Tecniche di merging per VoD

Sistemi di Elaborazione 3

Dipartimento di Scienze dell'Informazione

Università di Bologna

Ricci Luca

Uburti Luca

Scenario

- Server Video on Demand contenente video disponibili per essere visualizzati dagli utenti collegati in rete che ne fanno richiesta.
- Utilizzo del multicast per servire piú client richiedenti lo stesso video in modo da contenere la banda utilizzata dal server.

Obiettivi

Fattore limitante: larghezza di banda richiesta al server.

Obiettivi:

- Costruire un sistema VoD che riduca la larghezza di banda attraverso una politica di merging dei flussi.
- Utilizzo del multicast al termine della fusione degli stream.

Approcci non ottimali (1/2)

Broadcast Periodico

Idea: i video sono trasmessi in multicast periodicamente; un nuovo flusso viene trasmesso ogni B minuti (intervallo di batching).

Svantaggi: nel caso pessimo la latenza é di B minuti.

Approcci non ottimali (2/2)

Scheduled Batching

Idea: le richieste per ciascun video vengono accodate.

Il server trasmette in multicast uno dei batch secondo una politica di scheduling. Ad esempio puó essere scelto il batch contenente il maggior numero di richieste pendenti.

Svantaggi: latenza potenzialmente lunga.

Svantaggi del batching

I due sistemi presentati forniscono un modo semplice per minimizzare la larghezza di banda richiesta al server.

Tuttavia entrambe le tecniche di batching aumentano la latenza per il client.

Se sovrautilizzati sono inadatti per sistemi di VoD.

Piggybacking (1/4)

Obiettivo: determinazione di uno stream merging ottimale.

Perchè farlo?

- Ridurre la larghezza di banda richiesta al server.
- Non aumentare il tempo di latenza per il client.

Presupposto: possibilità di alterare in maniera non avvertibile per l'utente la velocità di visualizzazione dei video.

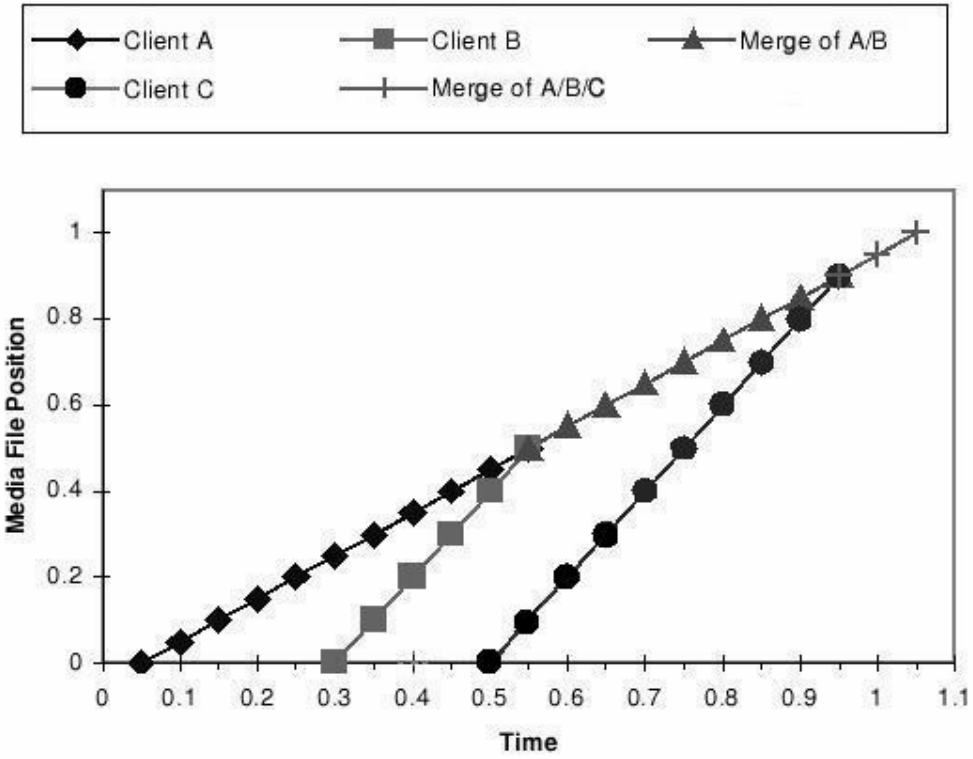
Piggybacking (2/4)

Idea: se due client vogliono visualizzare lo stesso video, verrà inviato il flusso secondario ad una velocità lievemente maggiore, rispetto a quella del normale playback.

Se l'intervallo di visualizzazione tra i due è sufficientemente piccolo, lo stream più veloce raggiungerà lo stream più lento. I flussi possono quindi essere fusi (piggybacked). Da questo momento in poi verrà visualizzato un solo flusso alla normale velocità di playback.

Il flusso raggiunge entrambi i client tramite multicast.

Piggybacking (3/4)



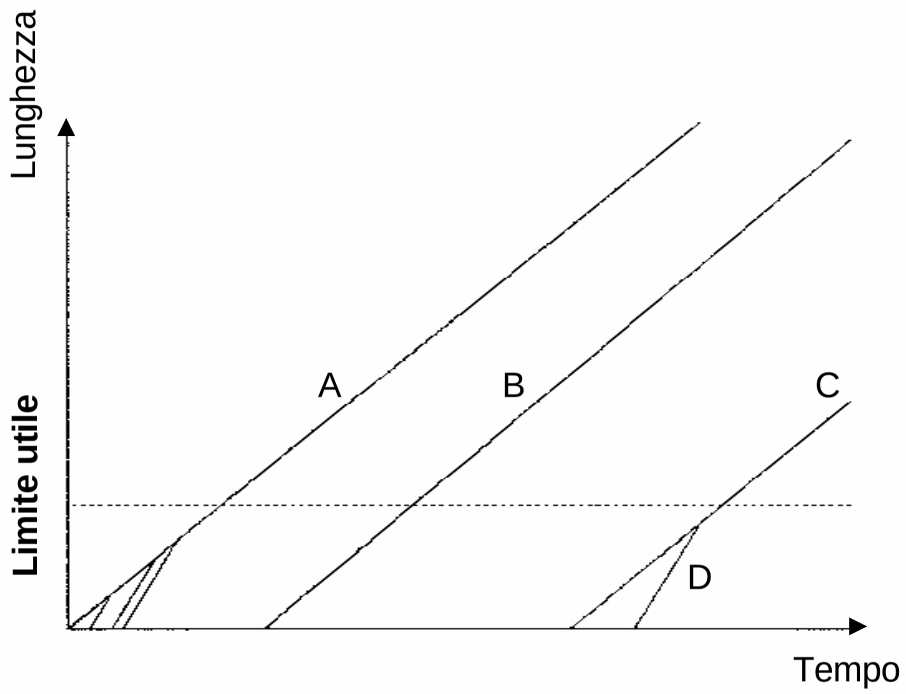
Il Piggybacking effettua merging multipli portando ad una struttura di merging rappresentabile con un albero binario.

Il costo del merge tree, misurato nella quantità totale di dati che il server deve trasmettere per soddisfare un insieme di client, è la somma della proiezione dei segmenti contenuti sull'asse x.

Esiste un limite oltre il quale non è possibile ottenere buone prestazioni.

Piggybacking (4/4)

Limite Prestazionale



L'intervallo di tempo che intercorre tra due flussi non deve essere superiore all'ordine di una percentuale della lunghezza totale del video.

Il flusso del client che per secondo vuole visualizzare un certo video, potrá fondersi con il primo solo se ne fa richiesta entro tale tempo, in caso contrario non sará conveniente effettuare il merging.

Piggybacking + Batching

Il Piggybacking costituisce una valida alternativa al Batching ed allo stesso tempo é possibile utilizzarlo congiuntamente a tale tecnica, a patto peró di non scegliere politiche di Batching troppo onerose nei confronti del tempo di attesa medio.

Il Piggybacking tuttavia abbraccia in maniera piú stretta lo spirito alla base del VoD.

Implementazione

Realizzazione di un sistema client/server che sfrutti la tecnica del Piggybacking.

Server: contiene il DB dei video disponibili e fornisce servizio di VoD ai client.

Client: deve scaricare un apposito sw per la connessione al server.

Il trasferimento dei video avviene tramite protocolli RTP/RTCP.

Primo collegamento al server

Gli utenti che intendono usufruire del servizio di VoD dovranno collegarsi alla pagina web di riferimento per scaricare un apposito software.

Tale software di connessione permette il collegamento al DB dei video disponibili sul server ed integra un lettore multimediale.

E’ prevista una procedura di registrazione che richiede la scelta da parte dell’utente di un username e password per il collegamento al server.

Utilizzo del SW

Il client attraverso il software richiede una connessione al server.

Il server lo identifica attraverso username e password.

Una volta autenticato il client puó eseguire una ricerca sul database dei video disponibili secondo alcuni criteri di ricerca (nome del video, durata, bit rate, risoluzione del video, ecc...).
L'utente non dovrà fare altro che selezionare il video di interesse ed avviarne la visualizzazione.

Funzionamento del SW

La fase di visualizzazione avviene tramite il lettore integrato nel software.

Le tecniche di Piggybacking e merging sono implementate sul server e sono il piú possibile trasparenti nei confronti dell'utente.

Ogni client dispone di un buffer di alcuni secondi per poter tollerare variazioni sulle prestazioni della rete.

Realizzazione

I protocolli utilizzati sono RTP e RTCP.

Software sviluppato in Java con l'ausilio delle API JMF (Java Media Framework) per semplificare le fasi di pacchettizzazione, trasmissione e ricezione del video via RTP e la fase di visualizzazione del flusso ricevuto.

Considerazioni finali

Scelta del codec problematica.

Implementazione di controlli VCR non banale.

Requisiti per sviluppare VoD non facilmente ottenibili in laboratorio.

Bibliografia

- C. C. Aggarwal, J. L. Wolf e P. S. Yu, "On Optimal Piggyback Merging Policies for Video−on−Demand Systems", Proc. ACM SIGMETRICS Conf. On Measurement and Modeling of Computer Systems, Philadelphia, May 1996, pp. 200−209.
- D. L. Eager, M. K. Vernon e J. Zahorjan, "Optimal and Efficient Merging Schedules for Video−on−Demand Servers", Proc. 7th ACM Int'l. Multimedia Conf. (ACM Multimedia '99), Nov. 1999, pp. 199−202.
- D. L. Eager, M. K. Vernon e J. Zahorjan, "Bandwidth Skimming: A Technique for Cost−Effective Video−on−Demand", Proc. IS&T/SPIE Conf. on Multimedia Computing and Networking (MMCN 2000), Jan. 2000. pp. 1−10.
- http://www.java.sun.com/jmf