento del gestore della connessione all'interno del sistema di gestione dei contenuti mutlimediali.

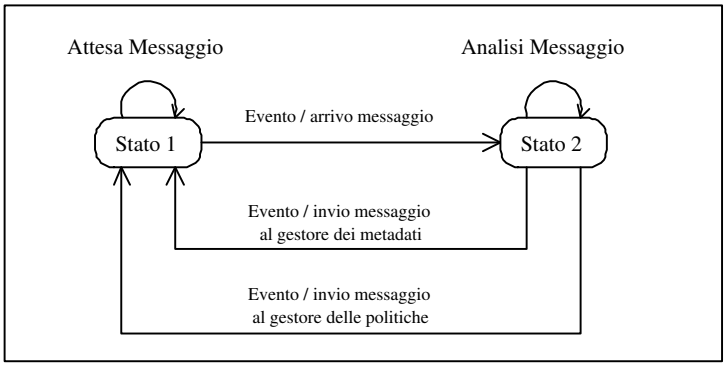

Figura 6.10: Diagramma degli stati del gestore della connessione.

6.2.2 Protocollo di caching dei metadati

Come già messo in evidenza precedentemente, il protocollo di caching dei metadati si occupa del modo in cui i gestori del sistema riescano a fornire le funzionalità di accesso al database dei metadati.

Il gestore, una volta avviato, resta in attesa di una richiesta. Questa può provenire dal gestore delle politiche oppure da un gestore dei metadati presente in un altro place. Anche un qualsiasi agente potrebbe aver bisogno di richiedere un servizio al gestore dei metadati: una sua richiesta però deve necessariamente passare attraverso il gestore delle politiche che, dopo averla analizzata, decide se inoltrarla o meno al gestore dei metadati.

Le operazioni fondamentali che possono essere richieste ad un gestore in attesa sono tre: il reperimento di un metadato, l'inserimento di un metadato e l'eliminazione di un metadato. Verrà ora descritto, attraverso una serie di diagrammi degli stati e di diagrammi di sequenza, il comportamento del gestore dei metadati nei confronti delle tre funzionalità fondamentali.

Richiesta di un metadato

In figura 6.11, è mostrato il diagramma degli stati del gestore dei metadati nel caso di richiesta di un metadato. Non appena il gestore riceve la richiesta di un metadato, si muove dallo stato di attesa (*stato 1*) e va in un altro stato (*stato 2*) in cui va a cercare il metadato richiesto nella sua cache: se lo trova, restituisce il metadato al richiedente e, finalmente, si riporta nello stato di attesa (*stato 1*); se il gestore non trova il metadato nella sua cache, inoltra la richiesta al gestore presente nel place padre, si mette in attesa di una risposta (*stato 3*) e, quando la riceve, la fornisce al richiedente e finalmente ritorna nello stato di attesa iniziale (*stato 1*).

Nel diagramma di sequenza di figura 6.12, sono messe in evidenza due richieste consecutive dello stesso metadato. La prima richiesta deve essere propagata fino al place di livello uno, in quanto il metadato non è presente in nessuna cache nel cammino che va dal gestore che ha ricevuto la richiesta al gestore presente nel place nel livello uno. Ad una successiva

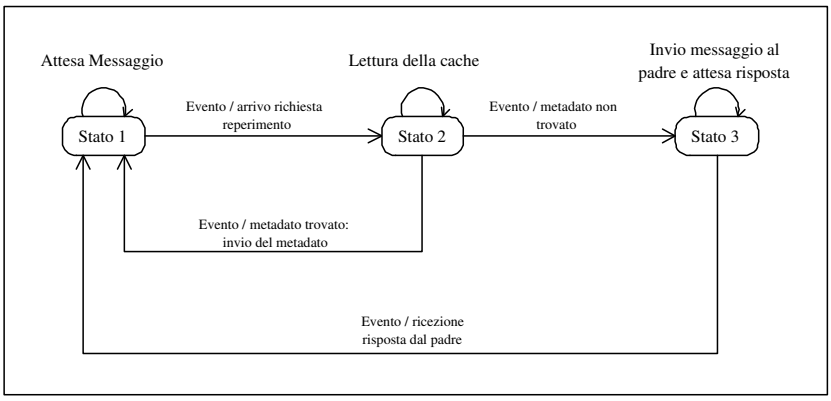

Figura 6.11: Diagramma degli stati del gestore dei metadati per la richiesta di un metadato.

richiesta dello stesso metadato, il gestore riesce a rispondere molto più velocemente in quanto ha memorizzato il metadato nella cache locale.

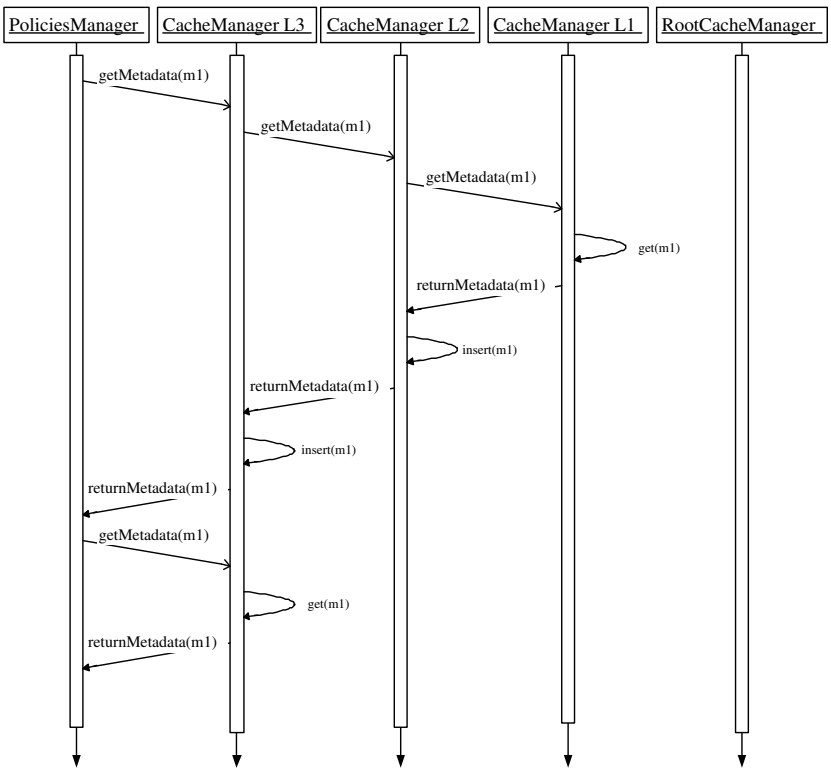

Figura 6.12: Diagramma di sequenza per la richiesta di un metadato.

Inserimento di un metadato

In figura 6.13, è mostrato il diagramma degli stati del gestore dei metadati nel caso di inserimento di un metadato. Quando il gestore in attesa riceve la richiesta di inserimento di un metadato, si muove dallo stato di attesa (*stato 1*) e si porta in uno stato in cui analizza la richiesta (*stato 2*). A questo punto il gestore può decidere di comportarsi in due modi differenti, prima di ritornare nuovamente nello stato iniziale (*1*). Nel caso in cui la richiesta provenga dal gestore delle policy, deve compiere tre operazioni: inviare la richiesta di inserimento al gestore dei metadati presente nel place radice, inserire il metadato nella propria cache, inviare la richiesta di inserimento al gestore presente nel place padre. Nel caso in cui, invece, la richiesta provenga da un altro gestore, le operazioni che il gestore dei metadati deve compiere sono solamente due: inserire il metadato nella propria cache e inviare la richiesta di inserimento al gestore presente nel place padre.

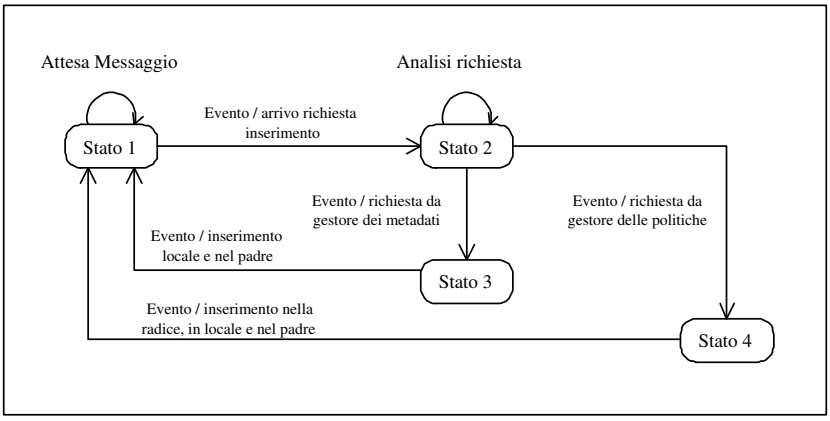

Figura 6.13: Diagramma degli stati del gestore dei metadati per l'inserimento di un metadato.

Nel diagramma di sequenza di figura 6.14, è mostrato quello che succede all'atto di inserimento di un metadato all'interno del sistema. La richiesta di inserimento si propaga dal gestore presente nel place in cui è stata ricevuta fino al gestore presente nel place radice e viene memorizzata da ogni gestore che si trova sul cammino.

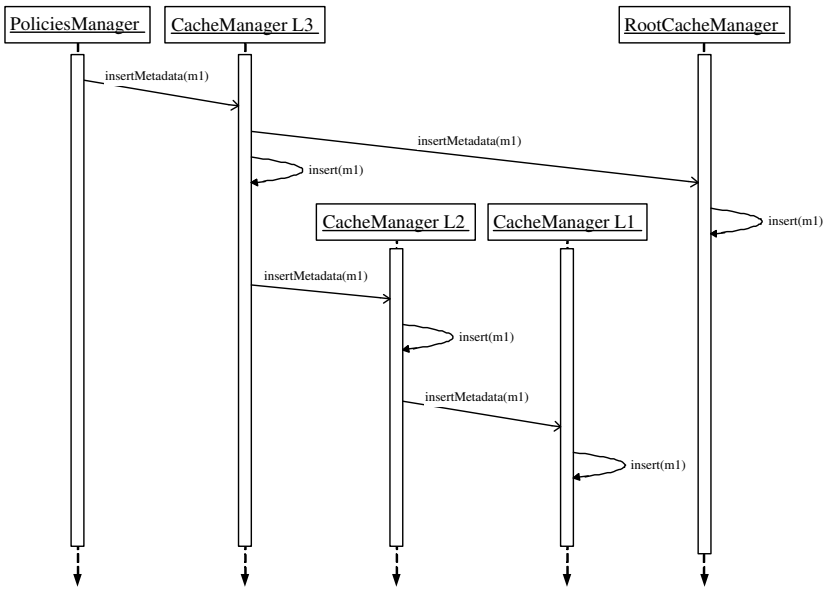

Figura 6.14: Diagramma di sequenza per l'inserimento di un metadato.

Eliminazione di un metadato

Prima di analizzare il comportamento del gestore dei metadati, bisogna mettere in evidenza un problema fondamentale, legato alla funzionalità di eliminazione, che nasce quando in due o più place vengono contemporaneamente richiesti il reperimento di un metadato e l'eliminazione della presentazione descritta da quel metadato. Il verificarsi di questa situazione costituisce una *corsa critica* per il sistema di caching e il suo comportamento è imprevedibile.

Potrebbe succedere, come è mostrato nel diagramma di sequenza della figura 6.15, che la richiesta di eliminazione giunga nel place in cui è stata fatta la richiesta del metadato prima che quest'ultima fosse stata eseguita e in questo caso il comportamento del sistema sarebbe corretto.

Potrebbe tuttavia capitare, come è mostrato nel diagramma di sequenza della figura 6.16, che il gestore dei metadati abbia già eseguito la richiesta di reperimento del metadato, e dunque che un agente stia già visualizzando la presentazione corrispondente a quel metadato, e che a quel punto arrivi la richiesta di eliminazione della presentazione che è correntemente

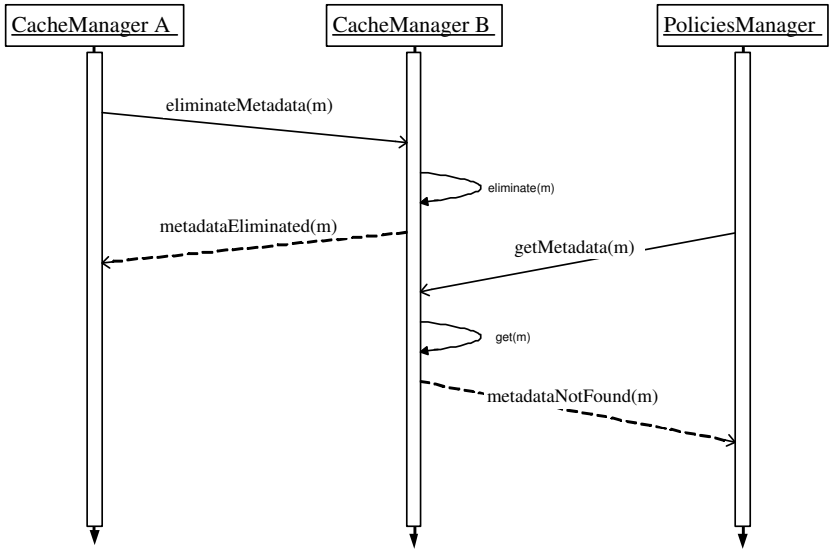

Figura 6.15: Diagramma di sequenza per la corsa critica: comportamento corretto.

in uso. La successiva eliminazione della presentazione dal sistema sottrarrebbe all'agente la risorsa che sta utilizzando, causando un'interruzione del servizio che si sta offrendo all'utente.

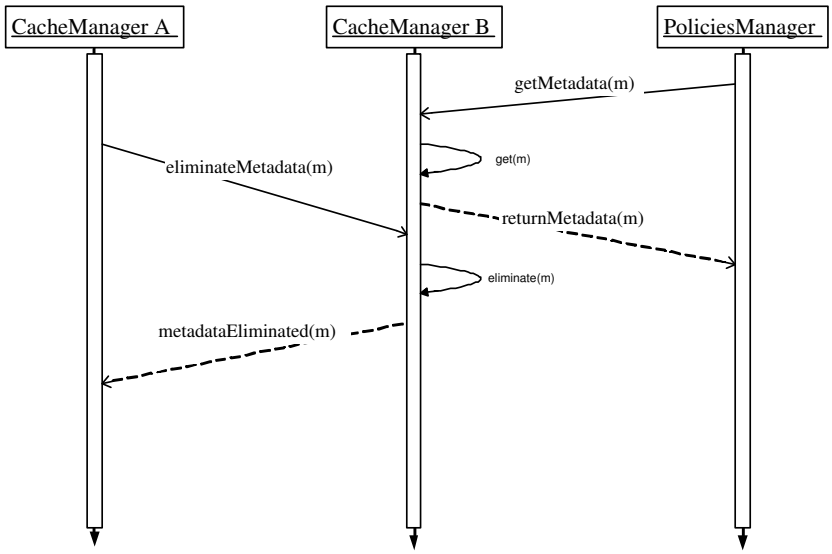

Figura 6.16: Diagramma di sequenza per la corsa critica: comportamento errato.

Si sono studiate due modalità per far fronte a questa corsa critica.

La prima consiste nell'inserire un *flag* all'interno della classe *PresentationMetadata* che indichi se la presentazione sia correntemente in uso da un agente. Quando un gestore riceve la richiesta di eliminazione di una presentazione che è correntemente utilizzata, invia un comando al gestore dei metadati presente nel place radice richiedendo il reinserimento della presentazione all'interno di tutto il sistema.

La seconda consiste nell'inserimento di un ulteriore flag, che sta ad indicare se la presentazione deve essere eliminata o meno dal sistema, e prevede una sotto-funzionalità all'interno della funzionalità di eliminazione: la *pre-eliminazione*. Quando un gestore riceve la richiesta di pre-eliminazione di un metadato può comportarsi in due modi differenti: se la presentazione non è correntemente in uso, deve settare il valore del flag di pre-eliminazione in maniera tale da non acconsentire ad una successiva richiesta del metadato relativo a quella presentazione; se la presentazione è correntemente in uso, deve comunicare agli altri gestori di annullare la richiesta di pre-eliminazione di quella presentazione.

In figura 6.17, è mostrato il diagramma degli stati del gestore dei metadati nel caso di eliminazione di un metadato. Quando il gestore dei metadati riceve la richiesta di eliminazione di un metadato, si sposta dallo stato iniziale (*stato 1*) e si porta in uno stato in cui analizza la richiesta (*stato 2*). Se la presentazione è correntemente utilizzata all'interno del sistema, il gestore non la elimina, ma invia il comando di inserimento al gestore presente nel nodo radice, in maniera tale da reinserire il metadato all'interno delle cache che lo avevano eliminato, e, finalmente, si riporta nello stato di attesa. Se la presentazione non è utilizzata il gestore può comportarsi in due modi differenti: se la richiesta proviene dal gestore delle politiche, prima di ritornare nello stato iniziale, il gestore dei metadati deve inoltrarla al gestore presente nel nodo radice, il quale provvederà a propagarla a tutto i gestori; se la richiesta proviene dal nodo radice, il gestore dei metadati elimina il metadato dalla propria cache, inoltra la richiesta di eliminazione a tutti i suoi figli e, finalmente, ritorna nello stato iniziale (*stato 1*).

Nel diagramma di sequenza di figura 6.18, è mostrato il comportamento dei gestori presenti all'interno del sistema all'atto di ricezione di una ri-

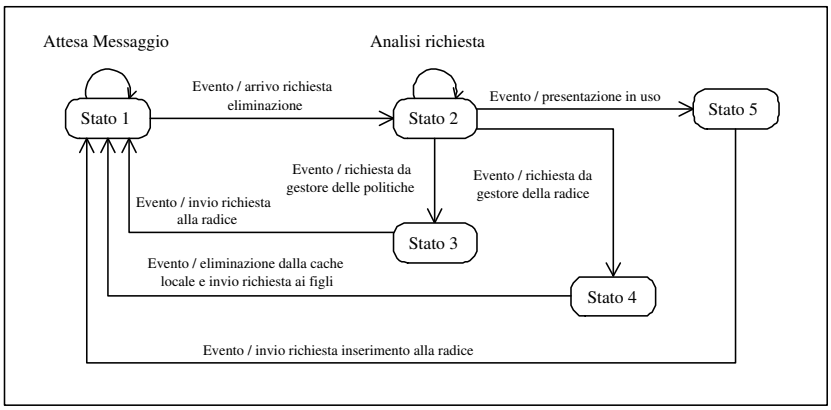

Figura 6.17: Diagramma degli stati del gestore dei metadati per l'eliminazione di un metadato.

chiesta di eliminazione di una presentazione non correntemente utilizzata all'interno del sistema.

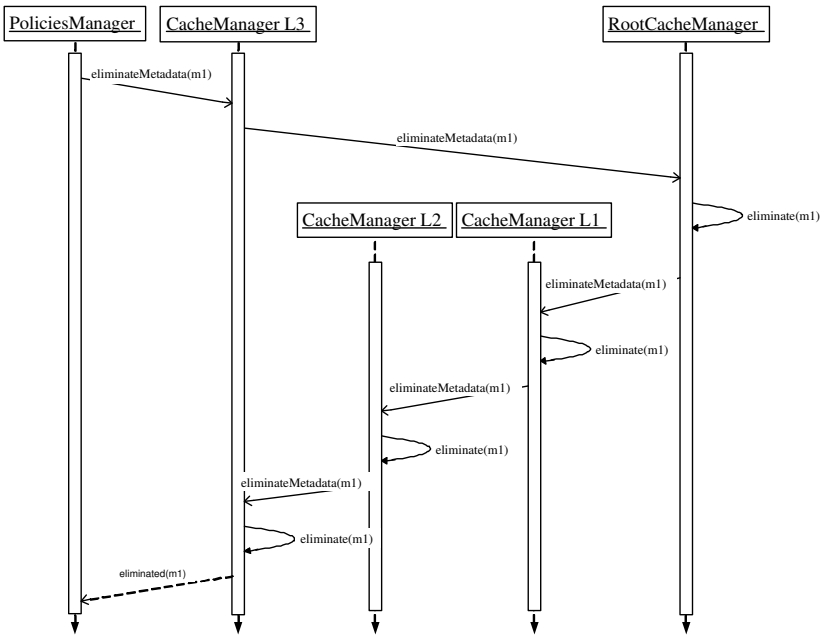

Figura 6.18: Diagramma di sequenza per l'eliminazione di un metadato.

Nel caso in cui il protocollo facesse uso della seconda soluzione proposta, i diagrammi appena presentati continuerebbero a rappresentare il comportamento del gestore, con l'unica differenza che nell'analizzare il

metadato (*stato 2*) il gestore dovrà tener conto, non soltanto del flag di utilizzo, ma anche di quello di pre-eliminazione e, in caso quest'ultimo sia attivo, dovrà semplicemente restituire una risposta negativa al richiedente.

La funzionalità di eliminazione avrebbe dunque al suo interno una sotto-funzionalità, la pre-eliminazione. In figura 6.19, è mostrato il diagramma degli stati del gestore dei metadati nel caso di pre-eliminazione di un metadato. Non appena il gestore dei metadati riceve la richiesta di pre-eliminazione si sposta dallo stato di attesa (*stato 1*), in uno stato in cui (*stato 2*) va a prelevare ed analizzare il metadato corrispondente alla presentazione che si vuole eliminare. Se non trova il metadato si riporta nello stato iniziale di attesa; se trova il metadato, controlla il flag che indica se la presentazione è in uso. A questo punto i casi sono due: se la presentazione non è in uso, imposta il flag e ritorna nello stato iniziale; se la presentazione è in uso, invia il comando per annullare la pre-eliminazione al gestore presente nel place radice, che successivamente lo propagherà a tutti i gestori del sistema, e si riporta, finalmente, nello stato iniziale (*stato 1*).

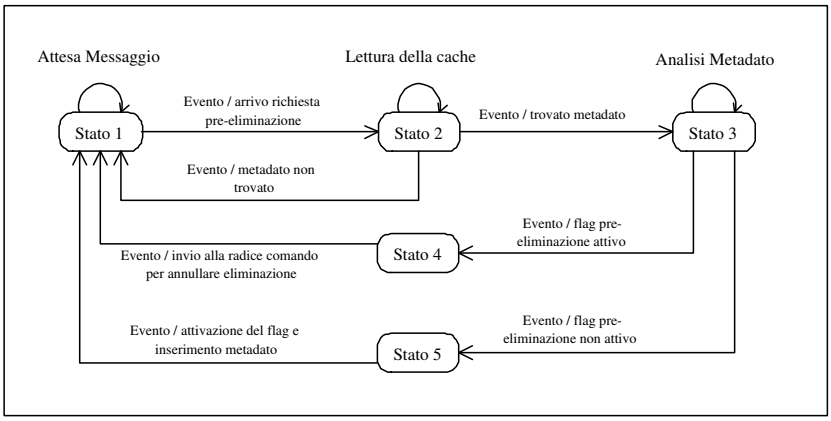

Figura 6.19: Diagramma degli stati del gestore dei metadati per la pre-eliminazione di un metadato.

6.2.3 Protocollo delle politiche di accesso

Il protocollo delle politiche di accesso si occupa del modo in cui i gestori delle politiche di accesso alla cache dei metadati si comportano nell'eseguire le proprie funzionalità.

Il gestore delle politiche di accesso alla cache rappresenta il punto di accesso al sistema di caching per un agente. Deve perciò offrire tre funzionalità principali: il reperimento di un metadato, l'eliminazione di un metadato e l'inserimento di un metadato. Le prime due non sono influenzate dalla particolare politica adottata dal gestore e, per questo motivo, le richieste di reperimento e di eliminazione vengono direttamente inoltrate al gestore dei metadati. Al contrario, la richiesta di inserimento deve essere prima di tutto analizzata in base alla politica scelta, e, successivamente, il gestore deciderà se inoltrare la richiesta di inserimento al gestore dei metadati o se restituire una risposta negativa all'agente che ha richiesto l'inserimento del metadato.

In figura 6.20, è mostrato il comportamento del gestore delle politiche di accesso precedentemente descritto.

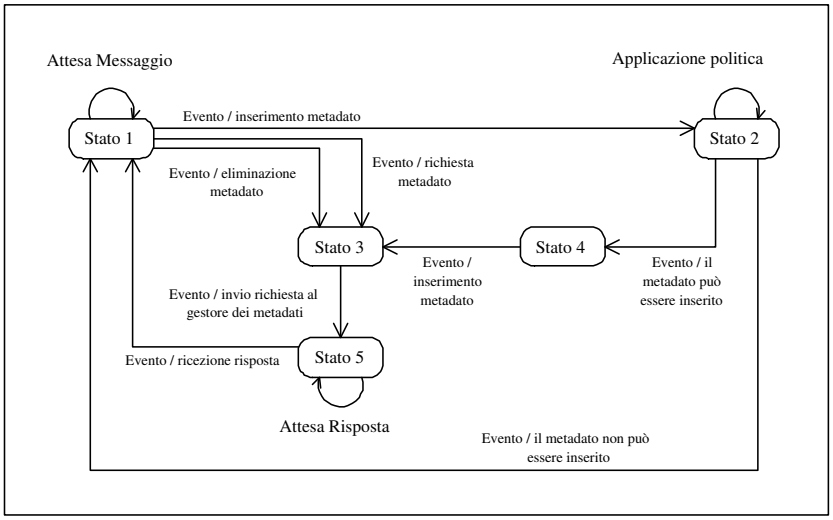

Figura 6.20: Diagramma degli stati del gestore delle politiche di accesso.

Possiamo, a questo punto, esaminare più in dettaglio il meccanismo di analisi di una richiesta di inserimento di un metadato. Il gestore delle

politiche richiede al sistema di caching le informazioni sulla presentazione che l'agente vuole inserire all'interno del sistema. Se la presentazione non è presente, il gestore delle politiche acconsente all'inserimento, inoltrando la richiesta al gestore dei metadati. Se la presentazione è già presente all'interno del sistema, esamina ogni metadato relativo a quella presentazione, sceglie quello che rappresenta la presentazione presente nel place più vicino ad esso, e confronta questa distanza con la distanza contenuta nella politica che sta adottando: se la politica impone una distanza maggiore, il metadato non viene inserito e l'agente riceve una risposta negativa; in caso contrario, inoltra la richiesta al gestore dei metadati.

6.3 Conclusioni

In questo capitolo è stata affrontata la fase di progettazione del sistema di gestione dei contenuti multimediali all'interno di MUM: si è riuscito, in questo modo, ad ottenere una rappresentazione dettagliata del nuovo componente e di tutte le sue parti.

Nel prossimo capitolo, verranno discusse alcune scelte in merito all'implementazione del componente e verranno esaminati i risultati del suo testing.

Si passerà, infine, ad esaminare le differenze tra le prestazioni che il sistema riesce ad ottenere grazie all'inserimento del nuovo componente e le prestazioni del sistema in assenza del componente, in maniera tale da verificarne l'effettivo miglioramento e, dunque, poter ritener corretta l'effettiva realizzazione del componente.

Capitolo 7

Scelte implementative e test del sistema

Nel capitolo precedente si è riusciti ad ottenere una rappresentazione dettagliata delle parti che compongono il sistema di gestione dei contenuti multimediali e del modo in cui le classi all'interno delle varie parti interagiscono fra di loro.

In questo capitolo, l'attenzione verrà posta dapprima su alcune scelte effettuate in fase di implementazione del nostro componente, mettendo in evidenza le motivazioni che ci hanno portato a compiere determinate scelte.

Si passerà, successivamente, alla verifica del corretto funzionamento del componente e, infine, all'analisi delle sue prestazioni complessive.

7.1 Alcune tracce sulle scelte implementative

In questo paragrafo, verranno esaminate alcune scelte effettuate in fase di implementazione del nostro componente.

La scelta è stata quella di utilizzare il linguaggio Java, sia perché il nuovo componente andrà ad inserirsi in un sistema completamente realizzato con questo linguaggio, sia per le sue caratteristiche fondamentali, come l'apertura o la portabilità.

Le scelte fondamentali in fase di implementazione sono le seguenti: il

modo in cui si inserisce il componente all'interno del sistema e come viene inizializzato; il modo in cui i gestori, sia quelli dei metadati sia quelli delle politiche, comunicano fra di loro; problematiche legate alle prestazioni del sistema, come l'occupazione di memoria o la velocità di risposta di una qualsiasi entità all'atto di una richiesta; l'utilizzo di classi aggiuntive, esterne al componente.

7.1.1 Inserimento e inizializzazione del componente

Prima di descrivere in che modo il nostro componente va ad inserirsi all'interno del sistema, è utile descrivere brevemente le classi principali di SOMA che intervengono nell'inserimento e nell'inizializzazione del componente.

La classe che rappresenta il singolo place è la classe **Environment**: dal momento che in ogni place deve essere presente un gestore del sistema di gestione dei contenuti multimediali, essa conterrà un riferimento ad esso. La classe incaricata di creare l'Environment è la classe **Creatore**: in base alla tipologia di place, default place radice, default place, place normale o place mobile, al suo interno vengono inizializzati tutti i componenti dell'Environment, compreso il gestore del sistema di gestione dei contenuti multimediali.

Nella fase di inizializzazione, ogni gestore dei metadati ha bisogno di comunicare con gli altri gestori, per ottenere le informazioni di cui ha bisogno: SOMA mette a disposizione delle funzionalità che permettono lo scambio di messaggi tra due place. A tal fine, è stato opportuno aggiungere all'interno del componente un package, denominato *commands*, contenente i comandi necessari alla comunicazione tra i gestori.

Il singolo comando può essere suddiviso in due parti: la prima parte è costituita da un metodo che viene eseguito nel place a cui il comando viene inviato, la seconda parte è costituita da un metodo che viene eseguito all'interno del place da cui è stato inviato il comando. Si tratta dell'applicazione del paradigma *Remote Evaluation*, che, come visto nel capitolo 3, evita la trasmissione di ingenti quantità di dati ed, in particolari do-

mini applicativi, comporta un utilizzo di risorse computazionali minore rispetto ad una comunicazione di tipo *Client/Server*.

I comandi necessari all'inizializzazione dei gestori del sistema sono i seguenti:

- **GetRootConnectionServiceCommand / SetRootConnectionServiceCommand**: è utilizzato dal gestore del sistema presente in ogni place di default, ad esclusione del place radice, per ottenere le informazioni per la connessione con il gestore del sistema presente nel place di default radice;
- **GetFatherConnectionServiceCommand / SetFatherConnectionServiceCommand**: è utilizzato dal gestore del sistema presente in ogni place di default, ad esclusione del place radice, per ottenere le informazioni per la connessione con il gestore del sistema presente nel place padre;
- **AddChildConnectionServiceCommand / RegisteredChildConnectionServiceCommand**: è utilizzato dal gestore del sistema per inviare le informazioni che lo riguardano al gestore presente nel place padre;
- **AddBrotherConnectionServiceCommand / RegisteredBrotherConnectionServiceCommand**: è utilizzato dal gestore del sistema per inviare ai figli le informazioni che riguardano i place dello stesso livello che vengono creati successivamente;
- **GetGenericMetadataServiceCommand / SetGenericMetadataServiceCommand**: è utilizzato dal creatore del place per inizializzare il gestore del sistema di gestione dei metadati presente in un place di default che non sia la radice della gerarchia;
- **GetNormalMetadataServiceCommand / SetNormalMetadataServiceCommand**: è utilizzato dal creatore del place per inizializzare il gestore del sistema di gestione dei metadati presente in un place normale;

- **GetCurrentPolicyCommand / SetCurrentPolicyCommand**: è utilizzato dal gestore del sistema per ottenere le informazioni sulla politica corrente di accesso alla cache.

Tutti i comandi sono organizzati in maniera tale da aspettare che le informazioni necessarie siano disponibili solamente per un determinato intervallo temporale. Questa scelta evita che l'intero sistema si blocchi indefinitamente nel momento in cui un comando non riuscisse, per un qualche motivo, ad ottenere le informazioni necessarie.

7.1.2 Comunicazione tra i gestori

La comunicazione tra i gestori del sistema di caching dei contenuti multimediali è uno degli aspetti fondamentali per il corretto funzionamento del componente. Lo scambio di informazioni tra i gestori è infatti necessario per il corretto funzionamento del protocollo di caching e del protocollo per le politiche di accesso alla cache.

Analizzando i requisiti che deve rispettare il sistema di comunicazione, si è arrivati alla conclusione che essa potesse essere realizzata in due differenti modi: utilizzando degli agenti mobili oppure utilizzando le funzionalità offerte dal protocollo TCP.

Dal momento che il componente va ad inserirsi in un'infrastruttura per il supporto di agenti mobili, sembrerebbe sensato utilizzarli per realizzare la comunicazione tra i gestori. Tuttavia, si è optato per l'utilizzo delle funzionalità offerte dal TCP, dal momento che per la comunicazione tra i gestori i vantaggi dovuti all'utilizzo degli agenti mobili non vanno ad influire sulle prestazioni. La comunicazione, infatti, essendo un semplice scambio di messaggi, non necessita né di risparmio di risorse né di dinamicità, esigenze alle quali si fa fronte con l'utilizzo degli agenti mobili.

In ogni gestore del sistema il MetadataServiceConnectionManager deve restare continuamente in attesa di messaggi: esso è perciò realizzato come un *Thread*, funzionalità offerta dal linguaggio di programmazione utilizzato, ed è in ascolto su una porta ad uno specifico indirizzo.

Quando il MetadataServiceConnectionManager riceve un messaggio, incarica un altro Thread, il **MetadataServiceProtocol**, dell'analisi del messaggio, che verrà poi fatto giungere al destinatario.

I messaggi sono contenuti all'interno di un nuovo package, chiamato **messages**: esiste una classe astratta, **MetadataMessage**, che rappresenta il generico messaggio e implementa l'interfaccia *Serializable* di Java in maniera tale da permettere la scrittura del messaggio nel canale di comunicazione tra i gestori; tutti gli altri messaggi derivano dalla classe astratta e ognuno implementa funzionalità differenti: il messaggio **GetMetadataMessage** utilizzato per richiedere un metadato, il messaggio **InsertMetadataMessage** utilizzato per richiedere l'inserimento di un metadato e il messaggio **EliminateMetadataMessage** utilizzato per richiedere l'eliminazione di un metadato.

7.1.3 Problematiche legate alle prestazioni

Dal momento che le problematiche legate alle prestazioni rivestono un'importanza fondamentale nello sviluppo del sistema, uno dei principi seguiti nella realizzazione del sistema è stato quello di minimizzare l'utilizzo di memoria: si è cercato dunque di mantenere in memoria solamente i dati strettamente necessari al corretto funzionamento del componente. Tuttavia, un'eccezione è stata fatta per la classe che fornisce i metodi per l'accesso alla cache dei metadati. Per velocizzare la ricerca e il recupero di un metadato a fronte di una eventuale richiesta, si è scelto di mantenere in memoria la lista dei metadati presenti nella cache.

Un'altra scelta che influisce sulle prestazioni del sistema è il modo in cui il protocollo di gestione del sistema di caching dei metadati realizza la funzionalità di eliminazione. Nel capitolo precedente sono state messe in evidenza due modalità differenti per l'eliminazione di un metadato: l'inserimento di un flag all'interno della classe *PresentationMetadata* per indicare che la presentazione è correntemente utilizzata oppure l'inserimento di un ulteriore flag e di una funzionalità aggiuntiva all'interno della funzionalità di eliminazione, la pre-eliminazione.

Dal momento che l'utilizzo della seconda modalità appesantisce il protocollo e, in caso di disconnessione non voluta di un dominio, potrebbe portare il sistema in uno stato inconsistente, la scelta è ricaduta sulla prima modalità.

Una volta che la presentazione non è più utilizzata, il flag di utilizzo deve essere in qualche modo azzerato. Per far fronte a questa esigenza sono state studiate due modalità: utilizzare un Thread, avente il compito di azzerare il flag all'interno della presentazione mediante considerazioni sulla sua durata; modificare l'agente che permette la fruizione della presentazione in maniera tale che sia esso stesso a comunicare al gestore della cache dei metadati che la presentazione non è più utilizzata.

La scelta è ricaduta sulla seconda soluzione per due limitazioni di fondamentale importanza legate alla prima soluzione: un Thread sottrae risorse computazionali al sistema, restando attivo per tutto il tempo di fruizione della presentazione; le considerazioni sulla durata di una presentazione potrebbero risultare inesatte dal momento che l'utente potrebbe anche decidere di interrompere la fruizione della presentazione prima che questa sia terminata.

7.1.4 Utilizzo di classi aggiuntive

All'interno del componente si fa uso di classi che non ne fanno parte. Fondamentalmente si possono suddividere in due gruppi: le classi facenti parte del linguaggio Java e le classi facenti parte di SOMA.

Le prime sono state utilizzate per la creazione di vettori, liste, thread e per realizzare la comunicazione tra i gestori attraverso il protocollo TCP.

Le seconde sono state utilizzate principalmente per introdurre ed inizializzare il componente all'interno di SOMA nonché per scambiare i messaggi tra i vari place e per interagire con gli altri componenti del sistema.

7.2 Test del sistema

La fase di test è di fondamentale importanza per la verifica del corretto raggiungimento degli obiettivi dello sviluppo del sistema di gestione dei contenuti multimediali e per la verifica delle prestazioni che esso ottiene.

Al fine di testare il sistema è stato creato un apposito agente, il **TestMetadataServiceAgent**, in grado di interagire con il sistema di gestione dei contenuti multimediali. Attraverso l'agente è possibile effettuare le richieste di inserimento, reperimento o eliminazione di un metadato.

Per monitorare i parametri fondamentali, come il tempo o le informazioni sui gestori del sistema, sono stati utilizzati gli strumenti messi a disposizione da **Log4j** [LOG4J]. Log4j è una libreria sviluppata da *Apache* che permette, attraverso semplici file di configurazione, di memorizzare comodamente in un server centrale tutti i messaggi generati da tutti i nodi all'interno del sistema distribuito.

7.2.1 Configurazioni di test

Il test del sistema è stato realizzato utilizzando tre differenti configurazioni di calcolatori:

- **Singolo calcolatore**: è stato simulato su un singolo calcolatore, grazie alle funzionalità offerte da SOMA, un sistema distribuito. Le caratteristiche del calcolatore utilizzato sono le seguenti:

  - **Processore**: Intel Pentium IV 1.8 GHz;
  - **RAM**: 768 MB;
  - **Sistema operativo**: Microsoft Windows 2000 Professional.

  Questa configurazione è stata utilizzata solamente per una prima verifica del corretto funzionamento del sistema e non per monitorarne le prestazioni.

- **Rete locale 1**: si tratta di una rete locale a 10 Mbps formata da tre calcolatori con le seguenti caratteristiche:

+ **PC1** desktop:
  - **Processore**: Intel Pentium IV 1.8 GHz;
  - **RAM**: 768 MB;
  - **Sistema operativo**: Microsoft Windows 2000 Professional.
+ **PC2** desktop:
  - **Processore**: Intel Pentium IV 2.6 GHz;
  - **RAM**: 512 MB;
  - **Sistema operativo**: Microsoft Windows XP Professional.
+ **PC3** notebook:
  - **Processore**: PowerPC G4 1.25 GHz;
  - **RAM**: 512 MB;
  - **Sistema operativo**: Mac OS X 10.3.2 Panther.

Questa configurazione è stata utilizzata per la verifica del corretto funzionamento del sistema e per il monitoraggio delle prestazioni del sistema.

- **Rete locale 2**: si tratta di una rete locale a 100 Mbps formata da più calcolatori con le seguenti caratteristiche:
  - **Processore**: UltraSPARC III 1.2 GHz;
  - **RAM**: 1024 MB;
  - **Sistema operativo**: SunOS Release 5.9.

Questa configurazione è stata utilizzata per il monitoraggio delle prestazioni del sistema.

7.2.2 Verifica del corretto funzionamento del componente

La verifica del corretto funzionamento del componente si è basata sul monitoraggio delle variabili principali all'interno del sistema.

Inizialmente si è verificata la corretta inizializzazione del gestore del sistema e di tutte le sue sotto-parti: il gestore della cache e il gestore delle politiche.

Di fondamentale importanza è stato verificare che le informazioni per la comunicazione tra i gestori del sistema presenti in ogni place fossero corrette. Basandosi sulla gerarchia di domini presente in figura 7.1, è stato appurato il corretto valore di tutte le informazioni utilizzate per la comunicazione.

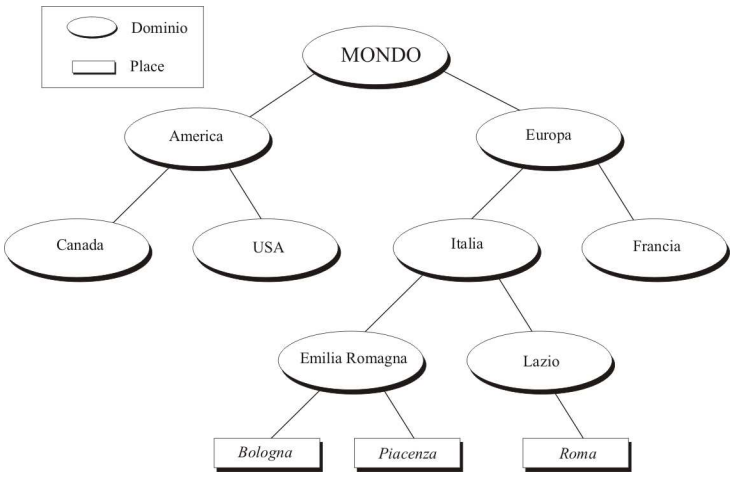

Figura 7.1: Gerarchia di dominii per il test del sistema.

Le informazioni per la comunicazione sono rappresentate da un indirizzo, che identifica il calcolatore su cui è in esecuzione il gestore, e una porta, alla quale il gestore è in ascolto. In ogni gestore sono presenti le informazioni che riguardano il gestore stesso, il gestore presente nella radice della gerarchia, il gestore padre, i gestori figli e i gestori fratelli, cioè quelli appartenenti allo stesso livello. Ad esempio, le configurazioni di alcuni domini e place appartenenti alla gerarchia di figura 7.1 potrebbero essere:

- **Mondo**: *Info*: [Mondo - 1188] / *Children Info*: [America - 1203] [Europa - 1228]
- **Europa**: *Info*: [Europa - 1228] / *Root Info*: [Mondo - 1188] / *Father Info*: [Mondo - 1188] / *Brothers Info*: [America - 1203] / *Children Info*: [Francia - 1238] [Italia - 1248]

- **Italia**: *Info*: [Italia - 1248] / *Root Info*: [Mondo - 1188] / *Father Info*: [Europa - 1228] / *Brothers Info*: [Francia - 1238] / *Children Info*: [Emilia Romagna - 1263] [Lazio - 1279]
- **Emilia Romagna**: *Info*: [Emilia Romagna - 1263] / *Root Info*: [Mondo - 1188] / *Father Info*: [Italia - 1248] / *Brothers Info*: [Lazio - 1279]
- ***Bologna***: *Father Info*: [Emilia Romagna - 1263]

7.2.3 Analisi delle prestazioni del sistema

In questo paragrafo si procederà all'analisi delle prestazioni del sistema per quanto riguarda la richiesta, l'inserimento e l'eliminazione di una presentazione.

Il test è stato condotto facendo uso della gerarchia descritta in figura 7.1, nelle due reti locali descritte precedentemente: nella prima rete locale, dato il numero limitato di calcolatori, è stato necessario simulare la presenza di più domini o più place in una sola macchina; nella seconda rete locale, invece, è stato possibile utilizzare una singola macchina per ogni singolo dominio o place.

Richiesta di una presentazione

Nell'analizzare le prestazioni del sistema all'atto di richiesta di una presentazione è utile descrivere il comportamento del sistema suddividendolo in due parti: una parte di *transitorio* e una parte di *regime*. Durante il transitorio, le presentazioni sono presenti solamente nel dominio radice: una eventuale richiesta da un altro dominio deve propagarsi fino alla radice prima di poter essere soddisfatta. Quando, invece, il sistema è a regime, le presentazioni sono distribuite all'interno del sistema nei domini in cui sono state richieste: una richiesta ha più possibilità di essere soddisfatta in un tempo più breve.

In figura 7.2, è mostrato il comportamento del sistema nella rete locale 1 all'atto di richiesta di una presentazione. Nel diagramma si vede come per la prima richiesta di una presentazione, cioè nel transitorio, i

tempi siano proporzionali alla distanza che la richiesta deve compiere per arrivare al dominio Mondo. Già ad una seconda richiesta della stessa presentazione, i tempi di risposta praticamente si annullano, ad eccezione di un minimo intervallo di tempo in cui la richiesta deve essere processata.

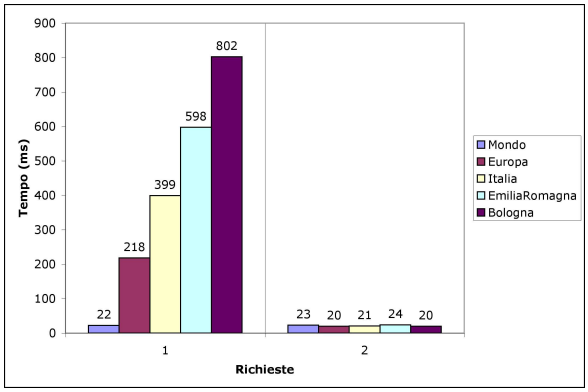

Figura 7.2: Comportamento del sistema nella rete locale 1 all'atto di richiesta di una presentazione.

In figura 7.3, è mostrato il comportamento del sistema nella rete locale 2 all'atto di richiesta di una presentazione. Il comportamento del sistema in questa rete è identico a quello che ha nella prima rete con la sola eccezione dei tempi: la velocità più elevata della rete e le maggiori risorse computazionali dei calcolatori, fanno sì che i tempi subiscano una riduzione.

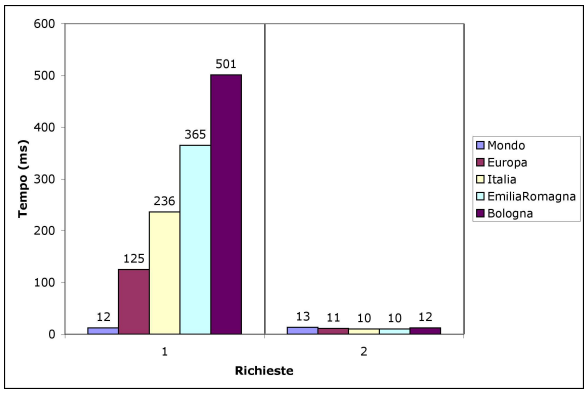

Figura 7.3: Comportamento del sistema nella rete locale 2 all'atto di richiesta di una presentazione.

Volendo fare un confronto con il precedente sistema centralizzato di gestione dei contenuti multimediali all'interno di MUM, possiamo evidenziare che i tempi di risposta a regime subiscono un notevole miglioramento dal momento che il sistema centralizzato mantiene costantemente dei tempi di risposta molto simili a quelli del transitorio che si ha nel sistema distribuito.

Inserimento di una presentazione

All'atto di inserimento di una presentazione all'interno del sistema, occorre un certo intervallo di tempo affinché la presentazione sia effettivamente disponibile: succede infatti che la richiesta di inserimento deve propagarsi prima verso la radice della gerarchia e successivamente attraverso tutti i domini, che si trovano tra il place in cui è stata effettuata la richiesta e il place radice, e che provvederanno alla sua memorizzazione.

In figura 7.4, è mostrato il comportamento del sistema nella rete locale 1 all'atto di inserimento di una presentazione. Per ogni dominio o place da cui viene effettuata la richiesta di inserimento, è indicato il tempo totale che il sistema impiega per completare la richiesta di inserimento di una presentazione.

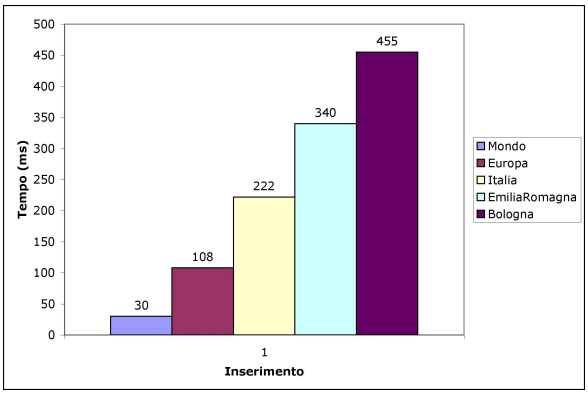

Figura 7.4: Comportamento del sistema nella rete locale 1 all'atto di inserimento di una presentazione.

In figura 7.5, è mostrato il comportamento del sistema nella rete locale

2 all'atto di inserimento di una presentazione. Il comportamento è identico a quello precedente con la sola eccezione dei tempi più brevi.

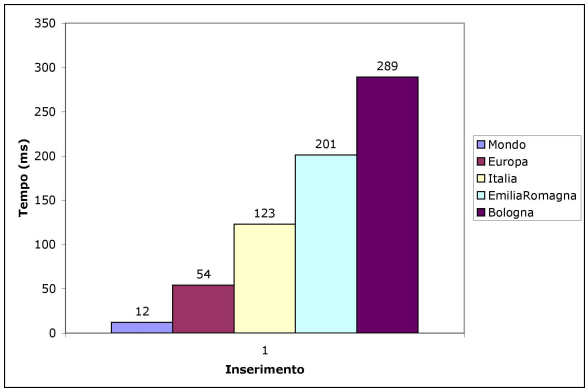

Figura 7.5: Comportamento del sistema nella rete locale 2 all'atto di inserimento di una presentazione.

Volendo, anche in questo caso, fare un confronto con il precedente sistema centralizzato di gestione dei contenuti multimediali, si può osservare che, nonostante il nuovo sistema impieghi un tempo maggiore per completare la richiesta di inserimento di una presentazione, questa è subito disponibile all'interno del sistema, dal momento che viene immediatamente memorizzata nel dominio radice, con un tempo uguale a quello del sistema centralizzato.

Eliminazione di una presentazione

La richiesta di eliminazione di una presentazione richiede un tempo variabile in quanto dipende dal numero di domini che hanno memorizzato la descrizione della presentazione.

Il caso peggiore si ha quando la descrizione della presentazione che si vuole eliminare è presente in tutti i domini nel tratto più lungo che congiunge la radice ad una foglia: in questo caso, la richiesta deve propagarsi dal dominio radice a tutti gli altri domini che fanno parte della gerarchia, fino a giungere alla foglia.

In figura 7.6, è mostrato il comportamento del sistema nella rete locale 1 all'atto di eliminazione di una presentazione. Sono evidenziati tre casi,

in base al numero di domini all'interno del tratto più lungo che va dalla radice ad una foglia della gerarchia.

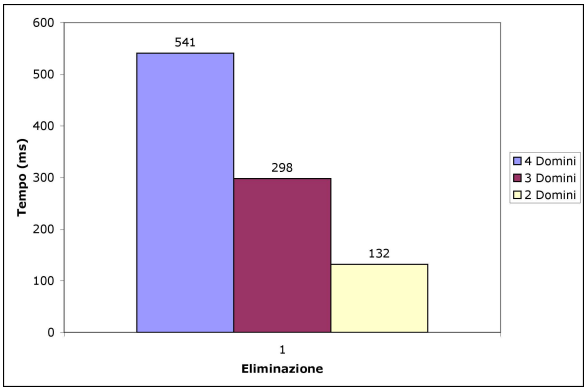

Figura 7.6: Comportamento del sistema nella rete locale 1 all'atto di eliminazione di una presentazione.

In figura 7.7, è mostrato il comportamento del sistema nella rete locale 2 all'atto di eliminazione di una presentazione, nelle stesse condizioni della rete precedente.

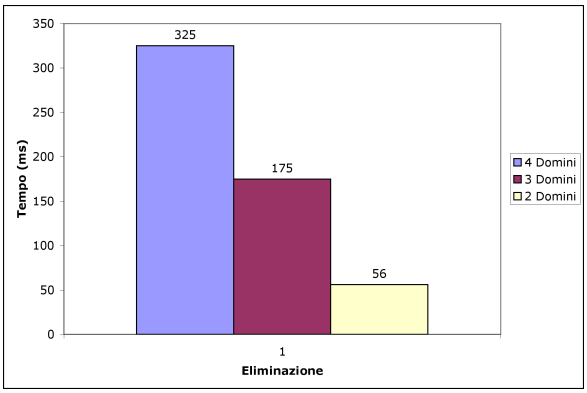

Figura 7.7: Comportamento del sistema nella rete locale 2 all'atto di eliminazione di una presentazione.

Rispetto al precedente sistema centralizzato di gestione dei contenuti multimediali, la richiesta di eliminazione richiede più tempo per essere eseguita, in quanto deve essere propagata più o meno in profondità all'interno della gerarchia dei domini. Questo peggioramento dei tempi, che

influisce negativamente sulle prestazioni del sistema, non va tuttavia a decrementare il livello di Qualità di Servizio che il sistema offre all'utente, dal momento che la funzionalità di eliminazione non rientra all'interno dei parametri che caratterizzano la Qualità di Servizio.

7.3 Conclusioni

Con la descrizione delle principali scelte implementative e l'analisi delle prestazioni complessive del sistema termina questo lavoro di tesi.

Quello che emerge è che gli obiettivi sono stati raggiunti e i requisiti soddisfatti: il sistema di gestione dei contenuti multimediali contribuisce all'incremento della Qualità di Servizio all'interno dell'applicazione per la fruizione di presentazioni multimediali in cui va ad inserirsi. Il dato più rilevante è il netto miglioramento dei tempi di risposta del sistema all'atto di richiesta di una presentazione multimediale grazie all'utilizzo di un sistema di caching. L'apertura del sistema verso altre applicazioni è garantita dall'utilizzo di uno schema per la descrizione delle presentazioni multimediali conforme ad alcuni standard correnti.

Nelle Conclusioni di questo lavoro di tesi, verranno messe in evidenza le scelte fondamentali per lo sviluppo del sistema e si accennerà ad alcuni suoi possibili sviluppi futuri.

Conclusioni

Nel corso dello sviluppo del progetto di tesi sono state affrontate due tematiche molto importanti nel campo dei sistemi multimediali distribuiti: il miglioramento della Qualità di Servizio all'interno di un'applicazione multimediale distribuita mediante la realizzazione di un sistema di caching e l'apertura di un'applicazione multimediale nei confronti di altre applicazioni che fanno uso di informazioni dello stesso tipo.

Siamo partiti dall'analisi dei principi che regolano la Qualità di Servizio all'interno di un sistema multimediale distribuito, osservando come una loro corretta applicazione stia alla base dell'incremento delle prestazioni dell'applicazione.

L'attenzione si è poi concentrata sulla scelta di un sistema di caching da inserire all'interno dell'applicazione, in maniera tale da incrementarne la Qualità di Servizio. La scelta è ricaduta su un sistema di caching di tipo distribuito che faccia uso di un protocollo di caching di tipo ibrido. Questo tipo di protocollo coniuga le funzionalità dei protocolli di tipo gerarchico e distribuito, ed è sembrato quello più adatto alle esigenze dell'applicazione, dal momento che garantisce un'efficiente e funzionale gestione del sistema di caching, utilizzando nello stesso tempo una quantità limitata di risorse computazionali.

Una delle prerogative fondamentali di un'applicazione multimediale è l'apertura, ossia la sua capacità a scambiare informazioni con altre applicazioni. Dall'analisi degli standard internazionali per la rappresentazione di informazioni di tipo multimediale è emerso che ancora molto deve essere fatto in quest'ambito per definire un modello esaustivo che possa essere utilizzato dalla maggior parte delle applicazioni. È stato perciò ne-

cessario definire un nuovo schema per la rappresentazione di contenuti di tipo multimediale, prendendo in considerazione le caratteristiche migliori degli standard correnti.

Lo sviluppo è stato articolato in più fasi, con livello di dettaglio crescente: si è partiti dalla fase di analisi, per poi passare alla progettazione e, infine, all'implementazione.

È stato necessario fissare delle ipotesi di funzionamento, che in qualche modo limitano le potenzialità dell'utilizzo di un sistema di caching. Non si è voluta, tuttavia, precludere la possibilità per il futuro sviluppo di altre modalità di funzionamento: per questo motivo nella fase di progettazione del componente sono stati presi in considerazione diverse tipologie di protocolli di caching ed è stato solo in fase di implementazione che si è optato per uno di questi.

Il test del sistema è riuscito a mettere in evidenza due aspetti fondamentali del nuovo componente: il suo corretto funzionamento all'interno dell'applicazione multimediale e l'effettivo incremento delle prestazioni dell'intero sistema. È emerso, infatti, che i tempi necessari al reperimento di informazioni sui contenuti multimediali presenti all'interno del sistema diminuiscono notevolmente grazie al suo utilizzo, a fronte di un minimo aumento dell'utilizzo di banda trasmissiva necessaria allo scambio di informazioni tra le entità che compongono il sistema di caching.

La realizzazione del sistema di caching apre nuove prospettive di miglioramento, che potrebbero essere oggetto di eventuali sviluppi futuri.

La creazione di meccanismi di *quoting* per l'ottimizzazione dello spazio su disco utilizzato dalle cache oppure l'applicazione di politiche di rimozione del dato presente nella cache sono due aspetti che potrebbero essere presi in considerazione in un successivo ampliamento del componente.

Infine, in futuro si potrebbe cercare di raffinare ulteriormente lo schema per la rappresentazione delle informazioni, incrementando il livello di dettaglio della descrizione di un dato multimediale, facendo fronte, in questo modo, alla continua evoluzione dei contenuti multimediali.

Ringraziamenti

I miei ringraziamenti vanno al prof. Antonio Corradi per avermi offerto l'opportunità di sviluppare il presente lavoro di tesi e per la fiducia mostrata nei miei confronti. Ringrazio i miei due correlatori, il prof. Paolo Bellavista e l'ing. Luca Foschini, per l'assistenza, le indicazioni, i consigli e la disponibilità fornitemi sempre con puntualità e precisione.

Un ringraziamento particolare e affettuoso va alla mia famiglia.

Ringrazio, infine, le mie amiche e i miei amici: Alberto, Fabrizio, Giuseppe, Benedetta, Elisa, Chiara, Edoardo, Francesco, Gabriele, David, Nico, Fé, Marco, Giacomino, Paolo, Andrea, Matteo, Enrico, Vincenzo, Antonio, Francesco, Carmen, Antonella, Mimmo, Lamberto, Emiliano, Stefano, Ennio, Gianni, Fabio, Riccardo, Ferdinando e tutti quelli che, non per mia volontà, ho dimenticato.

Bibliografia

[AUR95] C. Aurrecoecha, A. Campbell. L. Hauw, *A Review of Quality of Service Architectures*, ACM Multimedia Systems Journal, November 1995.

[BOC97] G. Bochmann, B. Kerherve, A. Hafid, P. Dini, A. Pons, *Some Principles for QoS Management*, Distributed System Engineering Journal, Vol. 4, N. 1, 16-27, 1997.

[BOH94] K. Bohms, T. Rakow, *Metadata for multimedia documents*, ACM Sigmond Record, 23, N. 4, 1994.

[CAM94] A. Campbell, G. Coulson, D. Hutchison, *A Quality Of Service Architecture*, ACM Computer Communication Review, April 1994.

[DC] http://dublincore.org/

[FER92] D. Ferrari, A. Banerjea, H. Zang, *Network Support for Multimedia*, Technical report 92-172, International Computer Science Institute, Berkeley, November 1992.

[CORBA] http://www.corba.org/

[COU01] G. Couloris, J. Dollimore, T. Kindberg, *Distributed Systems Concepts and Design*, Third Edition, Addison-Wesley, Milano, 2001.

[FIPA] http://www.fipa.org/

[FLO00] M. Fowler, K. Scott, *UML Distilled*, Addison-Wesley, Milano, 2000.

[FOS03] Foschini L., *Multimedia Streaming Management Over Wired and Wireless Networks*, Tesi di Laurea presso l'Università di Bologna, 2003.

[FOW99] M. Fowler, *Refactoring: improving the design of existing code*, Addison-Wesley, Milano, 1999.

[FUG98] A. Fuggetta, G. P. Picco, G. Vigna, *Understanding Code Mobility*, IEEE Transactions on Software Engineering, Vol. 24, N. 5, May 1998.

[GAM02] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: elementi per il riuso di software ad oggetti*, Addison-Wesley, 2002.

[HAF95] A. Hafid, G. v. Bochmann, *An Approach to Quality of Service Management for Distributed Multimedia Systems*, International Conference on Open Distributed Processing (ICODP-95), Australia, Feb. 1995, pp. 319-340.

[HAF98] A. Hafid, G. v. Bochmannn, R. Dssouli, *Distributed Multimedia Application and Quality of Service: A Review*, Electronic Journal on Networks and Distributed Processing, N. 6, pp. 1-50, 1998.

[HUT94] D. Hutchison, G. Coulson, A. Campbell, G. Blair, *Quality of Service Management in Distributed Systems*, Addison-Wesley, 1994.

[ITU89] CCITT Rec. I.350, *General Aspect of Quality of Service and Network Performance in Digital Networks*, International Telecommunication Union, Geneva, 1989.

[JAVA] http://www.java.sun.com/

[KAL94] G. Kalkbrenner, T. Pirkmayer, V. Dormik, P. Hoffman, *Quality of Service in Distributed Hypermedia-Systems*, The Second International Workshop on Principles of Document Processing, Darmstadt, April 1994.

[KAR98] N. M. Karnik, A. R. Tripathi, *Design Issues in Mobile-Agent Programming Systems*, IEEE Concurrency, 1998.

[LOG4J] http://logging.apache.org/log4j/docs/

[MAC02] L. A. Maciaszek, *Sviluppo di sistemi informativi con UML*, Addison-Wesley, 2002.

[MPEG7] http://archive.dstc.edu.au/mpeg7-ddl/

[MUM] http://www-lia.deis.unibo.it/research/MUM/

[OMG] http://www.omg.org/

[NWA96] H. S. Nwana, *Software Agents: An Overview*, Knowledge Engineering Review, Vol. 11, N. 3, pp. 1-40, September 1996.

[POV97] D. Povey, J. Harrison, *A Distributed Internet Cache*, Proc. 20th Australian Computer Science Conference, Sydney, Australia, Feb. 1997.

[PRE00] S. Pressman, *Principi di ingegneria del software*, Mc Graw Hill, Milano, 2000.

[RAC91] RACE IBC CFS D510, *General Aspects of Quality of Service and System Performance*, IBC, RIC, Brussels, 1991.

[ROD01] P. Rodriguez, C. Spanner, E. W. Biersack, *Analysis of Web Caching Architectures: Hierarchical and Distributed Caching*, IEEE/ACM Transactions on Networking, Vol. 9, N. 4, August 2001.

[ROD98] P. Rodriguez, K. W. Ross, E. W. Biersack, *Distributing frequently-changing documents in the Web: Multicasting or hierarchical caching*, Proc. Computer Networks and ISDN Systems: Selected Papers 3rd Int. Caching Workshop, 1998, pp. 2223-2245.

[ROU98] A. Rousskov, D. Wessels, *Cache Digests*, Proceedings of the 3rd International WWW Caching Workshop, June 1998.

[SOMA] http://www-lia.deis.unibo.it/research/SOMA/

[SPA98] C. Spanner, *Evaluation of web caching strategies: Distributed vs. hierarchical caching*, Masters thesis, Univ. Munich/Institut Eurocom, Sophia Antipolis, France, Nov. 1998.

[TAN98] A. S. Tanenbaum, *Reti di Computer*, Terza Edizione, UTET Libreria, Giugno 1998.

[TEW99] R. Tewari, M. Dahlin, H. M. Vin, J. S. Kay, *Beyond hierarchies: Design considerations for distributed caching on the Internet*, Proc. ICDCS '99 Conf., Austin, TX, May 1999.

[UML] http://www.uml.org/

[TRI98] A. R. Tripathi, N. M. Karmir, *Design Issues in Mobile-Agent Programming Systems*, IEEE Concurrency, July-September 1998.

[VOG95] A. Vogel, *Distributed Multimedia and QoS: A Survey*, IEEE Multimedia, pp. 10-18, Summer 1995.

[WES97] D. Wessels, K. Claffy, *Application of Internet Cache Protocol (ICP)*, IETF, Internet Draft: draft-wessels-icp-v2-appl-00. Work in progress., May 1997.

[WIL65] M. V. Wilkes, *Slave Memories and Dynamic Storage Allocation*, April 1995.

[XML] http://www.w3.org/XML/

Elenco delle figure