Analytical Model for BitTorrent-based Live Video Streaming

Saurabh Tewari, Leonard Kleinrock
Computer Science Department
University of California at Los Angeles
Los Angeles, U.S.A

Abstract— **Peer-to-peer live video streaming over the Internet has been measured to support over 100,000 concurrent users. While the approach is very attractive, established providers need to understand the performance of such a system before deploying such a system as a frequent loss in quality would jeopardize their reputation. This paper provides an analytical model to inform the design of BitTorrent-based live video streaming solutions. While, given the current broadband deployment scenario, a pure peer-to-peer solution can support only limited streaming rates, our analysis shows that the addition of a well-designed peer-to-peer solution to existing server-based streaming infrastructures can allow substantially higher streaming rates. The efficiency of a BitTorrent-like peer-to-peer solution depends on the peer group size and the number of fragments available for sharing at any given time. Our analysis suggests that the efficiency of the peer-to-peer solution is not sensitive to the size of the peer group for groups larger than 15-20 users. A similar threshold exists for the number of fragments available for sharing at any given time. For live streaming scenarios, this threshold dictates that the fragment size be substantially smaller than the default fragment size in BitTorrent to ensure that the stream latency is small.**

Keywords-Live Streaming, Peer-to-Peer, BitTorrent, Video, Efficiency, Latency

I. INTRODUCTION

In early peer-to-peer file-sharing systems, files could be exchanged only when peers had the files in their entirety. However, since most users leave the system when the file download completes, a very small fraction of the total peer upload capacity was ever utilized. By fragmenting files into small pieces (and by employing a Tit-for-Tat policy in choosing the peers to which a peer would upload fragments to), BitTorrent [1] allowed utilization of a large fraction of the total peer upload capacity. The idea of exchanging fragments also enabled supporting streaming applications over a peer-to-peer network and several peer-to-peer video streaming applications e.g. CoolStreaming [3] and PPLive [5] have come up in recent years. Measurements on these systems report over 100,000 concurrent users on a single channel for PPLive [6] and over 25,000 concurrent users on a single channel for CoolStreaming [4]. These numbers have generated interest in using this approach for commercial live video streaming. While the scalability of peer-to-peer solutions [8] is very attractive, established providers need to understand the performance of a system utilizing peer-to-peer technology before they deploy it in their systems as users have come to expect a certain level of quality from them and any large deviations from customer expectations would be detrimental to their reputation. In this paper, we provide an analytical model to inform the design of a BitTorrent-based live video streaming solution. Given the asymmetric nature of the broadband access links currently in deployment, it is obvious that a pure peer-to-peer solution can support only limited streaming rates. Our work in this paper allows service providers to estimate the server-side capacity needed to support a targeted number of users at a given streaming rate.

The key objective in using peer-to-peer technology for streaming (and other content distribution applications) is to utilize the end-user upload capacities to support the streaming application. We refer to the fraction of total end-user upload capacity that can be utilized as the *efficiency of fragment exchange* in BitTorrent-like systems in this paper. Our analytical work in this paper will allow system designers to select appropriate parameters of a BitTorrent-based live streaming solution with the knowledge of the fraction of peer upload capacities their design will be able to utilize and the playback delay the fragment exchange process will introduce.

In Section 2, we review the related work in the area. Section 3 illustrates the enormous reductions in the server-side capacity a well-designed peer-to-peer solution can offer. In Section 4, we discuss the effect of peer group size on the efficiency of fragment exchange. Section 5 discusses the importance of the number of fragments available for sharing in determining the efficiency of fragment exchange. We discuss the implications of our analysis in Sections 4 and 5 for design of a BitTorrent-based peer-to-peer live video streaming application in Section 6. Our analysis in Section 6 establishes the tradeoff between the efficiency of fragment exchange and the playback delay due to allowing for fragment exchange and illustrates the critical role of the fragment size in a peer-to-peer live streaming service. Section 7 concludes the paper.

II. RELATED WORK

Our work in this paper builds upon the BitTorrent peer-to-peer protocol [1]. File distribution over BitTorrent comprises of a tracker hosted on a website that maintains a list of the peers currently downloading the file (along with information on the amount of file data each peer has downloaded and uploaded and the amount of the file data they still need to obtain). Each

peer wishing to download the file obtains the *torrent* for the file which includes the location of the tracker and a list of all the pieces (fragments) into which the file is broken into and hash values for each file fragment (to enable verification of the integrity of the downloaded fragment). The peer then contacts the tracker which provides it a list of about 50 randomly selected peers (whenever that many peers are available) and each peer tries to maintain links to 20-30 other peers and keep information of the fragments each of these 20-30 peers has. Based on this information and the fragments a peer needs, a peer requests fragments to download from these peers. To maintain diversity of the fragments available for download in the network, peers request the fragments which fewest number of peers have (this is referred to as the *rarest-first* download policy in BitTorrent). A peer that has multiple download requests pending will upload fragments to the peers that are uploading data to it at the fastest rate. This mechanism is referred to as the *Tit-for-Tat* upload policy and is the key incentive mechanism in BitTorrent [2] considered responsible for it success. Peers try to upload fragments to 4-5 other peers at any given time (to keep their upload capacities utilized). To find new (faster) fragment sources, peers *optimistically unchoke* [2] a random peer every 30 seconds or so. Breaking a file into small fragments allowed BitTorrent to enable peers to upload fragments to others (Tit-for-Tat enforced such sharing) while they were still downloading a file. We refer to this mechanism of peers uploading fragments of the file they have to other peers that need them as the *fragment exchange* process in BitTorrent-like systems. The ability of a peer to contribute to the content distribution process is determined by whether there are other peers that want a fragment they have. Larger the peer group size and larger number of fragments of the file, more likely it is that there is at least one peer that wants a fragment that a peer has. While the number of fragments is on the order of a few thousand in typical file-sharing scenarios (and, hence, the probability that there would be at least one peer that wants a fragment that a peer has is nearly 1), in live streaming, only a few fragments are available for sharing (see discussion in Section 4 and 5). However, if the peer group size is large, there may yet be possible that there would be peers that want a fragment that a peer has. The effect of the peer group size and the number of fragments available for sharing at any given time on the probability that there would be at least one peer that wants a fragment that a peer has (which we refer to as the efficiency of file exchange and is equal to the fraction of the total end-user upload capacity that be utilized) is a key contribution of our work. Such a relation was first provided by [10] (to best of our knowledge) but they only developed the case where the number of fragments available for sharing was very large and, hence, as presented their results have limited utility for a live streaming scenario. We expand on the analysis they provide and provide important guidelines for the design of a BitTorrent-based live video streaming application. We note that when we refer to *peer group size*, we are referring to the number of peers a peer would have links to, and do not imply that these are isolated groups of peers – connecting to randomly selected peers would ensure a connected network (the connectivity considerations are same as that for an Erdos-Renyi random graph [19]).

There has been considerable work in the area of peer-to-peer live video streaming. We refer to only a small subset of the application layer work here omitting interesting approaches involving media encoding (e.g. [18]) and various network layer techniques (e.g. [17]). Overlay multicasting work (e.g. End-System Multicast [12]) can be considered a precursor to "pure" peer-to-peer streaming. The tree-based structures typically used in the initial systems are constrained by the limited end-user upload bandwidth which limits the tree fan-out leading to large tree depths. This leads to high playback delays for leaf users and renders these systems susceptible to nodes leaving the system. These initial systems were followed by mesh-based architectures (e.g. SplitStream [13], CoopNet [14], PRIME [15] etc.) and the BitTorrent approach we elaborate upon in this paper can be seen as taking this approach to the extreme in allowing a different tree for each fragment. The BiToS [7] and the BASS [11] works explore using BitTorrent for streaming and, thus, are similar in spirit to our work in this paper. While [7] provides insights into designing fragment request policies, they only mention the importance of appropriate buffer size selection and do not explore it. Reference [11] proposes a hybrid server and peer-to-peer based streaming architecture as expounded in this paper but only offers simulation results on the savings in the server-side capacity that can be achieved with a BitTorrent-like peer-to-peer approach to supplement the server-based streaming. Our work in this paper adds analytical basis to the work in [11] and complements the work in [7]. Reference [16] suggests modifications to peer selection and fragment request policies to support streaming applications and, thus, is complementary to [7] and our work. CoolStreaming [3] is a frequently cited work in peer-to-peer live streaming (other widely deployed publicly available peer-to-peer live streaming systems such as PPLive [5] are proprietary). CoolStreaming's core operations (periodically exchanging data availability information and downloading unavailable data from and uploading available data to appropriate peers) are same as in BitTorrent. However, [3] does not discuss the trade-offs involved in selecting the buffer size or the target number of peers to maintain connections. Thus, our work in this paper complements the ideas in [3] and provides an analytical basis for selecting the fragment size, the peer group size and the buffer size in a CoolStreaming-like system.

III. PEER-TO-PEER: BENEFITS AND LIMITATIONS

The target audience of large-scale live video streaming has asymmetric broadband access. For example, while the average residential broadband download rates in the United States are around 2.8 Mbps, the corresponding upload rates average to only about 500 kbps (see [9] for current statistics). Clearly, it is not possible to support any live streaming at rates greater than 500 kbps with a pure peer-to-peer live streaming solution even though the download rates allow offering MPEG-1 or better quality streams. Thus, major commercial vendors are unlikely to find pure peer-to-peer live streaming to be adequate in face of competitive pressures. However, effectively leveraging the peer upload bandwidths to assist server-based stream delivery can still offer tremendous competitive advantage. In Figure 1, we show the server-side capacity requirement as the number of users increases for different streaming rates and for different

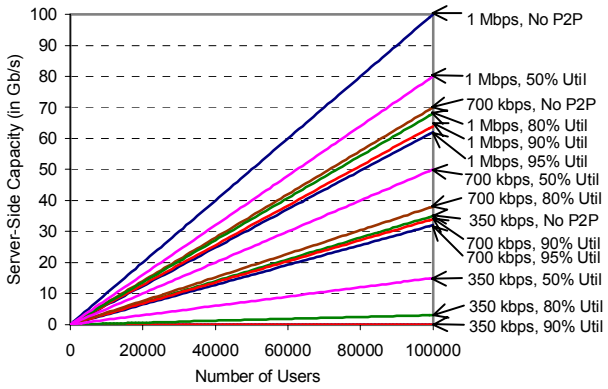

Figure 1. Benefits of using a well-designed peer-to-peer solution (Assumption: Peer Upload Capacity is 400 Kbps)

values of the effectiveness of the peer-to-peer solution in utilizing peer upload bandwidths[1].

The first observation we make from Figure 1 is that the server-side capacity required to support 350 kbps streaming rate for 100,000 users when no peer-to-peer solution is used is sufficient to support a streaming rate of 700 kbps if the server-side infrastructure is augmented with BitTorrent-like peer-to-peer fragment exchanges among the peers viewing the same stream which can utilize 90% of the total peer upload capacity.

The second observation we make is the importance of a well-designed peer-to-peer solution – the difference in the server-side capacity needed to support 100,000 users at 700 kbps streaming rate is 12 Gbps between a solution that can utilize 50% of the total peer upload capacity versus one that can utilize 80% of the total peer upload capacity.

In the remaining paper, we explore the key factors that determine the peer upload capacity utilization in BitTorrent-like peer-to-peer fragment exchange scenarios to inform the design of a peer-to-peer live video streaming system.

IV. Peer Group Size

As illustrated in the previous section, the fraction of the total peer upload capacity that can be utilized is the key metric for capacity planning purposes. The fraction of the total peer upload capacity that can be utilized to support the peer-to-peer live streaming application is the probability that, in a group of k peers, there exists at least one peer that needs one of the fragments a peer has[2]. Assuming that the fragments each peer has are chosen at random from the set of all the fragments that

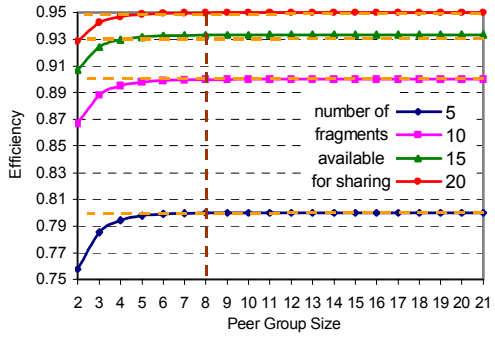

Figure 2. Effect of Peer Group Size on the Efficiency of Fragment Exchange

are currently relevant[3] and the fragments each peer has are independently chosen, this probability, the efficiency for BitTorrent-like fragment exchange scenarios, is computed in [10] to be[4]:

$$\eta = 1 - \sum_{n_i=0}^{N-1} \frac{1}{N} \left(\frac{N - n_i}{N(n_i + 1)} \right)^k \qquad (1)$$

where N is the number of fragments available for sharing and k is the number of peers in the group of cooperating peers in the fragment exchange. We note that, as [10] stated, their approximation for this expression of η is valid only when N is large as is the case in typical BitTorrent file-sharing scenarios (which have over a thousand fragments per file). However, in a live streaming scenario, future fragments are not available and, if the offering to remain competitive to a purely server-based streaming solution, the "live" stream must be played out within 10-20 seconds including any transmission delays. Thus, the aggregate size of the fragments available for sharing is limited to few seconds worth of playback buffer. In Figure 2, we show the effect of k, the peer group size, on the efficiency of fragment exchange process for different values of N, the number of fragments available for sharing (with η calculated using Eq. 1).

From Figure 2, we see that the peer group size has no effect on the efficiency once the peer group size is more than 7-8. Given that Eq. 1 is derived under the assumption of

[1] If the peer-to-peer solution can, on average, utilize η fraction the peer upload capacity and the peer upload capacity is c, the server side capacity needed to support a streaming rate of x to k users is equal to $\max(0, k(x-\eta c))$.

[2] To ensure the highest possible utilization of peer upload capacities, a peer should apply tit-for-tat-like upload policies only to select which peer to upload to when there are more than one peers requesting content from it. In other words, peers (e.g., say, peer A) should always be uploading fragments (i.e. peer upload capacity is being utilized) as long as there exists another peer, say, peer B that wants a fragment that peer A has.

[3] For BitTorrent file-sharing, this includes all the fragments of the file being downloaded. For live streaming, it includes only the fragments available in the few seconds of the playback buffer each peer maintains.

[4] For completeness, we outline the derivation in [10] here. Let peer i have n_i out of the N fragments where n_i is uniformly distributed in $\{0, \ldots, N-1\}$. When the required total streaming rate is more than the total peer upload capacity, a peer's upload capacity will not be utilized only if every other peer has all the fragments it has. Therefore,

$$\eta = 1 - \sum_{n_i=0}^{N-1} \{\Pr(\textit{Peer } j \textit{ needs no fragment from peer } i \mid n_i)\}^k$$

where j is a peer connected to peer i. As n_j is uniformly distributed in $\{0, \ldots, N-1\}$, $\Pr(\textit{Peer } j \textit{ needs no fragment from peer } i \mid n_i) = \sum_{n_j=n_i}^{N-1} \frac{1}{N} \Pr(j \textit{ has all fragments of } i \mid n_i, n_j) = \sum_{n_j=n_i}^{N-1} \frac{1}{N} \left[\binom{N-n_i}{n_j-n_i} \Big/ \binom{N}{n_j} \right] = \frac{N-n_i}{N(n_i+1)}$. Substituting this in the expression for η gives us (1).

independence, we believe that maintaining a peer group size of 15-20 peers would be a more appropriate target for a peer-to-peer live streaming application.

From Figure 2, we note that the number of fragments available for sharing is a more important factor in determining the level of efficiency possible. We explore this in the next section.

V. Number of Fragments Available for Sharing

Expanding the summation term in Eq. 1, we can write:

$$\eta = 1 - \frac{1}{N} - \frac{R'}{N}$$

where

$$R' = \left(\begin{array}{l} \frac{1}{2^k}\left(1-\frac{1}{N}\right)^k + \frac{1}{3^k}\left(1-\frac{2}{N}\right)^k + \\ \cdots + \frac{1}{(N-1)^k}\left(\frac{2}{N}\right)^k + \frac{1}{N^k}\left(\frac{1}{N}\right)^k \end{array} \right)$$

For peer groups of size larger than 10, the value of R'/N is negligible[5]. Therefore, a reasonable approximation for η, the efficiency of fragment exchange in BitTorrent-like systems, is:

$$\eta \approx 1 - \frac{1}{N} \qquad (2)$$

In Figure 3, we plot η, the efficiency of fragment exchange, against N, the number of fragments available for exchange, using Eq. 2.

As we can see from Figure 3, the efficiency of fragment exchange depends heavily on the number of fragments available for exchange but there are diminishing returns as the number of fragments get large. For example, if the number of fragments available for sharing is doubled from 5 to 10, the efficiency increases from 0.8 to 0.9 (which, for the scenario in Figure 1, translates into a saving of 4 Gbps of server side capacity with 100,000 users) but doubling the number of fragments available for sharing from 40 to 80 increases the efficiency from 0.975 to only 0.9875 (equivalent to a saving of 0.5 Gbps of server side capacity for 100,000 users for the scenario in Figure 1). The diminishing returns behavior of the relation between the number of fragments available for sharing and the efficiency is important as increasing the fragments available for sharing directly increases the lag a user will experience in the stream playback. In the next section, we explore the implications of this on BitTorrent-based live video streaming system design.

[5] R is strictly decreasing with increasing k. In our computations for $k = 10$ and N varying from 2 to 100, the value of R/N never exceeded 3.5×10^{-5} and was decreasing with N for $N > 3$.

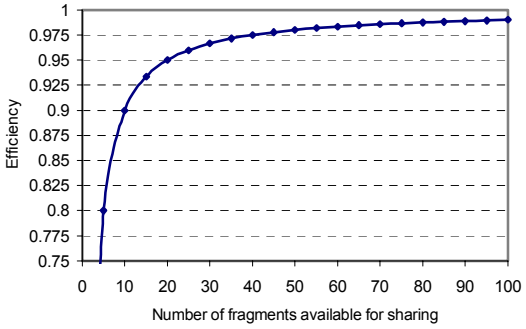

Figure 3. Effect of the Number of Fragments available for sharing on the Efficiency of Fragment Exchange

VI. BitTorrent-Based Live Video Streaming Design

Our analysis in Section 4 indicates that a peer group size of 15-20 peers is sufficient to realize the peer upload capacity utilization benefits a large peer group offers. BitTorrent does try to maintain over 20 peers so the peer group maintenance aspect of the basic BitTorrent protocol does not need any modification.

Let us now consider the implications of our results in Section 5 on the effect of the number of fragments available for sharing on the efficiency of fragment exchange. Let us say that we wish to achieve 97.5% utilization of the peer upload capacity. From Figure 3, this implies that there be 40 fragments available to be shared. For the default BitTorrent fragment size of 256KB, this is equivalent to a buffer size of 10 MB. This buffer is in addition to what a streaming player keeps "internally" to account for network uncertainties. For a 700 kbps stream, 10 MB of buffering is equivalent to an additional playback delay of nearly 2 minutes which is unlikely to be acceptable for a "live" streaming application (this delay does go down for higher-rate streams but even with a 1.5 Mbps stream it remains over 50 seconds).

One option to reduce the playback delay while maintaining high efficiency values is to use smaller fragments. BitTorrent allows fragments as small as 32KB while the sub-piece size is 16KB (BitTorrent breaks each fragment into sub-pieces to allow pipelining of requests). By using a fragment size of 32KB, the required buffer size and, consequently, the playback delay can be cut by a factor of 8 (e.g. for a 1.5 Mbps stream it goes down to under 7 seconds). The argument against small fragment size for BitTorrent file transfers is that a small fragment size will result in a very large torrent file (as a hash for each fragment is included in the torrent file) which increases the load on the server hosting the torrent. In the live streaming case, the fragments are not available ahead of time so this is not an issue. However, there are limitations on how small a fragment ought to be. If we wish to support a 700 kbps stream and target a playback delay on account of achieving high fragment exchange efficiency to be around 6 seconds, this is equivalent to 540 KB. If we wish to achieve 97.5% utilization of the peer upload capacity, we need 40 fragments of buffering or a fragment size of 13.5 KB. The equivalent fragment size for a 350 kbps stream would be 6.75 KB which is

getting close to the popular MTU size of 1500 bytes (if we wish to pipeline requests, a smaller fragment size implies a smaller sub-piece size but going below a sub-piece size smaller than the MTU size will be wasteful).

We can summarize the relation between the playback delay introduced to allow fragment exchange, the fragment size, the streaming rate and the fraction of the peer upload capacity that can be utilized as follows:

$$\eta \approx 1 - \frac{S}{\tau R} \quad (3)$$

where η is the fraction of the peer upload capacity that can be utilized, S is the fragment size, τ is the playback delay introduced to allow fragment exchange, and R is the streaming rate (N fragments of size S each imply a buffer size of NS which at streaming rate R would lead to a playback delay of NS/R; substituting $N = \tau R/S$ in (2) gives (3)).

There are other modifications that one may wish to do to the basic BitTorrent protocol. As we noted in Footnote 1, the peers should use tit-for-tat policy only for peer selection and not "choke" uncooperating peers if that implies not uploading to any peer. One may also wish to alter the tit-for-tat policy to fulfill fragment requests that are close to missing the playback deadlines. One would also want to alter the rarest-first fragment request policy (i.e. asking for the fragment that fewest peers have) to one where the fragments close to missing the playback deadlines are requested with a higher priority (see [7]). There can be other modifications e.g. in the information that the tracker keeps or the list of peers that the tracker returns and we hope to evaluate the other design choices in future work.

VII. CONCLUSION

Using BitTorrent-like peer-to-peer technology to support live video streaming over the Internet is an attractive option for major commercial players in the video streaming area. In this paper, we argue that a pure peer-to-peer live streaming solution cannot deliver high-quality streams and the peer-to-peer solution must be supplemented with server-side capacity. The paper points out the importance of a well-designed peer-to-peer solution in such a system and proceeds to explore the key design factors that determine the server-side capacity that would be needed to support a target streaming rate to a given number of users. We show that a peer group size of 15-20 peers is sufficient to achieve any benefits a large peer group size would offer. A more important factor in determining the server-side capacity that would be needed to support a target streaming rate to a given number of users is the number of fragments that are available for sharing. We provide an analytical expression to compute the fraction of the total peer upload capacity that can be utilized given the number of fragments that are available for sharing. The number of fragments available for sharing directly affects the playback delay users would experience and it is important to balance the goal of small playback delay against the objective of utilizing as much of the peer upload capacities as possible. To inform the selection of the appropriate fragment size, we provide an analytical expression to compute the fraction of the peer upload capacity that can be utilized for a given fragment size given a target playback delay on account of fragment exchange and the streaming rate.

REFERENCES

[1] BitTorrent Protocol, http://www.bittorrent.org/protocol.html.

[2] B. Cohen, "Incentives Build Robustness in BitTorrent," In IPTPS, February 2003.

[3] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "DONet/CoolStreaming: A Data-driven Overlay Network for Live Media Streaming," in Proc. of IEEE INFOCOM'05, Miami, FL, USA, March 2005.

[4] X. Zhang, J. Liu, and B. Li, "On Large Scale Peer-to-Peer Live Video Distribution: CoolStreaming and Its Preliminary Experimental Results", in IEEE International Workshop on Multimedia Signal Processing (MMSP), October 2005.

[5] PPLive, http://www.pplive.com/en/about.html.

[6] X. Hei, C. Liang, J. Liang, Y. Liu and K.W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system", in Workshop on Internet Protocol TV (IPTV) services over World Wide Web in conjunction with WWW2006, May 2006.

[7] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for Supporting Streaming Applications", in Global Internet Workshop in conjunction with IEEE INFOCOM 2006, April 2006.

[8] S. Tewari and L.Kleinrock, "Proportional Replication in Peer-to-Peer Networks", in Proc. of IEEE INFOCOM 2006, April 2006.

[9] ISP Speed Tests, http://www.dslreports.com/archive?c=us.

[10] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-Peer Networks," In Proc. of SIGCOMM 2004, September 2004.

[11] C. Dana, D. Li, D. Harrison, and C. N. Chuah, "BASS: BitTorrent Assisted Streaming System for Video-on-Demand," in IEEE International Workshop on Multimedia Signal Processing (MMSP), October 2005.

[12] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast", in IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Networking Support for Multicast, Vol. 20, No. 8, 2002.

[13] N. Magharei, R. Rejaie, "Understanding Mesh-based Peer-to-Peer Streaming", In Proc. of the International Workshop on Network and Operating Systems Support for Digital Audio and Video, May 2006.

[14] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment", in Proc. of SOSP'03, October 2003.

[15] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient Peer-to-Peer Streaming", in Proc. of IEEE ICNP, November 2003.

[16] G. Wu, T.C. Chiueh, "How Efficient is BitTorrent?," in 13th Annual Multimedia Computing and Networking (MMCN'06), January 2006.

[17] X. Jin, K.-L. Cheng, and S.-H. Chan, "SIM: Scalable Island Multicast for Peer-to-Peer Media Streaming," in Proc. IEEE International Conference on Multimedia Expo (ICME), July 2006.

[18] J. Li, "PeerStreaming: A Practical Receiver-Driven Peer-to-Peer Media Streaming System", Microsoft Research TR-2004-101, Sept. 2004.

[19] Bollobas. B, *Random Graphs*, Academic Press, London, 198.