4° Convegno Tecnico Scientifico

Torino 25, 26, 27 Ottobre 2004

Una architettura peer-to-peer per la visualizzazione 3D distribuita

Visualizzazione 3D distribuita

Claudio Zunino
Politecnico di Torino, Dip. di Automatica e Informatica (DAUIN)
C.so Duca degli Abruzzi 24, I-10129
claudio.zunino@polito.it

Andrea Sanna
Politecnico di Torino, Dip. di Automatica e Informatica (DAUIN)
C.so Duca degli Abruzzi 24, I-10129
andrea.sanna@polito.it

ABSTRACT

Intrattenimento, e-learning, e-commerce, lavoro on-line sono solo alcuni dei settori dove le rappresentazioni 3D sono estremamente importanti. Allo stesso tempo nuove tecnologie permettono la condivisione di file (ma più in generale di risorse) usando il peer-to-peer (P2P).

In questo articolo viene proposto l'uso di un approccio P2P per progettare e realizzare una piattaforma multi utente, efficiente e altamente scalabile, per la visualizzazione 3D distribuita. In particolare, la tecnologia P2P è usata per progettare e realizzare un sistema di visualizzazione dove più utenti possono analizzare contemporaneamente una scena virtuale. Per lo sviluppo dell'infrastruttura di rete è stato usato il framework JXTA sviluppato dalla Sun Microsystem, mentre l'interfaccia grafica è stata realizzata con le librerie Java3D.

Il sistema offre agli utenti anche la possibilità di comunicare direttamente usando un sistema di videoconferenza basato su flussi di dati audio/video.

Introduzione

Negli ultimi anni Internet è stata usata per gli scopi più diversi: intrattenimento, e-learning, e-commerce, lavoro on-line, e molti altri. L'intrattenimento è un settore dove le rappresentazioni 3D sono estremamente importanti: per esempio ci sono delle vere e proprie comunità on-line come ActiveWorlds, o giochi 3D dove ogni utente sfida gli altri giocatori usando Internet. Anche l'e-learning sta diventando un modo concreto per fornire conoscenza. In questo contesto l'introduzione del 3D può migliorare la chiarezza e rendere possibile l'interazione con i modelli studiati, per esempio di un motore o di un altro oggetto complesso.

Inoltre, negli ultimi anni sta crescendo anche un altra tecnologia: quella dei peer-to-peer (P2P). Il P2P premette di progettare e realizzare grandi sistemi distribuiti destinati alla condivisione di risorse (tipicamente file di dati) tra i vari peer. Il termine peer indica un nodo della rete che, in generale, contemporaneamente può richiedere dei servizi agli altri nodi (agendo come client) e fornire esso stesso un servizio (agendo come server).

In questo articolo si è usato un approccio P2P per progettare e realizzare una piattaforma distribuita per la visualizzazione 3D multi utente efficiente e altamente scalabile. L'obiettivo principale è realizzare un sistema per condividere modelli 3D, permettendo a più utenti di interagire con essi. Il sistema fornisce anche un ambiente virtuale dove gli stessi utenti, rappresentati da un avatar, possono incontrarsi e comunicare. Un utente può caricare nel mondo virtuale oggetti in formato 3DS e VRML; l'oggetto può essere reso visibile agli altri utenti condividendo la sessione di visualizzazione.

L'applicazione è stata realizzata usando tecnologie Java, in particolare l'architettura JXTA per la rete P2P e le librerie Java3D per il rendering degli oggetti. Il vantaggio nell'uso di Java3D è dovuto alla possibilità di sfruttare l'accelerazione hardware delle schede grafiche.

I principali obiettivi considerati in fase di progettazione sono stati:

- Scalabilità: l'aggiunta di nuovi nodi alla rete deve essere semplice e le prestazioni del sistema non devono degradare
- Indipendenza dalla piattaforma: l'accesso al sistema deve essere possibile usando qualunque piattaforma (Windows, Unix/Linux, Mac, ecc.)
- Modularità: l'intero sistema è progettato come una serie di moduli, ognuno dei quali accessibile esclusivamente tramite un ben precisa interfaccia

Nel paragrafo 2 sarà presentata una panoramica sui lavori presentati in letteratura, mentre nel paragrafo 3 viene descritta brevemente la tecnologia JXTA. Il paragrafo 4 presenta l'infrastruttura di rete e il paragrafo 5 una descrizione dell'architettura del peer. Alcune considerazioni sui test eseguiti sono illustrate nel paragrafo 6. Infine, il paragrafo 7 presenta alcune conclusioni e i possibili sviluppi futuri.

Background

Uno dei primi lavori su un'architettura distribuita per la visualizzazione è stato presentato da Ang *et al.* [1]: VIS. VIS è uno strumento di visualizzazione che distribuisce la generazione di modelli 3D a partire da un volume di dati sull'hardware disponibile.

Engel *et al.* [2] usarono un desktop locale poco potente e un potente server remoto con hardware grafico per la visualizzazione interattiva di dati riguardanti tomografie, combinando tecniche di rendering ibride (in parte locali e in parte remote). In [3] viene presentato sistema basato su CORBA per la connessione tra PAMCRASH, un risolutore ad elementi finiti, e CrashViewer, un tool di visualizzazione. In [4] e [5] vengono realizzati due sistemi distribuiti per la visualizzazione e riproduzione sonora di dati, basati rispettivamente su CORBA e un protocollo XML.

Visapult è un tool per la visualizzazione parallela basato su componenti di tipo Grid e reti ad alte capacità per permettere la visualizzazione remota di grandi quantità di dati. In [6] viene inoltre descritto un sistema che migliora l'utilizzo della banda in Visapult.

In [7] viene presentato un sistema di visualizzazione platform-independent per visualizzare, tramite il tracciamento di particelle, l'evolversi dinamico di un fluido. Il sistema utilizza una architettura distribuita basata sulla tecnologia Jini

Negli ultimi anni si è sviluppata una nuova tipologia di architettura di rete: il peer-to-peer (P2P). Questa architettura è caratterizzata da un accesso diretto tra i differenti peer computer, piuttosto che attraverso un server centralizzato. In particolare un sistema P2P è differente dal tradizionale modello client/server perché le applicazioni coinvolte agiscono sia come client sia come server: possono contemporaneamente richiedere informazioni ad altri server e fornire dati ad altri client.

In generale è possibile classificare i sistemi P2P usando due aspetti: il grado di decentralizzazione e la struttura di rete. Un sistema P2P puro non ha server centrali [8]. Un P2P parzialmente centralizzato ha alcuni nodi che assumono un ruolo più "importante" rispetto agli altri, assumendo la funzione di indici per la ricerca. Alcune architetture di questo tipo sono Morpheus, (http://www.morpheussoftware.net/) e Kazaa (http://www.kazaa.com/). Infine, ci sono i P2P con server centralizzati (per esempio Napster).

Per quello che riguarda la struttura di rete, si possono avere reti completamente non strutturate dove la posizione dei dati è completamente scorrelata dalla topologia. Al contrario, nelle reti strutturate, (come CAN [9], Pastry [10], Chord [11] e altri) la topologia della rete è molto controllata e i dati sono disposti secondo un preciso schema. Ulteriori e più approfondite informazioni sulle topologie delle reti P2P possono essere trovate in [12].

Una breve descrizione della tecnologia JXTA

JXTA [13][14] è un progetto open-source che definisce un insieme di protocolli ad-hoc per il peer-to-peer. I protocolli definiscono una rete virtuale sulla stessa rete Internet o su reti non IP, permettendo ai peer di interagire direttamente e di organizzarsi indipendentemente dalla loro posizione sulla rete. L'architettura software del progetto JXTA è suddivisa in tre strati principali:

- Il Platoform Layer (JXTA Core). Il platform layer racchiude le primitive minime ed essenziali comune alle reti P2P. Esso include dei blocchi di costruzione per rendere disponibili i meccanismi chiave per le applicazioni P2P, includendo la scoperta, il trasporto (incluso il trattamento di firewall), la creazione di peer e gruppi di peer, e le primitive del sistema di sicurezza associato.
- The Services Layer. Il services layer include i servizi di rete che possono non essere assolutamente necessari affinché una rete P2P sia funzionante, ma sono di uso comune o desiderabili in un sistema P2P. Esempi di servizi di rete includono la ricerca e l'indicizzazione, servizi di directory, i sistemi di memorizzazione, la condivisione dei file, i sistemi di file distribuiti, l'autenticazione, e i servizi per il PKI (infrastruttura a chiave pubblica).
- The Applications Layer. L'Applications layer include l'implementazione di applicazioni integrate, come lo scambio di messaggi del P2P, la condivisione di documenti e risorse, la gestione e la diffusione di servizi di intrattenimento multimediale, un sistema di posta elettronica P2P e molti altri.

La rete JXTA è costituita da una serie di nodi interconnessi, o peer. I peer si possono organizzare dentro dei gruppi di peer, che forniscono un set comune di servizi. In particolare, JXTA indica due peer particolari, chiamati: Rendezvous e Edge. I peer Rendezvous possono essere usati per interconnettere gruppi diversi, separati in modo logico. Un peer Rendezvous agisce come una guida; tutti i peer Edge connessi ad un Rendezvous possono interagire, con peer non appartenenti al gruppo, soltanto spedendo e ricevendo messaggi attraverso un peer Rendezvous. I peer Rendezvous naturalmente permettono di costruire una topologia controllata (quando un peer è posto come Rendezvous, deve essere specificata una lista di altri peer Rendezvous da usarsi per assegnare delle connessioni dirette) in grado di gestire tutte le comunicazioni necessarie per il sistema proposto. Inoltre, in JXTA viene fornita anche una sorta di capacità di tolleranza dell'errore; per esempio, se cade un Rendezvous, tutti i peer Edge connessi ad esso possono essere automaticamente rediretti ad un altro Rendezvous preservando il funzionamento del sistema. In JXTA è anche implementato un algoritmo per cercare delle richieste inoltrate attraverso la rete JXTA dei Rendezvous.

Architettura della rete

Una descrizione generale di un sistema di visualizzazione distribuito può essere: *l'uso di uno o più computer per fornire ad uno o più utenti in immagine di set di dati*. Ci sono molti metodi per fare questo, come descritto in [15]:

- Visualizzazione parallela: il tempo necessario per una visualizzazione può essere ridotto usando architetture con CPU parallele.
- Visualizzazione di sorgenti multiple: i risultati sono combinati da sorgenti multiple di dati.
- Visualizzazione orientata alla rete: utenti che hanno il permesso di creare una visualizzazione a partire da dei dati su un server remoto.

C'è anche un altro tipo do visualizzazione distribuita, la visualizzazione d'insieme: questa è tipica nel caso di una squadra, in cui ogni membro sta guardando lo stesso soggetto, che è spesso manovrato da uno dei componenti della squadra.
La struttura proposta si basa su questo concetto: l'obiettivo principale è creare un ambiente virtuale in cui gli utenti possono incontrarsi, comunicare e condividere oggetti 3D. Ogni utente può muoversi ovunque (un meccanismo di scoperta delle collisioni evita scontri utente-oggetto e utente-utente), può comunicare usando una chat e può esaminare oggetti 3D usando tastiera e mouse. La performance degli ambienti di visualizzazione in 3D dipendono dalle connessioni di rete. Funkhouser [16] and Macedonia [17] descrivono la classificazione delle connessioni di rete e la topologia della rete. Principalmente le connessioni della rete possono essere suddivise in due gruppi: ci sono le connessioni basate sul server e quelle basate sul peer-to-peer. In un ambiente basato sul peer-to-peer, ogni client deve mantenere le connessioni per tutti gli altri client. Si può utilizzare un sistema multicast per trasmettere l'informazione ad un gruppo di client. Se la rete non è in grado di supportare il multicast, allora devono essere utilizzate connessioni singole: se l'ambiente dovesse supportare un elevato numero di client sarebbe necessario un numero di connessioni proporzionale alla quantità di client.

La struttura proposta va oltre questo limite, poiché usa il multicast per una rete locale e un collegamento singolo tra peer lontani.
L'architettura D3D (Distributed 3D) usa i due principali tipi di peer di JXTA: i Rendezvous e gli Edge (vedere paragrafo precedente).

In particolare, una tipica configurazione di rete deve essere costituita da un peer Rendevous per ogni rete locale (LAN) e possibilmente da molti peer Edge locali. JXTA usa un protocollo multicast per scoprire altri peer in una LAN, mentre in una rete geograficamente distribuita i peer Rendezvous agiscono come meccanismo di instradamento permettendo ai nodi posizionati in differenti LAN di comunicare.

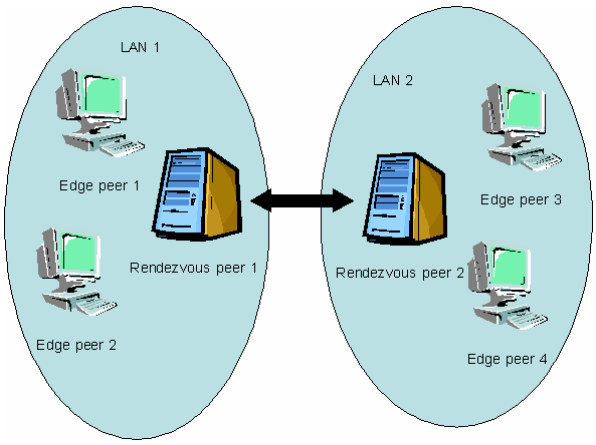

Figura 1: Una tipica rete D3D.

I peer Edge possono interagire con peer non appartenenti al gruppo (un gruppo è guidato almeno da un peer Rendezvous) solamente spedendo e ricevendo messaggi attraverso un peer Rendezvous. Come mostrato in Figura 1, dove è rappresentato un tipico scenario D3D, gli Edge peer in LAN 1 possono comunicare con ogni altro e con Rendezvous 1, mentre per comunicare con il peer Edge 3 e il peer Edge 4, devono usare il Rendezvous 1 che, comunicando con il Rendezvous 2, inoltra i messaggi ai peer di destinazione. I peer Rendezvous e i peer Edge differiscono solo per una configurazione del layer JXTA sottostante, ma dal punto di vista dell'utente, essi presentano la stessa interfaccia. Nella configurazione più semplice, il peer Rendezvous è il nodo che inizia la sessione di visualizzazione distribuita, mentre gli altri utenti entrano in questa sessione connettendosi al sistema come peer Edge.

L'architettura del peer

Il sistema è sviluppato per ottenere scalabilità e facilità di sviluppo. Per questo scopo l'applicazione è suddivisa in pacchetti che possono interagire attraverso interfacce. In particolare, i pacchetti principali, come mostrato in Figura 2, sono cinque:

1. GUI (Graphical User Interface): questo pacchetto contiene tutte le classi per gestire l'interfaccia grafica.
2. J3D: questo pacchetto contiene tutte le classi per gestire l'ambiente virtuale.

3. JXTA: questo pacchetto contiene tutte le classi necessarie per comunicare con i peer remoti. Contiene anche il servizio XMS descritto di seguito.
4. Background: questo pacchetto contiene altre classi per la gestione dell'applicazione o per alcuni controlli. Per esempio, contiene l'utility per il controllo della memoria usata.
5. Util: questo pacchetto contiene tutte le classi utili per l'interazione tra tutti i pacchetti.

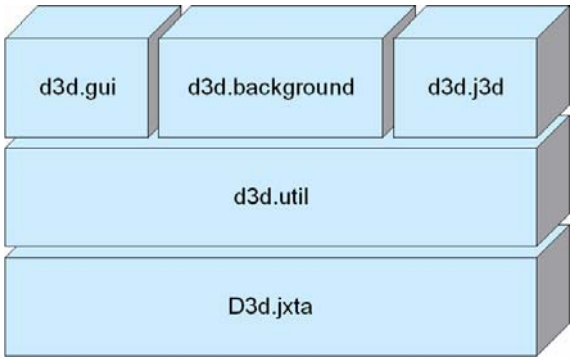

Figura 2: L'architettura del peer

Questa suddivisione dei pacchetti permette anche di supportare differenti formati di dati. L'output di alcune applicazioni può essere un oggetto grafico come una mesh di poligoni, mentre altre applicazioni potrebbero usare oggetti più sofisticati. Il pacchetto J3D può visualizzare differenti formati di file e, se fosse necessario supportare altri formati, nuovi moduli di software potrebbero essere aggiunti a questo pacchetto senza alcun cambiamento in altre parti dell'architettura.

JXTA fornisce il CMS (Content Management System) per gestire le risorse condivise in un ambiente distribuito. Incominciando dal CMS, un altro componente è stato creato per il sistema proposto: l'XMS (eXchange Management System). Questo componente, diversamente dal CMS che permette solo la condivisione dei file, fornisce lo scambio di messaggi e di oggetti 3D. In particolare l'XMS permette la condivisione delle risorse, lo scaricare oggetti 3D da peer remoti e la ricerca di peer (per trovare tutti i peer partecipanti ad una sessione di visualizzazione). Poiché non c'è una lista centrale di tutte le risorse disponibili per ogni peer, usando XMS, si deve mantenere e scambiare la lista delle proprie risorse. Inoltre, quando un peer vuole condividere un oggetto 3D, XMS fornisce un meccanismo di pipe per scaricare il modello. Infine, XMS, durante la fase iniziale, è responsabile di mandare tutte le informazioni a tutti gli altri peer della rete.

Ogni peer manda dei messaggi asincroni a tutti gli altri peer per comunicare ogni azione nell'ambiente virtuale (un movimento di un avatar o una roto-translazione di un oggetto caricato nella scena). Questo può causare un sovraccaricamento a causa della gestione di troppi messaggi. Per risolvere questo problema è stato sviluppato un componente aggiuntivo che gestisce una coda FIFO.

Questo gestore permette di ottenere due vantaggi:

- Il carico dei messaggi generato da ogni peer è indipendente dal numero di messaggi ricevuti. In questo modo un peer può occuparsi di altre operazioni, mentre il manager può processare i messaggi.

- La perdita di messaggi (e perciò la perdita di coerenza nel mondo virtuale) viene evitata.

Questo meccanismo causa un ritardo legato al sovraccarico della rete, ma questo inconveniente è compensato dall'affidabilità che introduce. Infine, per avere contemporaneamente una GUI funzionale e bella da vedere, il sistema offre sette principali aree per interagire con altri peer e per muoversi nell'ambiente virtuale.

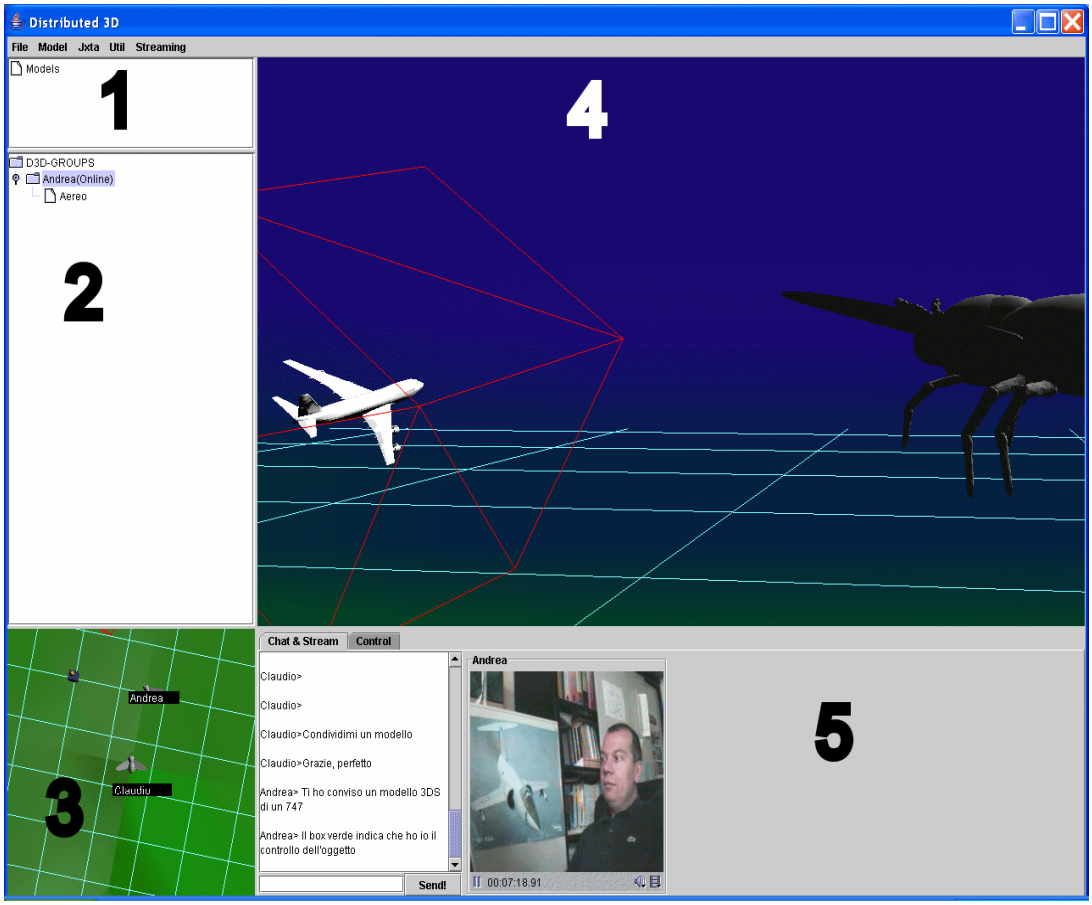

Figura 3: L'interfaccia grafica di D3D

Come mostrato in Figura 3, la finestra può essere suddivisa in 5 aree:

1. In questa area c'è una lista dei modelli 3D caricati. L'utente può gestire ed ottenere informazione usando un menu popup. Nell'esempio di Figura 3 gli utenti sono rappresentati da modelli a forma di vespa.
2. In questa area sono mostrati tutti i peer connessi, così come i modelli condivisi. In Figura 3 "Andrea" condivide un modello di aereo Boeing 747. L'utente, che sta manipolando un oggetto vede l'oggetto stesso incapsulato in un cubo verde; allo stesso tempo, gli altri utenti vedono lo stesso modello incapsulato in un cubo rosso, come in figura. Un semplice click con il mouse sul modello permette di ottenere il controllo. In Figura 3 "Andrea" controlla il modello.

3. In questa area viene presentata una mappa dell'ambiente virtuale. Questa è molto utile nei mondi virtuali molto grandi per trovare facilmente la collocazione di oggetti e utenti.
4. Questa area mostra il punto di vista peer/user. L'utente può interagire usando la tastiera per muovere l'avatar e il mouse per gestire gli oggetti 3D. Un meccanismo di controllo della collisione evita che l'utente possa penetrare oggetti o altri utenti.
5. In questa area ci sono due pannelli di tipo tab: il primo è per il sistema di chat, mentre il secondo è per controllare alcuni parametri: la posizione verticale dell'avatar, il fattore di zoom, l' introduzione di un nuovo oggetto attraverso un'animazione, e così via. Il sistema di chat incorpora sia un meccanismo di scambio di messaggi tra gli utenti partecipanti alla sessione sia un meccanismo di video conferenza vero e proprio. Un utente può scegliere di ricevere, da parte di un altro partecipante alla sessione, il flusso video, il flusso audio o entrambi, a seconda di quelle che sono le capacità di elaborazioni locali. Il sistema permette di visualizzare contemporaneamente il flusso video di tre utenti remoti, selezionabili dalla lista di tutti i partecipanti correntemente connessi, più, in una finestra separata, il flusso video prodotto dalla webcam locale. Negli esperimenti realizzati si è evidenziata l'importanza di avere dei microfoni con un sistema di soppressione dell'eco e, nel caso di un numero elevato di partecipanti, di rete veloci al fine di avere un'elevata qualità del servizio. Per realizzare il meccanismo di videoconferenza è stata adottata la libreria Java Media Framework (JMF) che permette di incorporare in applet e applicazioni Java audio, video e altri dati di tipo time-based. La libreria JMF utilizza il protocollo RTP (Real-Time Protocol) per il trasferimento dei dati multimediali; in particolare, i dati provenienti da una sorgente possono essere riprodotti localmente, come nel caso dell'applicazione proposta, e/o salvati su file.

Note

La valutazione delle performance delle architetture distribuite è un compito estremamente critico e difficile. Si deve prendere in considerazione un insieme di parametri; in particolare, per un sistema di visualizzazione distribuito basato sul P2P, dobbiamo considerare:

- I tempi di risposta
- La scalabilità
- La realizzabilità

Sono stati eseguiti diversi test. Il primo considera una rete che include solo peer posizionati sulla stessa LAN; in particolare, un peer è rappresentato come un Rendezvous connesso a quattro peer Edge. I ritardi nelle comunicazioni sono quasi impercettibili, mentre i problemi di realizzabilità sorgono nel caso del crollo di un Rendezvous. In questo caso, gli utenti connessi dai peer Edge sono ancora in grado di muoversi dentro il mondo virtuale, ma tutti i servizi (condivisione di file e così via) supportati da XMS (vedi la Sezione sull'architettura del peer) non funzionano più.

Il secondo test utilizza una configurazione dove tre peer Rendezvous sono connessi in una rete geograficamente distribuita nel nord-est dell'Italia settentrionale (vedi la Figura 4, in cui il

Rendezvous di Torino (1) conosce il Rendezvous di Vercelli (2) e di Ivrea (3), ma i Rendezvous 2 e 3 non si conoscono direttamente. Tutti i messaggi si propagano senza loop e, inoltre, nel caso del crollo del Rendezvous 1, dopo un timeout, i Rendezvous 2 e 3 sono in grado di cominciare una connessione saltando il Rendezvous 1. Questo esempio mostra come parti diverse di una complessa rete distribuita possono superare i crolli di nodi Rendezvous intermedi.

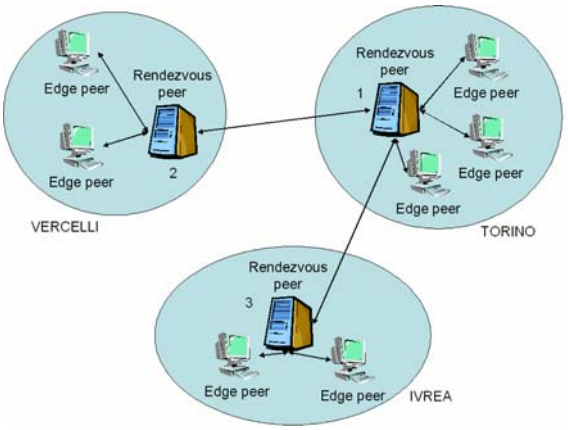

Figura 4: La rete del secondo test

Il terzo ed ultimo test utilizza una rete complessa con molti peer Rendezvous ed Edge. I tempi di risposta dipendono da diversi parametri, ma, in particolare, dalla larghezza di banda delle connessioni di rete.

I tempi di risposta dovuti ai comandi della roto-translazione possono variare da decimi di secondi ad alcuni secondi, mentre la pubblicazione di un modello (gli utenti devono scaricare localmente il modello per essere in grado di visualizzarlo), dipende certamente dalla dimensione del file che deve essere trasmesso. D'altro canto, la realizzabilità di un sistema non è completamente garantita dai meccanismi di tolleranza all'errore forniti da JXTA.

Per questa ragione, è stata sviluppato un meccanismo di monitoraggio del peer:

Ogni peer manda a intervalli di tempo regolari un messaggio is_alive a tutti gli altri peer. Quando un peer cade, tutti gli altri nodi non ricevono il suo messaggio is_alive e possono riconfigurarsi:

- Se il crollo riguarda un Rendezvous, tutti i peer Edge connessi a quel Rendezvous troveranno un altro peer Rendezvous che li "adotta".
- Nel caso in cui il crollo riguardi l'unico peer Rendezvous della rete, un peer Edge può essere scelto come nuovo Rendezvous da un meccanismo automatico fornito da JXTA.

Infine, dal punto di vista della scalabilità, JXTA può offrire il supporto per interconnettere centinaia di peer (maggiori dettagli riguardo alla performance di JXTA si possono trovare all'indirizzo: http://bench.jxta.org).

I nostri test usano dieci peer e in questi scenari non abbiamo avuto esperienza di problemi di scalabilità.

Conclusioni e lavori futuri

L'architettura proposta si basa sulla tecnologia JXTA per realizzare architetture di rete distribuite e sulle librerie di Java3D per visualizzare e gestire oggetti 3D. Gli utenti possono incontrarsi dentro al mondo virtuale rappresentati da un avatar; inoltre possono comunicare usando un sistema di chat direttamente supportato da JXTA e un sistema di video conferenza.

D3D può essere usato come una piattaforma per realizzare i servizi per la Computer Graphics in 3D. D3D permette agli utenti di condividere una sessione di visualizzazione distribuita dove ogni partecipante può caricare e condividere i propri modelli. Quando un utente carica un oggetto, può esaminarlo (tutte le operazioni fatte, sono tradotte in messaggi e spedite agli altri peer/utenti che vedranno le stesse azioni) o concedere il controllo.

L'architettura proposta è indipendente dalla piattaforma così come tutte le tecnologie usate sono disponibili per i principali sistemi operativi e le configurazioni dell'hardware; inoltre, tutto il software è gratuito.

Bibliografia

[1] C. S. Ang, D. C. Martin, and M. D. Doyle. Integrated control of distributed volume visualization through the world-wide-web. In Proceedings of IEEE Visualization' 94, pages 13–20, 1994.

[2] K. Engel, P. Hastreiter, B. Tomandl, K. Eberhardt, and T. Ertl. Combining local and remote visualization techniques for interactive volume rendering in medical applications. In Proceedings of IEEE Visualization' 00, pages 449–452, 2000.

[3] N. Frisch and T. Ertl. Embedding visualization software into a simulation environment. In Proceedings of SCCG 2000, 2000.

[4] R. Minghim, V. C. L. Salvador, B. S. Freitas, M. C. F. Oliveira, and L. G. Nonato. Distributed sound for volumes-data analysis using distributed visualization and sonification. In Proceedings of SPIE – Visualization and Data Analysis 2002, volume 4665, pages 379–390, January 2002.

[5] V. C. L. Salvador, R. Minghim, and H. Levkowitz. DSVOL II - a distributed visualization and sonification application communicating via an xml-based protocol. In Proceedings of the XV Brazilian Symposium on Computer Graphics and Image Processing, 2002.

[6] E.W. Bethel and J. E. Shalf. Cactus and visapult: An ultra-high performance grid-distributed visualization architecture using connectionless protocols. IEEE Computer Graphics and Applications, 23(2):51–59,2003.

[7] C. Zunino, B. Montrucchio, A. Sanna, and C. Demartini. A distributed visualization environment for scientific visualization based on Jini technology. In IEEE Proceedings of SCCG'2001, pages 95–101, 2001.

[8] I. Clarke, O. Sandberg, and B.Wiley. Freenet: A distributed anonymous information storage and retrieval system. In Proceedings of the Workshop on Design Issues in Anonymity and Unobservability, 2000.

[9] S. Ratnasamy, P. Francis, R. Handley, M.and Karp, and S. Shenker. A scalable content-addressable network. In ACM SIGCOMM, pages 161–172, 2001.

[10] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems. In Proceedings of the IFIP/ACM International Conference on

Distributed Systems Platforms, 2001.

[11] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In ACM SIGCOMM, pages 149–160, 2001.

[12] S. Androutsellis-Theotokis. A survey of peer-to-peer file sharing technologies. White paper, Athens University of Economics and Business, 2002.

[13] S. Botros and S.Waterhouse. Search in jxta and other distributed networks. In Proceedings of the First International Conference on Peer-to-Peer Computing, pages 30–35, 2001.

[14] L. Gong. Search in jxta and other distributed networks. IEEE Internet Computing, 5:30–35, 2001.

[15] J. Heijmans. An introduction to distributed visualization. Technical report.

[16] T. Funkhouser. Network topologies for scalable multi-user virtual environments. Proceedings of VRAIS'96, Santa Clara CA, pages 222–229, 1996.

[17] Michael R. Macedonia and Michael J. Zyda. A taxonomy for networked virtual environments. IEEE MultiMedia, 4(1):48–56, 1997.