Delayed Chaining: A Practical P2P Solution for Video-on-Demand

Jehan-François Pâris
Department of Computer Science
University of Houston
Houston, TX , USA
paris@cs.uh.edu

Ahmed Amer
Department of Computer Engineering
Santa Clara University
Santa Clara, CA, USA
a.amer@acm.org

Abstract—**Most home computers have upload bandwidths that do not allow them to stream good quality video. As a result, they cannot fully participate in peer-to-peer video streaming solutions. We propose to address that issue by letting these clients use the storage space they have on one or more remote servers. Each time a client plays a streaming video, it will upload a copy of that video to one of these servers. Once uploaded, that copy can then service another client wanting to play the same video. Our simulations indicate that our delayed chaining scheme can dramatically reduce video server workloads at all stable request arrival rates. Daily usage spikes can be easily accommodated as long as cloud copies remain at least 24 hours online. Strategies for handling the launching of a new video include introducing a price differential, using predictive prefetching and requiring advance purchases.**

Keywords- video-on-demand; P2P systems; chaining

I. Introduction

Video on Demand (VoD) constituted 27 percent of all internet traffic in 2009 [13]. This traffic is presently being supported by a very expensive infrastructure comprising huge server farms, high-bandwidth Internet connections and extensive content delivery networks.

Peer-to peer (P2P) technology avoids this predicament by allowing clients to share the burden of distributing video data. Most of the work now done by the server infrastructure becomes delegated to clients that will require little additional power to forward the video data they are receiving to their "peers." As a result, P2P solutions can potentially eliminate the need for large power-hungry server farms. Since each client is a potential co-worker, they can handle very large and sudden surges of demand, such as those caused by flash crowds. In addition, P2P solutions do not require any special support from the network, be it IP multicast or any specific content distribution infrastructure.

Adapting P2P technology to VoD is not a trivial task because VoD presents some important differences from other P2P applications, such as file sharing. First, extant file sharing systems do not account for the real-time needs of streaming applications because they do not download video data in sequence. As a result, such data remains unusable until the download is complete [20]. Second, these real-time needs mean that most data transfers among peers will be unidirectional as peers that are already watching the video will forward their video data to more recently arrived peers without receiving any video data from them. Conversely, there will be fewer bilateral data exchanges among peers. In addition, most home computers have very limited upload bandwidths, which do not allow them to forward high-quality video in real time. For instance, the authors of CoolStreaming noted that most of its peers could not reliably forward data at rates as low as 1/6 of their video consumption rate [6]. This situation is likely to persist because most consumers are more interested in their download rates than their upload rates.

We propose a new streaming solution that allows computers with very limited upload bandwidth to fully participate in the video distribution process as long as they have access to enough cloud storage space to store and later forward entire videos. Our solution applies to services distributing previously recorded programs, such as a VoD service distributing movies or older episodes of a television series. We recognize that many computer users have access to gigabytes of free or almost free storage space on one or more cloud servers. In our scheme, each peer that plays a streaming video will upload a copy of that video to one of these cloud servers. Later arriving peers wanting to play the video will then be directed to that copy.

To reduce the impact of these transmissions on the cloud server workload, we do not let cloud copies persist for more than 24 to 48 hours and only allow each cloud copy to serve a single client request before being deleted.

Our approach is orthogonal to the efforts of the DECADE working group to define a framework that would allow P2P applications to interact with various kinds of in-network storage in a direct fashion. [14]. All peer uploads would be eliminated as the data exchanged by the peers move among their in-network storage. Our solution requires no changes to the current Internet infrastructure and does not force peers to share write access to their cloud storage with other peers.

Our simulations indicate that our delayed chaining scheme can dramatically reduce the videos server workload at all stable request arrival rates. In addition, daily usage spikes can be easily accommodated as long as cloud copies remain online 24 hours before being deleted.

The remainder of the paper is organized as follows. Section 2 reviews relevant work on P2P solutions for video-on-demand. Section 3 introduces our delayed chaining scheme and Section 4 analyzes its performance. Section 5 addresses

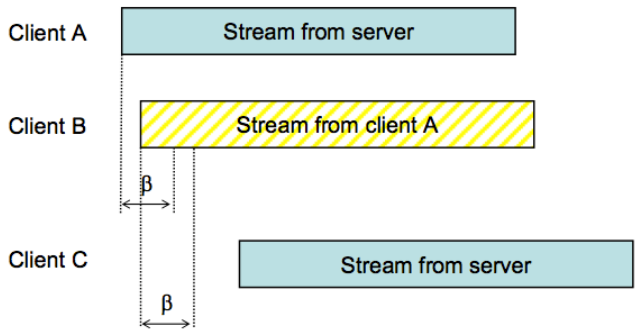

Fig. 1. How chaining works.

implementation issues. Finally, Section 6 presents our conclusions.

II. Previous Work

For brevity, we will focus our discussion on chaining protocols for video-on-demand.

Standard chaining [11] 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 try to get most of their data from their immediate predecessor. As a result, video data are "pipelined" through the clients belonging to the same chain. Because chaining does not require clients to have very large data buffers, a new chain`has to be restarted whenever the time interval between two successive clients exceeds the capacity β of the buffer of the previous client. Fig. 1 shows three sample client requests. Since client A is the first customer, it will get all its data from the server. As client B arrives less than β minutes after client A, it can receive all its data from customer A. Finally client C arrives more than β minutes after client B and must be serviced directly by the server.

The main weakness of standard chaining is its poor performance at low arrival rates, more precisely, whenever the time interval between two consecutive requests exceeds β minutes. Several variants of the chaining protocol address that issue. *Advanced chaining* [7] proposes to bridge this gap by inserting every β minutes idle peers that will relay the data. *Optimal chaining* [15] addresses the same issue by letting clients "borrow" the buffers of other clients in order to bridge gaps between requests. *Expanded chaining*, also known as the *cooperative* video distribution protocol [8, 9], assumes that today's clients now have large enough buffers to store the videos, and does not require them to keep forwarding data to their successors in the chain after they have finished playing the video.

As in all other chaining schemes, client failures will result in a shortening of the chain with the predecessor X of a failed client Y taking over the duties of Y and forwarding its video data to the successor Z of Y.

III. Delayed Chaining

To remain scalable, any P2P data distribution scheme must ensure that members forward as much data to the other peers as they receive from them. While a group of peers can tolerate

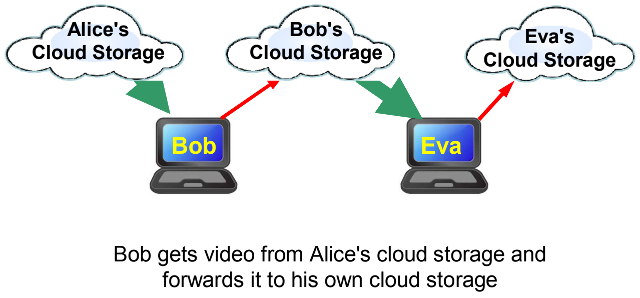
Alice's Cloud Storage
Bob's Cloud Storage
Eva's Cloud Storage
Bob
Eva
Bob gets video from Alice's cloud storage and forwards it to his own cloud storage

Fig. 2. How video data get forwarded from client to client.

selfish peers, which forward as little data as they can, it would not survive without the presence of "unselfish" peers whose contributions to the group exceed what is expected from them.

This condition is more difficult to meet in video streaming applications because all video data must be transmitted and received in a timely manner. In addition, most home computers have very limited upload bandwidths, which do not allow them to forward high-quality video in real time. For instance, the various ADSL standards allocate between 75 and 95 percent of their bandwidth to download traffic leaving only between 5 and 25 percent of it available for upload traffic [18].

We propose a new streaming solution that addresses that issue and allows computers with lower upload bandwidth to fully participate in the video distribution process. Our delayed chaining scheme applies to services that distribute previously recorded programs, such as a VoD service distributing movies or older episodes of a television series. Live transmissions of events are specifically excluded. In addition, we assume that the videos offered to the clients of the service will remain on line for several days, if not several weeks. This is similar to common usage for DVRS and is the model employed by the hugely popular BBC iPlayer in the UK [2].

We recognize that many computer users have access to gigabytes of free or almost free storage space on one or more cloud servers. To provide a decent quality of service to their users, these cloud servers are likely to have better upload and download bandwidths than most home computers.

In our scheme, each client that plays a streaming video will upload a copy of that video to one of these cloud servers. Later arriving clients wanting to play the video will then be directed to that copy. As Fig. 2 shows, client Bob gets its video data from the copy of the video that client Alice had previously uploaded to her cloud storage. While he plays the video, Bob simultaneously uploads it to his cloud storage. Client Eva will later get her video data from that copy.

Our scheme thus forms chains of requests where clients get their video data from a cloud copy created by a previous client and create a new copy of the video in their cloud storage for the benefit of a future client. What distinguishes our scheme from other chaining schemes is the presence of significant delays between the time a client X starts watching a video and the time a second client Y can access the cloud copy of the video created by client X.

Let D denote the duration of the video, b_v the video consumption bandwidth and b_u the client upload bandwidth.

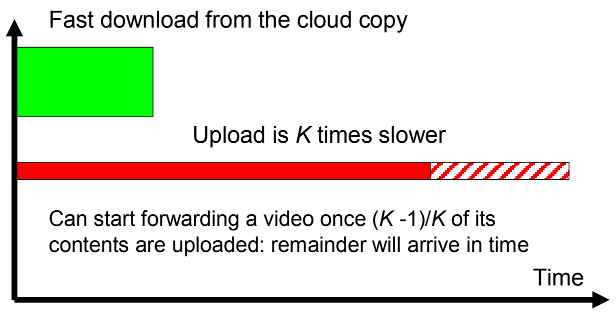
Fast download from the cloud copy
Upload is K times slower
Can start forwarding a video once (K -1)/K of its contents are uploaded: remainder will arrive in time
Time

Fig. 3. The upload delay.

We assume that $b_u < b_v$ because otherwise each client could forward the video directly to the next client without any intermediary. Thus $K = b_v/b_u$ is greater than one.

To fully upload the video to the cloud will require KD time units. This delay can be reduced by observing that another client can start streaming the video as soon as a fraction $(K – 1)/K$ of its contents is stored in the cloud. As seen in Fig. 3, we can thus reduce the video upload delay to $(K – 1)D$. Consider a video consumption rate of 1.5 Mb/s, which is the lowest video quality offered by NetFlix and claims to offer "stereo audio and video quality comparable to DVD" [19], and a client upload bandwidth of 500 kb/s, which is the maximum upload bandwidth offered by ITU G.992.2—better known as ADSL Lite or G.Lite. The video upload delay will then be equal to twice the duration of the uploaded video, that is, four hours for a two-hour video. This will cause problems during rapid increases of the video request arrival rate as there will be no cloud copies available to accommodate the arriving requests. We will address this issue in section 5.

A last issue to consider is the potential impact of our proposal on cloud servers. Any excessive burden imposed on these servers could be viewed by their owner as a violation of the client's terms of service. This would likely result in a cease and desist notice, if not a loss of the service. For this reason, we need to limit this impact to a minimum. We propose to achieve this goal by not allowing each cloud copy to serve more than one client request and limit the lifetime of these copies to one or two days. As we will see, cloud copies of the videos need only persist for two to four hours to realize the benefits of our scheme, and that these benefits are realized in spite of any delays in uploading the videos caused by low upload bandwidth. In addition, service providers could mitigate the storage costs of our solution by eliminating redundant copies of individual videos through deduplication [5].

While we focus on the case of clients uploading their videos to their cloud servers, and suffering the slow upload speeds that would typically entail, our scheme is poised to take advantage of mechanisms that would allow client to initiate server-to-server, such as from the VoD server to the cloud server and between two cloud servers.

IV. Performance Evaluation

We present in this section two models that estimate the server bandwidth requirements of our delayed chaining scheme. Our first model is a simplified Markov model that

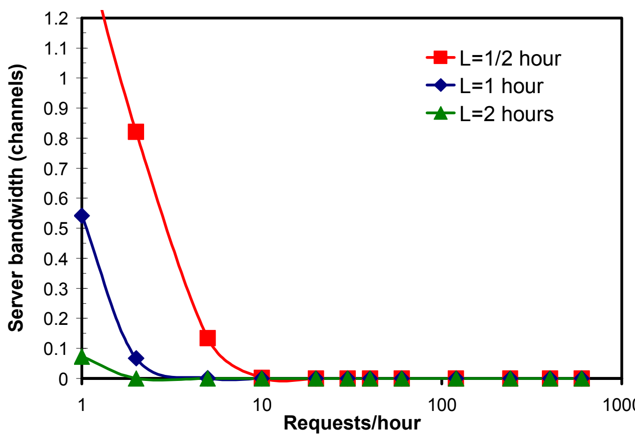

Fig. 4. Server bandwidth requirements of the delayed chaining protocol assuming that cloud copies can serve an arbitrary number of requests.

assumes each cloud copy of a video can serve, concurrently or not, an unlimited number of client requests during its lifetime. Our second model is a more realistic simulation model that assumes that each cloud copy of a video cannot serve more than one client.

In both cases, we will focus on the behavior of customers watching entire videos from beginning to end. We reserve for a further study the case of customers interrupting their viewing before the end of the video, pausing while watching the video, and/or moving forwards and backwards. We will postulate the existence of a reward mechanism that motivates all users to upload to the cloud a copy of the video they are watching.

We will assume that customer requests arrive continuously and independently of each other with a constant rate λ. They are therefore modeled with a Poisson process. The time between arrivals is then governed by the exponential distribution. The probability density of this distribution is $p(t) = \lambda\, e^{-\lambda t}$.

In both models, D will denote the video duration, L the lifetime of the cloud copy of the video and t the time elapsed between two consecutive requests. Bandwidths will always be expressed in multiples of the video consumption rate. In addition, our models assume that both downlink and uplink bandwidths are utilized solely for the video workload, as it would be very difficult to estimate the impact of other workloads.

A. A simplified Markov model

In our first model, we will assume that a client will be able to obtain all its video data from the cloud copy of a previous client as long as *at least one* cloud copy of the video remains available when it sends its request to the server. Conversely, the client will get all its video data from the server if no such copy is available.

Assuming that all clients experience the same upload delay T_u, we see that the probability of not having a cloud copy available is the same as the probability of no request arrival during a past time interval whose duration is equal to the cloud copy lifetime L. Hence the probability that a given request will be serviced by the server is $e^{-\lambda L}$.

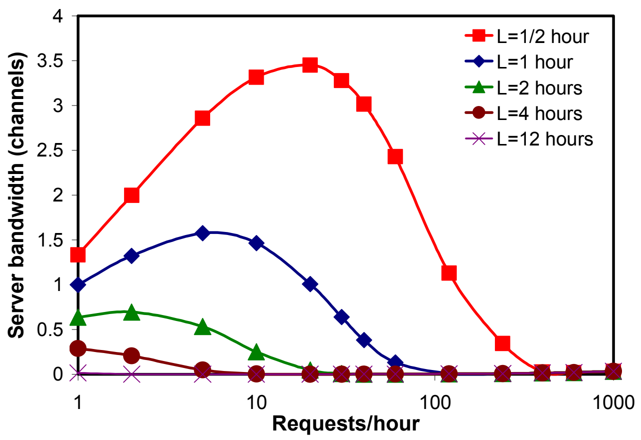

Fig. 5. Server bandwidth requirements of the delayed chaining protocol assuming that cloud copies can only serve one request.

The average server workload per video will thus be $w = e^{-\lambda L} D$ and the average server bandwidth B will be

$$B = \lambda w = \lambda e^{-\lambda L} D. \quad (1)$$

Fig. 4 displays this estimated bandwidth for a two-hour video when customer arrival rates vary between one and one thousand requests per hour. We did not consider higher arrival rates as they seemed unrealistic.

As we can see, the server bandwidth requirements of our delayed chaining scheme remain almost negligible as long as either the client arrival rate exceeds ten requests per hour or the lifetime of cloud copies exceeds one hour, which is well below our proposed lifetimes.

B. *A more realistic simulation model*

Our Markov model assumes that each cloud copy of a given video can service an arbitrary number of client requests. As we mentioned in a previous section, two main reasons make this assumption unrealistic. First, it is not clear that all servers that hold cloud copies will always be able—or willing—to forward its video data to more than one client at a time at the right rate. Second, we already mentioned that we did not want to impose a too heavy burden on the server as it would risk to be perceived as a violation of the client's terms of service. For these two reasons, we will only allow each cloud copy to serve a single client request before being deleted.

To obtain a more realistic evaluation of the performance of our scheme, 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.

We assumed that the K ratio between the video consumption rate b_v and the client upload bandwidth b_u was equal to four, which means that the video upload delay would be equal to six hours for a two-hour video. This value is rather conservative but takes into account the gap between the nominal upload bandwidths advertised by Internet access providers and actual upload bandwidths.

Since no data are shared among clients 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.

Fig. 5 summarizes our results. As we can see, not letting each cloud copy serve more than one client does not result in a significant increase of the server bandwidth as long as the client arrival rate exceeds five requests per hour and the lifetime of cloud copies exceeds two to four hours. Even at lower arrival rates, our scheme performs much better than a naïve scheme allocating a separate video channel to each client.

V. Implementation Issues

In this section we address two issues concerning the implementation of our delayed chaining scheme, namely its incentive mechanism and the techniques we may use for improving its behavior during periods of rapid changes of the client arrival rate.

A. *Incentive mechanism*

Tit-for-tat incentive mechanisms have been popularized by the BitTorrent protocol [3]. While Piatek et al. have shown these incentives can be subverted [10], they are still being used by many P2P video distribution protocols based on BitTorrent [16]. Because these mechanisms only control how data are exchanged within pairs of cooperating peers, they are totally decentralized and do not require any centralized entity, which makes them eminently scalable.

Similar mechanisms cannot apply to our delayed chaining scheme as clients will never forward a video to another client before they have finished playing it. We need instead an incentive mechanism letting clients collect rewards that they can claim the next time they order a video. This could be done through a virtual currency mechanism.

A currency-based mechanism would let clients "purchase" video data from their predecessors with the currency they collect "selling" their own video data to their successors. Its sole drawback is that it requires the server to act as a virtual banker. Such management overheads become less of a concern when we are dealing with lengthy videos as the bandwidth savings would greatly outweigh any disadvantages introduced by the incentive scheme.

Among the various virtual currency models we could implement, we should mention Micro-Payment [4], Dandelion [12], PACE [1] and the lightweight currency incentive mechanism proposed by Wang et al. [17].

B. *Transient behavior*

As we have seen, our delayed chaining scheme introduces a significant delay between the time a client X starts watching a video and the time a second client Y can access the cloud copy of the video created by client X. Depending on the duration of the video D and the ratio b_v / b_u between the video consumption rate b_v and the client upload bandwidth b_u, this delay can reach four to six hours. This will result in an unacceptably high server workload during times of rapid increases of the popularity of a video. Let us show how we can address this issue. We will have to consider two different cases, namely daily usage spikes at different times of the day and the launching of new videos.

Handling daily usage spikes is a fairly easy task as it only requires selecting a minimum cloud copy lifetime of 24 to 36 hours. Handling the launching of new videos is a more challenging task. We propose three solutions that are not mutually exclusive.

- *Introducing a price differential*: We could charge a higher price to clients wanting to be the first to watch a new video. This would create a two-tier pricing such that clients wanting to watch a video the day of its release would be charged more than clients willing to wait one day or two. This solution would not be very different from the policies of many DVD rental stores that charge more for recently released DVDs than for the DVDs they have in stock. We could even introduce a dynamic pricing policy where the price charged for watching a given video would depend on the presence or the absence of cloud copies of that video. In both cases, the price differential should take into account the cost of renting extra servers on a content delivery network.
- *Using predictive prefetching*: We could use a predictive scheme similar to the one used by NetFlix and use it for preloading on clients the soon-to-be-released videos they are the most likely to request. This process should take place one or two days ahead of time in order to let these clients create a cloud copy that will be ready by the time the video is officially released. Unlike Netflix, where the prediction accuracy is important, our task would not be to preload the video for every client that will watch it, but to seed the video to enough clients to serve the needs of all peers. This option would be the most user-friendly but also the hardest to implement.
- *Requiring advance purchases*: We could require people who want to be the first to watch to preorder their video one or two days in advance, which would give time to seed enough cloud copies of the video. This is not very different of what customers often have to do to be the first to get a popular DVD or a popular video game. This third option seems to offer the best compromise between user friendliness and ease of implementation.

VI. Conclusion

We have proposed a delayed chaining video-on-demand distribution protocol that allows clients with limited upload bandwidth to forward the video at a much higher rate to other clients by using the storage space these clients have on one or more remote servers. Each time a client plays a streaming video, it will upload a copy of that video to one of these servers. Later arriving client wanting to play the video will then be directed to that copy. To avoid server overuse, our scheme assumes that these cloud copies will serve no more than one client request and will be deleted after 24 to 48 hours.

Our simulations indicate that our delayed chaining scheme can dramatically reduce the videos server workload at all stable request arrival rates. In addition, daily usage spikes can be easily accommodated as long as cloud copy lifetimes remain above 24 hours. Of the three strategies for handling the launching of a new video that we discussed, requiring advance purchases seems the most natural choice.

More work is still needed to select and implement an incentive scheme and build a prototype that would validate the feasibility of our approach. In particular, we could explore the use of *helpers*, that is, unreliable "servelets" that contain some of the video data and participate in their distribution [21].

References

[1] C. Aperjis, C. A. Freedman, and R, Johari. "Peer-assisted content distribution with prices," *Proc. ACM CoNext '08 Conf.*, pp. 17:1–17:12, Dec. 2008.

[2] http://www.bbc.co.uk/iplayer/ retrieved on March 12, 2012

[3] B. Cohen. "Incentives Build Robustness in Bit Torrent," *Proc. Workshop on Economics of Peer-to-Peer Systems*, May 2003.

[4] P. Golle, K. Leyton-Brown, and I. Mironov. "Incentives for sharing in peer-to-peer networks," *Proc. ACM EC '01 Conf.*, pp. 264–267, Oct. 2001

[5] S. Jones, *Online De-duplication in a Log-Structured File System for Primary Storage*, Technical Report UCSC-SSRC-11-03, May 2011.

[6] B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu and X. Zhang, "Inside the new CoolStreaming: Principles, measurements and performance implications," *Proc. 27th INFOCOM Conf.*, pp. 1031–1039, Apr. 2008.

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

[8] J.-F. Pâris. "A cooperative distribution protocol for video-on-demand." *Proc. 6th Mexican Int. Conf. on Computer Science,* pp. 240–246, Sep. 2005.

[9] J.-F. Pâris and T. J. Schwarz. "An Analysis of Chaining Protocols for Video-on-Demand," *Proc. 9th I2TS Symp.*, Dec. 2010.

[10] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, A. Venkataramani. "Do incentives build robustness in BitTorrent?" *Proc. 4th USENIX NSDI Symp.*, Apr. 2007.

[11] S. Sheu, K. A. Hua, and W. Tavanapong. "Chaining: a generalized batching technique for video-on-demand systems," *Proc. IEEE ICMS* '97 Conf., pp. 110–117, June 1997.

[12] D. Sirivianos, J.H. Park , X. Yang and S. Jarecki. "*Dandelion: Cooperative Content Distribution with Robust Incentives*," *Proc 2007 USENIX Conf.*, pp. 157–170, June 2007.

[13] T. Spangler October 2009. *Video-On-Demand Now 27% Of Internet Traffic.* http://www.multichannel.com/article/366266-Video_On_Demand_Now_27_Of_Internet_Traffic_Study.php, retrieved on August 12, 2011.

[14] H. Song, N. Zong, Y. Yang and R. Alimi. "DECoupled Application Data Enroute (DECADE): Problem statement," Internet Draft. Oct. 14, 2009.

[15] T.-C. Su, S.-Y. Huang, C.-L. Chan and J.-S. Wang. "Optimal chaining and implementation for large scale multimedia streaming," *Proc. 2002 ICME 2002 Conf.*, Vol.1, pp. 385–388, Aug. 2002.

[16] A. Vlavianos, M. Iliofotou, and M. Faloutsos. "BiToS: Enhancing BitTorrent for supporting streaming applications," *Proc. 9th IEEE Global Internet Symp.*, pp. 1–6, Apr. 2006.

[17] C. Wang, H. Wang, Y. Lin and S. Chen, "A lightweight currency-based P2P VoD incentive mechanism," *Proc. 3rd Int. Joint CSO Conf.*, Vol. 1, pp. 272–276, 2010.

[18] Wikipedia, *Asymmetric Digital Subscriber Line*, http://en.wikipedia.org/wiki/Asymmetric_Digital_Subscriber_Line, retrieved on August 12, 2011.

[19] Wikipedia, *NetFlix*, http://en.wikipedia.org/wiki/Netflix, retrieved on August 12, 2011.

[20] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava, "On peer-to-peer media streaming," *Proc. 22nd ICDCS Conf.*, pp. 363–371, July 2002.

[21] H. Zhang and K. Ramchandran, "A reliable decentralized peer-to-peer video-on-demand system using helpers," *Proc. 27th PCS Symp.*, pp. 509–512, May 2009.