Accelerated Chaining: A Better Way to Harness Peer Power in Video-on-Demand Applications

Jehan-François Pâris
Department of Computer Science
University of Houston
Houston, TX 77204-3010
+1 713-743-3341
paris@cs.uh.edu

Ahmed Amer
Department of Computer Engineering
Santa Clara University
Santa Clara, CA 95053
+1 408-551-6064
a.amer@acm.org

Darrell D. E. Long
Department of Computer Science
University of California
Santa Cruz, CA 95064
+1 831-459-2616
darrell@cs.ucsc.edu

ABSTRACT

We present a more efficient chaining protocol for video-on-demand applications. Chaining protocols require each client to forward the video data it receives to the next client watching the same video. Unlike all extant chaining protocols, our protocol requires these clients to forward these data at a rate slightly higher than the video consumption rate. Our simulations indicate that increasing the client video forwarding rate by 5 percent is sufficient to virtually eliminate the server workload for a two-hour video when request arrival rates remain above 30 requests per hour

Categories and Subject Descriptors

H.5.1 [**Information Systems**] INFORMATION INTERFACES AND PRESENTATION — Multimedia Information Systems.

General Terms

Algorithms, Performance, Design.

Keywords

P2P, Streaming, Video-on-demand.

1. INTRODUCTION

One of the major impediments to the wider development of video-on-demand (VOD) services is their high bandwidth requirements. Assuming that the videos are in MPEG-2 format, each user request will require the delivery of around six megabits of data per second. Hence a video server allocating a separate stream of data to each request would need an aggregate bandwidth of six gigabits per second to accommodate one thousand concurrent clients.

This situation has led to numerous proposals aimed at reducing the bandwidth requirements of VOD services. These proposals can be broadly classified into two groups. Proposals in the first group assume that servers can *multicast* data to several clients at the same time. For instance, *batching* [6] delays requests for a popular video in order to form batches of requests that can be serviced with a single video stream. *Pyramid broadcasting* [16] and *stream tapping* [4] are more sophisticated examples of the same approach. While these schemes can provide very impressive bandwidth savings, they only apply to networks supporting multicast, which is not the case for the vast majority of systems on the Internet [1].

Proposals in the second group require clients to participate in the video distribution process [3, 17]. In essence, their approach is the same as that of *peer-to-peer* (P2P) file sharing systems such as Gnutella [7] or BitTorrent [5]. By letting clients serve each other, P2P solutions overcome many limitations of traditional client-server architectures. They can handle flash crowds (that is, very large and sudden surges of demand) as well as overcome the bandwidth limitations of the server. In addition, P2P solutions do not require any special support from the network, be it IP multicast or any specific content distribution infrastructure. At the same time, they present some important differences from P2P file sharing systems [2, 12].

First, existing file sharing systems do not account for the real-time needs of streaming applications. As they do not download video data in sequence, these data remain unusable until the download is complete.

Second, these real-time needs require unidirectional data transfers among clients in addition to data exchanges between pairs of clients: the clients that are already watching the video will forward their video data to more recently arrived clients without receiving any video data from them. As a result, tit-for-tat policies clearly do not apply.

Chaining [13] is the oldest P2P protocol for VOD. It organizes all clients watching a video into chains where each client forwards the video data it receives to the next client in the chain. Since the original protocol assumed that clients did not have enough buffer space to store an entire video, it did not perform as well at low request arrival rates as at higher rates. While this limitation has been addressed in two more recent protocols [8, 11], the actual performance of all chaining protocols is still affected by client departures. Experience has shown that most clients will disconnect and stop forwarding data once they have finished playing the video. As a result, the next client will have to receive the missing data from the server. *Accelerated chaining* addresses these concerns by requiring clients to forward their video data at a rate slightly higher than their video consumption rate. As we will see, even very small increases of this forwarding

SAC'11, March 21-25, 2011, TaiChung, Taiwan.

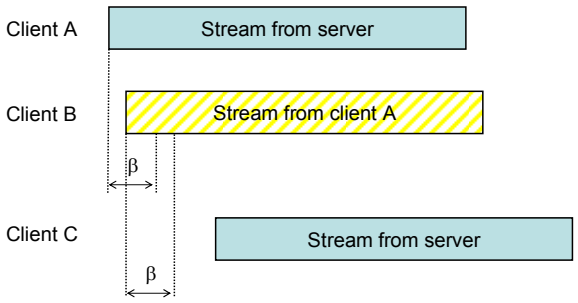

Figure 1. How chaining works.

rate have a dramatic impact on the server workload. To give an example, increasing the peer forwarding rate by as little as 5 percent will virtually eliminate the server workload for a two-hour video whenever the request arrival rate exceeds 30 requests per hour.

The remainder of the paper is organized as follows. Section 2 reviews previous work on chaining protocols. Section 3 introduces our protocol, Section 4 discusses its performance and Section 5 investigates an alternative implementation for small portable devices. Finally, Section 6 offers our conclusions.

2. PREVIOUS WORK

For brevity's sake, we will focus our discussion on the distribution protocols that are directly relevant to our work, and refer the reader to the work of Liu *et al.* [9] for a more general survey.

Standard chaining [13] constructs chains of clients such that (a) the first client in the chain receives its data from the server and (b) subsequent clients in the chain receive their data from their immediate predecessor. As a result, video data are in some way "pipelined" through the clients belonging to the same chain. Since standard chaining only requires clients to have very small data buffers, a new chain has to be restarted every time the time interval between two successive clients exceeds the capacity β of the buffer of the first client. Figure 1 shows three sample customer requests. Since customer *A* is the first customer, it will get all its data from the server. As customer *B* arrives less than β minutes after customer *A*, it can receive all its data from customer *A*. Finally customer *C* arrives more than β minutes after customer *B* and must be serviced directly by the server.

The main weakness of standard chaining is its poor performance at low arrival rates, that is, whenever the mean time between consecutive requests exceeds β minutes. *Extended chaining* [13] takes into account the maximum time clients are willing to wait and delays some requests in order to produce chains of requests where consecutive requests are separated by less than β minutes. *Advanced chaining* [8] proposes to bridge the gap between requests separated by more than β minutes by inserting idle peers that will relay the data. *Optimal chaining* [14, 15] addresses the same issue by managing all client buffers as a single shared resource. As a result, clients can "borrow" the buffers of other clients in order to bridge gaps between incoming requests. The protocol can also integrate streaming proxies in order to increase chain responsiveness and resiliency.

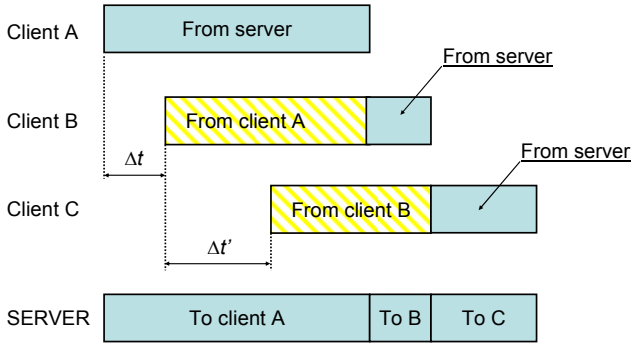

Figure 2. How the cooperative protocol works.

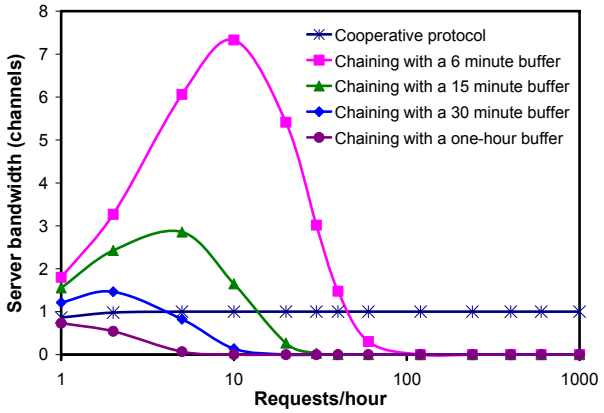

Figure 3. Comparing the required server bandwidths of the cooperative protocol with that of chaining at various buffer sizes.

The *cooperative* video distribution protocol, also known as *expanded chaining* [11] extends the chaining protocol by taking advantage of the larger buffer sizes of modern clients and requires them to keep in their buffers the previously watched portion of the video. At the same time, it assumes that clients will disconnect and stop forwarding data once they have finished playing the video. In the cooperative protocol, each client forwards to its immediate successor video data starting with the beginning of the video at the same rate it consumes them. When a client has finished playing a video, it will disconnect itself and will stop forwarding video data to its successor in the chain, thus forcing the server to transmit the missing part of the video.

Consider for instance how the protocol would handle the three requests displayed in Figure 2 for a video of duration *D*. The first request to the video will be entirely serviced by the server. Since client B's request arrives while the first request is still being serviced, the server will thus instruct client *A* to forward the first $D - \Delta t$ minutes of the video to the client *B* and schedule a transmission of the last Δt minutes of the video to the same client at a later time. When client C's request arrives, the server will similarly order client *B* to forward the first $D - \Delta t'$ minutes of the video to the client *C* and schedule a transmission of the last $\Delta t'$ minutes of the video to the same client.

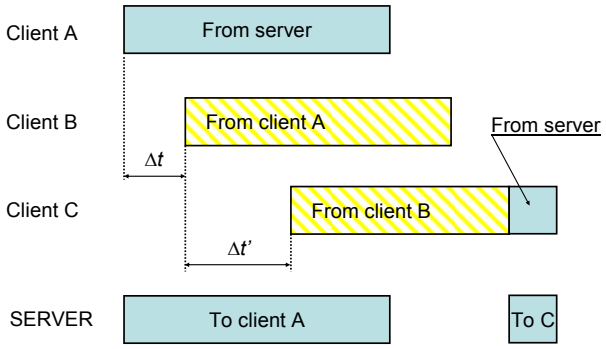

Figure 4. How accelerated chaining works.

More generally, the amount of time spent by the server to service a request will always be given by min(D, Δt), where D is the duration of the video and Δt is the time interval between the request being serviced and its immediate predecessor. Since the service times of these requests will never overlap, the server instantaneous bandwidth will never exceed the video consumption rate.

3. ACCELERATED CHAINING

Figure 3 compares the server bandwidth requirements of the cooperative protocol with that of the chaining protocol for various buffer sizes. The x-axis indicates the request arrival rates in requests per hour on a logarithmic scale while the y-axis expresses the server bandwidths in multiples of the video consumption rate. The video duration was assumed to be two hours, which is not far from the average duration of a full-length movie.

As we can see, the cooperative protocol performs much better than chaining at low-to-medium request arrival rates. This should be expected because the chaining protocol must restart a new chain every time two consecutive requests are separated by more than β minutes.

Conversely, chaining performs better than the cooperative protocol at high arrival rates. In fact, the server workload is completely eliminated whenever the time interval between two consecutive requests remains below β minutes. We must however observe that this excellent performance is based on the assumption that clients will always keep forwarding data to their successor in the chain after they have finished playing the video. We do not believe that clients will do so and assume instead that most clients will disconnect once they have played the video. In addition, a significant number of clients will disconnect without having played the full video.

Our *accelerated chaining* protocol overcomes this limitation by requiring clients to forward video data to their successor in the chain at a slightly higher rate than the video consumption rate, say, between one and ten percent faster.

Let b denote the video consumption rate and $b_a > b$ the accelerated video forwarding rate. We define the forwarding acceleration factor f of the video as

$$f = b_a / b .$$

Forwarding a video of duration D at the accelerated video forwarding rate b_a will then take D/f time units.

Consider now a pair of consecutive clients that are separated by a time interval Δt. The second client will be able to get the entire video from the first client as long as

$$D/f \le D - \Delta t ,$$

that is, as long as

$$\Delta t \le D\frac{f-1}{f} , \quad (1)$$

or

$$f \ge \frac{D}{D-\Delta t} , \quad (2)$$

Consider for instance how the protocol would handle the three requests displayed in Figure 3 for a video of duration D. The first request to the video will be entirely serviced by the server. Assuming that the time interval Δt between the first and the second request satisfies Equation (1), client B will receive all its video data from client A. Assuming that the time interval $\Delta t'$ between the second and the third request does not satisfy Equation (1), the server will instruct client B to forward the first $f(D-\Delta t')$ minutes of the video to client C and schedule a transmission of the last $D - f(D-\Delta t')$ minutes of the video to the same client.

More formally, let us consider a video of duration D and a request for that video from a customer C arriving at the server at time t. Let Δt denote the time interval between that request and the last request for the same video and let B designate the customer who issued that last request. Define

$$\Delta t^* = D\frac{f-1}{f} ,$$

where f is the video acceleration factor.

Accelerated chaining will operate in the following fashion:

1. If $\Delta t \ge$ D, there is no overlap between the two requests; the server will then initiate a transmission of the full video, starting at time t and ending at time $t+D$.
2. If $\Delta t \le \Delta t^*$, there is a sufficient overlap between the current request and the previous request to allow client C to get all its video data from client B.
3. If $\Delta t^* < \Delta t < D$, client C will receive the first $f(D-\Delta t)$ minutes of that video from client B and the last $D - f(D-\Delta t)$ minutes of the video directly from the server. This transmission will start at time $t + f(D-\Delta t)$ and end at time $t+D$.

3.1 Implementation Issues

While accelerated chaining greatly reduces the server workload at high arrival rates, is does not eliminate it. The server will mostly act as a dispatcher instructing clients when and where to forward video data and at which rate.

Another important server task will be handling client disconnection. To operate correctly, our protocol requires all clients in a chain to forward the video data they have received to the next customer in the chain. As a result, any client disconnection will deprive all subsequent clients from their video data. This is clearly not an acceptable state of affairs.

There is a simple solution to the problem. Consider for instance the scenario of Figure 4 where client *C* receives most of its video data from client *B*, which receives all its video data from customer *A*. Should client *B* stops forwarding data to client *C,* client *C* will immediately notify the server. In such a situation, one of the following two cases would apply:

1. If the disconnection happens while client *A* is still playing the video, the server will order client *A* to stop forwarding data to client *B* and forward them instead to client *C*. As a result, client *C* will get the first $f(D-(\Delta t+\Delta t'))$ minutes of the video from clients *A* and *B*. In addition, the server will cancel its scheduled transmission to client *B* and replace it by a transmission of the last $D-f(D-(\Delta t+\Delta t'))$ minutes of the video to client *C*.
2. If the disconnection happens after client *A* has finished playing the video, the server will cease transmitting any data to client *B* and start transmitting the missing video data to client *C*.

3.2 Implementing interactive controls

Requiring client buffers to be able to store an entire video greatly simplifies the implementation of *pause* and *rewind* video controls as they can be implemented in the client without any server intervention. Activating these commands will indeed improve the performance of the system as they will increase the length of time the client will remain connected and able to forward data. Once the client has forwarded the entire contents of the video to the next client, it would then remain available to act as a seed and help the server handle fast forward requests.

4. PERFORMANCE EVALUATION

To evaluate the performance of our protocol we wrote a simple simulation program assuming that request arrivals for a particular video were distributed according to a Poisson law. Our program was written in C and simulated requests for a single two-hour video. Simulation durations were selected in a way that guaranteed that each run simulated a minimum of 10,000 hours of simulated time and a minimum of 100,000 request arrivals.

Since no data are shared among customers watching different videos, the total bandwidth of a server distributing several videos would always equal the sum of the bandwidths it dedicates to each video.

We considered four possible values for the video acceleration factor *f*, namely 1.01, 1.02, 1.05 and 1.10. We measured the average server bandwidth at request arrival rates varying between one and one thousand requests per hour. We did not consider higher arrival rates as they seemed unrealistic.

Our results are summarized in Figures 5 and 6. Request arrival rates are expressed in arrivals per hour and all bandwidths are expressed in multiples of the video consumption rate.

As we can see, accelerated chaining performs much better than the cooperative protocol at all medium to high request arrival rates. This is to say that even a small forwarding acceleration factor will eliminate most of the server workload at high arrival rates while an acceleration factor of at least 1.10 is required to have a significant impact on the server workload at low to medium arrival rates, that is, at less than ten requests per hour.

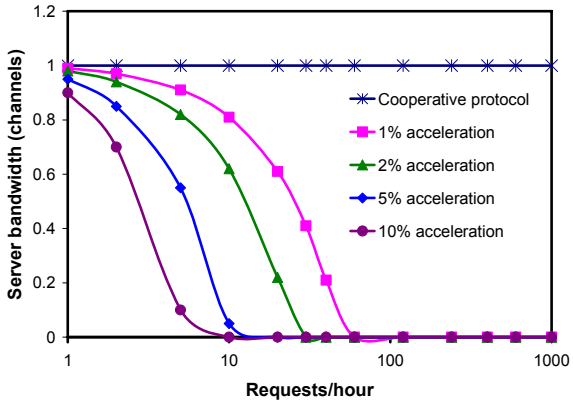

Figure 5. Server bandwidth requirements of the accelerated chaining protocol for selected values of the acceleration factor.

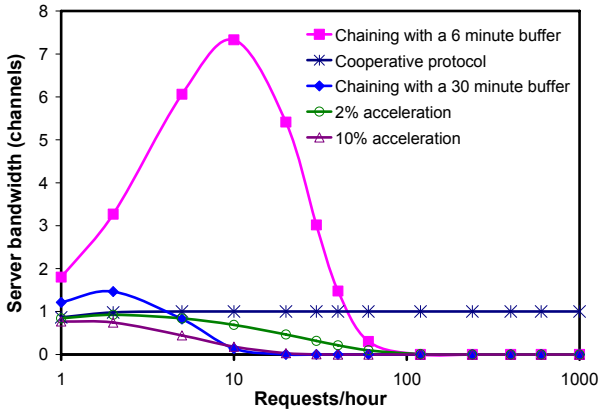

Figure 6. Server bandwidth requirements of the three protocols.

These savings are significant because the server will still have to manage client arrivals and departures and this workload will increase linearly with the request arrival rate. As the server becomes busier with such administrative aspects of its workload, reducing its video distribution burden become more critical.

5. HANDLING SMALL DEVICES

Not all small portable devices are able or willing to keep in their buffers more than a few minutes of video, say β minutes. As it was the case with the standard chaining protocol, this limitation forces the server to initiate a transmission of the full video each time that two successive requests are separated by more than β minutes. We should thus expect to see a significant increase of the server workload at low arrival rates and no impact whatsoever at high arrival rates.

Figure 7 displays the average server bandwidth at various arrival rates and various buffer sizes. We assumed the same two-hour video duration as we did before and selected a video acceleration factor of 1.05. As we expected, client buffer size limitations do not impact the server's workload at high request arrival rates and have only a limited impact at medium arrival rates. The situation is quite different at low arrival rates because clients that can only store β minutes of video cannot effectively participate in the video

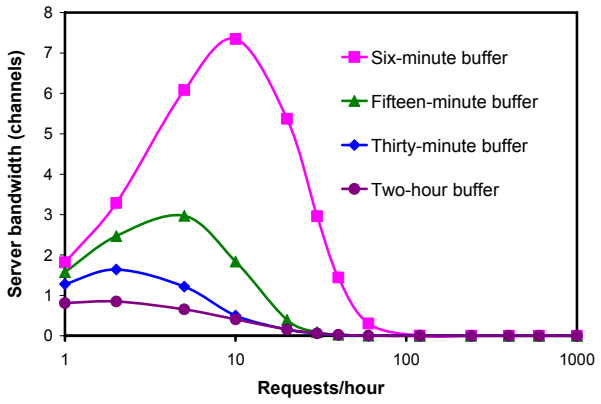

Figure 7. Server bandwidth requirements of the accelerated chaining protocol when clients cannot store entire videos in their buffers.

distribution process when too many consecutive requests are separated by more than β minutes. As a result, clients with the smallest buffers will make the highest demands on the server's workload at low arrival rates. Conversely, clients that can store at least 30 minutes of video, that is, 25 percent of the video duration have a much more moderate impact. We can thus safely conclude that we could easily integrate clients with these buffer sizes in the chaining process.

This is not to say that the integration process would be fully transparent. For one, we would lose the ability to implement pause and rewind controls at the client level. While these two commands had previously a beneficial impact on the server's workload, they will now require contingent streams coming directly from the server and break the current chain of requests.

A more difficult issue is raised by the uploading bandwidth limitations of small devices. Any self-sustaining P2P system requires that the aggregate forwarding bandwidth of all its peers remains equal to their aggregate receiving bandwidth. This will be difficult to achieve without drastically reducing the video transmission quality or assuming the presence of a large number of selfless peers acting as seeds.

6. CONCLUSIONS

We have presented an accelerated chaining protocol for video-on-demand applications that makes the modest assumptions that all clients can

1. Store in their buffer the entire contents of the video they are playing; and
2. Forward to their successor in the chain video data at a rate slightly higher than the video consumption rate, say, one to five percent faster.

At the same time, our protocol does not require clients to keep forwarding video data once they have finished playing the video and handles premature disconnections.

Our simulations indicate that increasing the client video forwarding rate by 5 percent virtually eliminates the server workload for a two-hour video whenever the request arrival rate exceeds 30 requests per hour.

More work is still needed to define the best implementation of the fast forward control and specify an incentive mechanism that motivates clients to forward video data at the appropriate rate [10].

REFERENCES

[1] Ammar, M. Why Johnny can't multicast: lessons about the evolution of the Internet. Keynote Address, In *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (NOSDAV 2003), Monterey, CA, p. 1, June 2003.

[2] Annapureddy, S., S, Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez. Exploring VoD in P2P swarming systems. In *Proceedings of the 26th IEEE International Conference on Computer Communications* (INFOCOM 2007), Anchorage, AK, pp. 2571–2575, May 2007.

[3] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, SplitStream: high-bandwidth multicast in cooperative environments. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles* (SOSP 2003), New York, NY, pp. 298–313, Oct. 2003.

[4] S. W. Carter. and D. D. E. Long. Improving video-on-demand server efficiency through stream tapping. In *Proceedings of the 5th International Conference on Computer Communications and Networks* (ICCCN '97), Las Vegas, NV, pp. 200–207, Sep. 1997.

[5] Cohen, B. Incentives Build Robustness in Bit Torrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, 2003.

[6] Dan, A., P. Shahabuddin, D. Sitaram and D. Towsley. Channel allocation under batching and VCR control in video-on-demand systems. *Journal of Parallel and Distributed Computing*, 30(2):168–179, Nov. 1994.

[7] Kirk, P. Gnutella—A protocol for a revolution. http://rfc-gnutella.sourceforge.net/

[8] Lin, F. C. Zheng, X. Wang, X. Xue. ACVoD: A peer-to-peer based video-on-demand scheme in broadband residential access networks, *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4)4, 2007

[9] Liu, Y., Y. Guo and C. Liang. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28, Mar. 2008.

[10] Mol, J .J. D., J. A. Pouwelse, M. Meulpolder, D. H .J. Epema, and H. J. Sips. Give-to-get: An algorithm for P2P video-on-demand. In *Proceedings of the 13th Annual Multimedia Computing and Networking Conference* (MMCN 2008), San Jose, CA, Jan. 2008.

[11] Pâris, J.-F. A cooperative distribution protocol for video-on-demand. In *Proceedings of the 6th Mexican International Conference on Computer Science*, Puebla, Mexico, pp. 240–246., Sep. 2005.

[12] Shah, P. and J.-F. Pâris. Peer-to-peer multimedia streaming using BitTorrent, In *Proceedings of the 26th*

International Performance of Computers and Communication Conference (IPCCC 2005), New Orleans, LA, pp 340–347, Apr. 2007.

[13] Sheu, S., K. A. Hua, and W. Tavanapong. Chaining: a generalized batching technique for video-on-demand systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems* (ICMS '97), pp. 110–117, June 1997.

[14] Su, T.-C., S.-Y. Huang, C.-L. Chan and J.-S. Wang. Optimal chaining and implementation for large scale multimedia streaming, In *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo* (ICME 2002), Lausanne, Switzerland, Vol.1, pp. 385–388, Aug. 2002.

[15] Su, T.-C., S.-Y. Huang, C.-L. Chan and J.-S. Wang. Optimal chaining scheme for video-on-demand applications on collaborative networks. *IEEE Trans. on Multimedia*, Vol. 7, No 5, pp. 972–980, Oct. 2005.

[16] Viswanathan, S. and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *Multimedia Systems*, 4(4):197–208, Aug. 1996.

[17] Zhang, X., J. Liu, B. Li, and T.-S. P. Yum, CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of the24th IEEE International Conference on Computer Communications (INFOCOM '05)*, Vol. 3, pp. 2102–2111, March 2005.