

# BASS: BitTorrent Assisted Streaming System for Video-on-Demand

Chris Dana  
cdana@ece.ucdavis.edu

Danjue Li  
dli@ucdavis.edu

David Harrison  
dosirrah@gmail.com

Chen-Nee Chuah  
chuah@ece.ucdavis.edu

**Abstract**— This paper introduces a hybrid server/P2P streaming system called BitTorrent-Assisted Streaming System (BASS) for large-scale Video-on-Demand (VoD) services. By distributing the load among P2P connections as well as maintaining active server connections, BASS can increase the system scalability while decreasing media playout wait times. To analyze the benefits of BASS, we examine torrent trace data collected in the first week of distribution for Fedora Core 3 and develop an empirical model of BitTorrent client performance. Based on this, we run trace-based simulations to evaluate BASS and show that it is more scalable than current unicast solutions and can greatly decrease the average waiting time before playback.

## I. INTRODUCTION AND RELATED WORK

With the increasing growth of consumer broadband, multimedia applications such as video-on-demand (VoD) have become accessible to a larger audience than ever before. Current VoD services mainly rely on content distribution networks (CDNs) and local streaming proxies to increase system scalability and shorten the delay perceived by end users. However, system performance still deteriorates rapidly as the number of clients increases due to the current server-client service model. When a flash crowd appears, the server can be easily overloaded.

To solve this problem and increase scalability, a number of end-system multicast (ESM) and non-multicast peer-to-peer (P2P) video streaming systems have been developed. For live media streaming, ESM is favorable. One such system, CoopNet [1], builds a multicast tree by establishing *parent-child* relationships between end systems that govern how data is routed. Castro et al. proposed Splitstream [2], which is built upon a distributed hash table (DHT) based overlay network called Pastry [3]. Splitstream also builds multicast trees but is decentralized and highly scalable.

The peer-to-peer concept has also been applied to more general video-on-demand services as well. To the best of our knowledge, Chaining [4] is the first work to do this. In Chaining, each client caches a small portion of recently received video content. Newly arriving clients can stream from an earlier client as long as the earlier client still has the first block of the video cached. Pinho et al. introduced a scalable VoD system called GloVE [5], where active clients cooperate to create a sharable video cache as the primary source of video content for subsequent client requests. Hefeeda et al. [6] proposed a P2P media streaming system called PROMISE, which relies on an application level P2P service called CollectCast

to select peers and dynamically adapt to network fluctuations and peer failures. A recent work on P2P streaming called Coolstreaming [7] appears similar to BitTorrent at first glance. However, rather than being tracker-based, Coolstreaming uses a gossip protocol to disseminate lists of peers. In addition, it has very different internal policies to cope with the strict deadlines that video streaming imposes. However, almost all past P2P-based streaming systems completely rely on peer connections, which make the system vulnerable to peer or connection failures. In this paper, we combine P2P techniques with the current *server-client* streaming model to build a hybrid system that is both scalable and robust.

Specifically, the contributions of this work are:

- First, we propose a novel streaming system, *BitTorrent Assisted Streaming System (BASS)* for VoD services, where we add the use of an external streaming server to a slightly modified BitTorrent. Clients can simultaneously stream from the media server as well as each other via BitTorrent P2P connections. By maintaining these connections, we can reduce the aggregate bandwidth used by the media server and decrease client waiting times.
- Second, by analyzing the BitTorrent traces collected in the first week of distribution for Fedora Core 3 [8] distribution hosted at *linux.duke.edu*<sup>1</sup>, we present a method for modeling the download performance of a BitTorrent peer. To this end, we extend previous analysis of BitTorrent [9] by focusing on the evolution of the download rate distribution over time. Based on this, we derive a simple model for the downlink throughput of a BitTorrent client.
- Third, we define the hybrid server/peer streaming process in BASS as a delay-restricted-file-sharing (DRFS) problem. By integrating the BitTorrent trace analysis model, we design trace-based simulations to evaluate the BASS performance and present our simulation results.

The remaining parts of this paper are structured as follows. Section II presents BASS. Section III introduces the trace based simulation studies to evaluate BASS performance and presents our simulation results. We will summarize our work and discuss future research directions in Section IV.

## II. BASS: BITTORRENT ASSISTED STREAMING SYSTEM

In this section, we present our proposed hybrid video streaming system. Figure 1(a) illustrates BASS for VoD services, where clients can stream from each other via BitTorrent P2P connections and media servers simultaneously.

This work is supported in part by the UC Micro 04-05 program with matching funds from Hewlett Packard.

<sup>1</sup>We are grateful to Seth Vidal for making this data available to us.

The traditional server-client solution to multimedia distribution is simply to send a separate copy to each client that requests the file. The bandwidth usage at the server scales linearly with the number of concurrent users and a flash crowd can quickly overwhelm the system. For general file distribution, BitTorrent targets this problem by leveraging the upstream bandwidth of the clients. It splits a file into many pieces and sends different pieces to different clients, allowing them to trade pieces amongst each other. Clients that have completed the file and are only uploading are known as *seeders* and those that are still swapping pieces are *leechers*. BitTorrent uses a *tracker* program running on a server (as opposed to a gossip protocol) to disseminate lists of peers. To govern how pieces of the file are requested and swapped amongst peers, it follows *rarest-piece-first* and *tit-for-tat* policies [10], respectively. In *rarest-piece-first*, the client requests a piece based on the number of copies it sees available and chooses the least common one. In *tit-for-tat*, a leecher reciprocates to other leechers that send it pieces by giving higher priority to their requests. The interested reader is directed to a detailed explanation of the BitTorrent protocol [11].

For multimedia streaming, end-system multicast can suffer from degradation in quality as the tree depth increases. Lossy wireless access points and congestion at access routers (causing delays or tail drops) become greater issues when the data has to traverse a greater number of hops. In addition, there is no incentive for clients to contribute upstream bandwidth other than good faith. BitTorrent suffers from neither of these, but due to the *rarest-piece first* policy, is wholly inappropriate for multimedia streaming, which requires in-order reception of data. Simply forcing BitTorrent to request pieces in-order would be similarly disastrous because then clients would only contain subsets of each others data and *tit-for-tat* would fail.

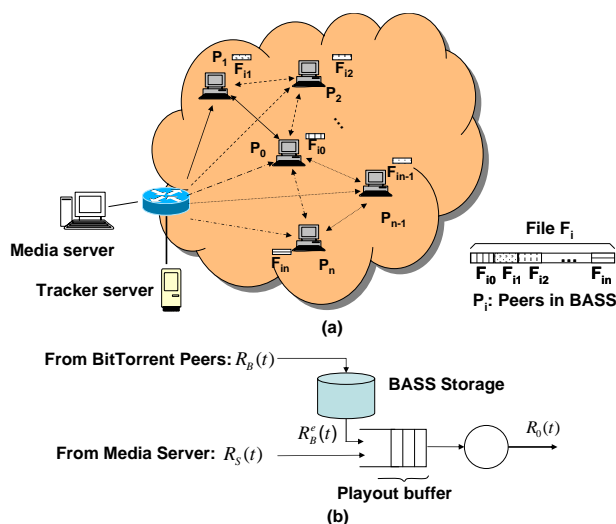


Fig. 1. BASS: (a) System Overview, (b) Client Model

Instead of trying to modify BitTorrent for streaming (which it was not designed to do), BASS augments it with an external media server (Figure 1a), with the only modification to BitTorrent being that it does not download any data prior to the current playback point. It is allowed to use the *rarest-piece-first* (subject to the previous condition) and *tit-for-tat* policies. As shown in Figure 1b, data from BitTorrent is held in local storage until it is needed. From the media server, BASS downloads pieces in-order, skipping over pieces that have already been downloaded by BitTorrent, or are currently in the process of being downloaded and are expected to finish before their playout deadline arrives. If the media server were altered to limit the amount of data a client is allowed to stream from it, BASS could also encourage users to participate in distribution using the *tit-for-tat* policy.

### III. TRACE-BASED SIMULATION STUDY

In this section, we will present the BitTorrent trace analysis, discuss how we simulate BASS, and show trace-based simulation results that compare BASS performance to existing server-based solutions.

#### A. BitTorrent Trace Analysis

Due to its popularity, BitTorrent has led to a number of papers analyzing [12], [13] and modeling [14] its performance. Our work falls in between the two categories, analyzing data to generate a simple model for the download performance. To this end, we were able to obtain tracker logfile for the first week of release of Fedora Core 3 (FC3), an open-source operating system. The logfile actually contains data for six releases of FC3 (three binaries and three sources), but we focus on the single, most popular one, accounting for 37% of the total data.

Since some clients send incorrect information to the tracker, we must first sanitize the data by removing entries for clients that are obviously incorrectly configured or otherwise misbehaving. Examples include entries with negative bytes downloaded or uploaded, multiple instantaneous resets, and misleading (or missing) *event* flags among others. We also focus our analysis on *single-session downloaders*, that is, clients who download the entire file in one shot, since this corresponds more closely to user behavior for streaming media. For each client we record a series of  $time_{start} < t < time_{stop}$  intervals and calculate the average download rate  $\bar{r}_{down}$  on that interval as  $\bar{r}_{down} = \frac{B_{stop} - B_{start}}{t_{stop} - t_{start}}$ , where  $B_{start}$  and  $B_{stop}$  are the bytes downloaded up to  $t_{start}$  and  $t_{stop}$ , respectively. We then generate the download density every five minutes, corresponding to the default interval of time that clients wait to request more peers from the tracker server.

Using the tracker logfile, we determine the average download rate for each peer-tracker intercommunication interval. We apply the Freedman-Diaconis rule that sets bin width  $w = \frac{2IQR}{K^{1/3}}$ , where IQR is the interquartile range of the samples and  $K$  is the number of samples. Scaling to make the area

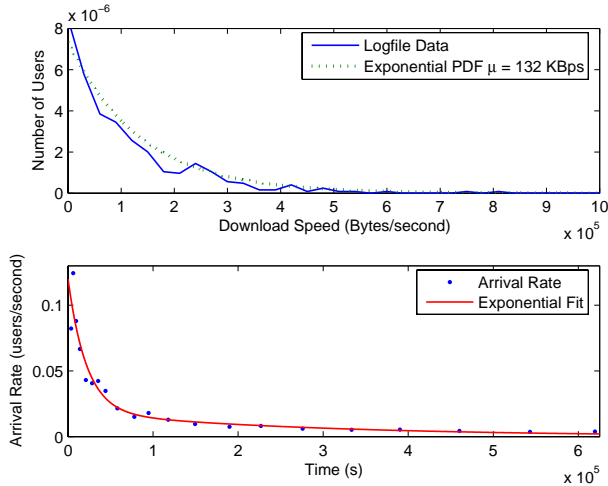


Fig. 2. (top) Typical density of download rates with and exponential fit (bottom) BitTorrent client mean arrival rate across time

integrate to unity yields an exponential probability density function with mean  $\mu$  equal to 132KBps (Figure 2 (top)).

Another important aspect of the system we examine is the arrival rates of users across time. We find that the average arrival rate is well approximated by a double-exponential curve. In the case of our tracker logfile, we found that

$$\lambda(t) = .3081e^{-0.00004287t} + .0397e^{-0.000002261t}$$

Given the average arrival rates and download rates over time, as well as their distributions, we construct a purely *leecher-side* model for BitTorrent performance.

### B. Simulating BASS

Using the results of our trace analysis, we can simulate BASS for various performance metrics. We begin by splitting the file into windows of length  $\alpha$  seconds, which will be the minimum unit over which we calculate contributions from the various sources. Another window is  $\beta$ , which denotes the range of the file over which BitTorrent operates. We set  $\beta$  equal to the length of the entire file both for simplicity and because it allows BitTorrent to achieve the best possible performance. Our major assumption is that the probability of getting future pieces is uniform and thus we spread the information gained from BitTorrent evenly over  $\beta$  (Figure 3). For a system where the number of BASS users is small compared to the number of BitTorrent users, it is reasonable that this assumption holds due to the rarest-piece-first policy. If the majority of clients are using BASS, then the in-order downloading from the media server would most likely skew the distribution of pieces among peers and rarest-piece-first might not be able to compensate to make our assumption hold. For now we assume there are many fewer BASS users than BitTorrent users and leave the other case to future work.

To examine the performance over time, we use the parameters gleaned from trace analysis to run a simple queueing

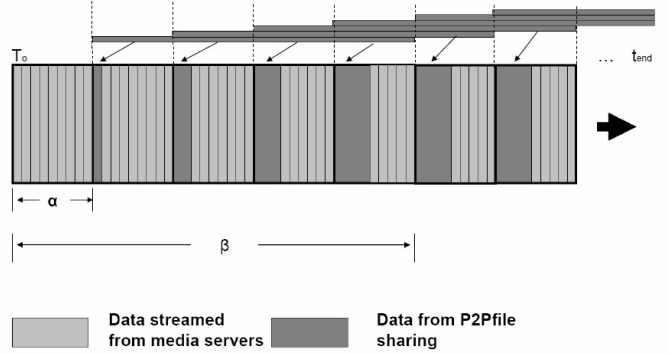


Fig. 3. Diagram of data handling in the simulation. In general  $\alpha \leq \beta \leq filelength$ . We use  $\beta = filelength$

system simulation. Clients arrivals are Poisson with the time-varying parameter shown in Figure 2 (bottom). Upon arrival they are assigned a rate from BitTorrent according to the exponential distribution in Figure 2 with a constant mean. This rate is maintained until the client finishes its download, and in our case, no client leaves before it has finished.

### C. Simulation Results

We begin by looking at the bandwidth required to support streaming for a 173 MB, 22 minute file (with a resulting average playback bitrate of 131 KBps). The server-only case requires an average of 131 kbps per user while BASS requires 87 kbps, achieving a savings of 34%. At the beginning of the simulation, during the heaviest client arrival rate, BASS performance is similar to the server-only case. However, once the arrival rate has subsided, BASS scales linearly with the number of users in the system.

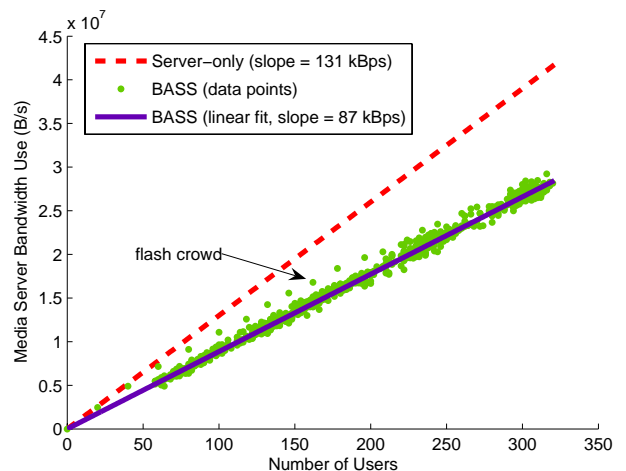


Fig. 4. Required throughput as a function of concurrent streaming clients.

Next we look at the time clients must wait before playback if they are given a certain bandwidth guarantee by the media

server (Figure 5). For this metric we let the server provide bandwidth up to the playback bitrate. Because we use an exponential distribution of download rates from our trace-based model, the possibility exists that the BitTorrent download rate can be arbitrarily close to zero. In order to determine when pieces are successfully downloaded, we divide the number of bytes by this rate, the result being a random variable that does not easily (if at all) converge to a finite mean. To avoid this problem, we lower-bound the rate to 5 Kbps, which greatly reduces convergence time. Using only BitTorrent (Figure 6), we must wait a constant amount of time since the possibility exists that the final piece downloaded is the first playout piece in the file. In contrast, for the server-only case, all of the pieces are downloaded in order and the wait time simply depends on the ratio of the download rate to the playback bitrate. As expected, BASS behaves as a mixture of the two, with the wait time upper-bounded by the Bit-Torrent only case and then decreasing in a curve similar to the server-only case. We see that if half of the playback bitrate is guaranteed, then the average wait time for BASS is 27% of the wait time for server-only and 10% of that for BitTorrent only.

#### IV. CONCLUSION AND FUTURE WORK

We have proposed a hybrid server/P2P streaming system, called BitTorrent-Assisted Streaming System (BASS), for large-scale Video on Demand (VoD) service. By analyzing the torrent trace data collected in the first week of the Fedora Core 3 release, we develop a simple model for evaluating BitTorrent client performance and integrate that into simulations to evaluate BASS. Simulation studies show that BASS can efficiently reduce the streaming load on the media server, and provide much lower initial streaming delay than relying solely upon the server. In the future, we will look at other performance metrics such as the probability of successfully receiving a stream given an overall cap on the media server (versus individual rate guarantees). We also plan to implement BASS and compare its performance both simulation and analytical models.

#### REFERENCES

- [1] P. Chou V. Padmanabhan, H. Wang and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," May 2002.
- [2] A.-M. Kermarrec A. Nandi A. Rowstron M. Castro, P. Druschel and A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," October 2003.
- [3] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," .
- [4] Simon Sheu, Kien A. Hua, and Wallapak Tavanapong, "Chaining: A generalized technique for video-on-demand systems," in *ICMCS*, June 1997.
- [5] L. Pinho, G. Amorim, and E. Ishikawa, "Glove: A distributed environment for low cost scalable vod systems," in *SCAB-PAD*, October 2002.
- [6] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "Promise: Peer-to-peer media streaming using collectcast," in *ACM International Conference on Multimedia*, Berkeley, California, USA, Nov. 2003.
- [7] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Donet/coolstreaming: A data-driven overlay network for live media streaming," in *IEEE INFOCOM*, 2005.
- [8] "Fedora Core 3," <http://fedora.redhat.com/>.

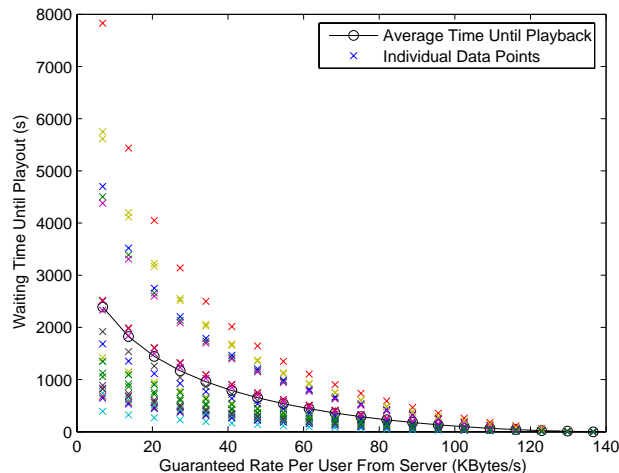


Fig. 5. Client waiting time given a certain media server guarantee

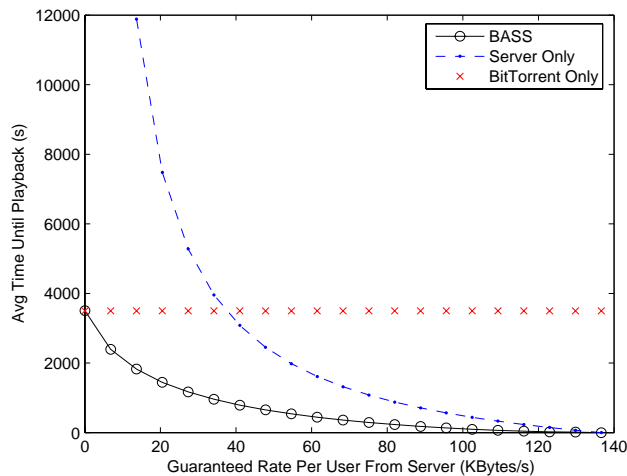


Fig. 6. Comparison of avg. waiting time for BASS, server, and BT only

- [9] C. Dana and C.-N. Chuah, "Perceiving methods for the file distribution performance of bitTorrent," Tech. Rep. ECE-CE-2004-2, University of California, Davis, August 2004.
- [10] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer Systems*, May 2003.
- [11] "Bittorrent protocol," <http://www.bittorrent.com/protocol.html>.
- [12] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice, "Dissecting bittorrent: Five months in a torrents lifetime," in *PAM Workshop*, April 2004.
- [13] D.H.J. Epema H.J. Sips J.A. Pouwelse, P. Garbacki, "The bittorrent p2p file-sharing system: Measurements and analysis," February 2005.
- [14] X. Yang and G. Veciana, "Service capacity of peer to peer networks," in *IEEE INFOCOM*, March 2004.