Storage Optimization for a Peer-to-Peer Video-on-Demand Network *

Jagadeesh M. Dyaberi, Karthik Kannan, and Vijay S. Pai
Purdue University
West Lafayette, IN, USA
jdyaberi@purdue.edu, kkarthik@purdue.edu, vpai@purdue.edu

ABSTRACT

This paper explores requirements for efficient pre-seeding of video-on-demand (VoD) movie data onto numerous customer set-top boxes in a cable ISP environment. The pre-seeded content will then be distributed to other set-top boxes in the same cable community using a peer-to-peer (P2P) network protocol such as BitTorrent. The challenges and solutions required for P2P VoD provided by a fixed provider such as a cable company are fundamentally different from those seen in traditional P2P networks or client-server VoD solutions.

Our work pre-positions data into set-top boxes using a mathematical programming algorithm. The objective of the algorithm is to minimize uplink traffic, given a popularity model for various pieces of content and information about storage and bandwidth capacity constraints at the customer nodes. Given the complex non-linear nature of P2P interactions, these mathematical programs are solved using non-linear optimization approaches. Using a BitTorrent-like peer-to-peer data delivery system, we show through extensive simulations that our mathematical model for pre-seeding data based on object popularity and node bandwidth availability leads to noticeably greater reductions in uplink traffic and VoD server load than a weighted-random pre-seeding scheme that only considers object popularity.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems–Distributed Applications; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks–Internet; H.5.1 [**Multimedia Information Systems**]: Video

General Terms

Performance, Experimentation, Design, Measurements, Algorithms

*This work is supported in part by the National Science Foundation under Grant Nos. CCF-0532448 and CCF-0621457, and by AT&T Labs under a VURI award.

MMSys'10, February 22–23, 2010, Phoenix, Arizona, USA.

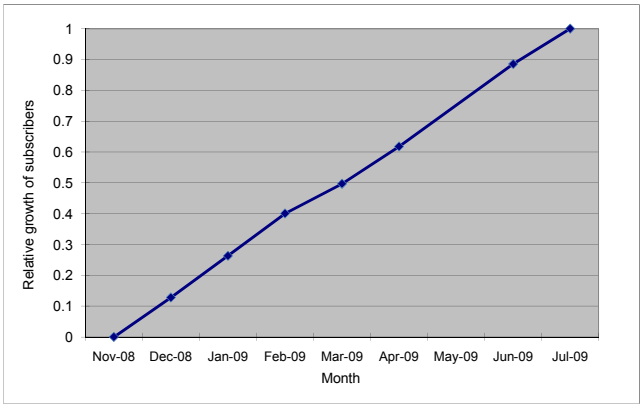


Figure 1: Relative growth in VoD subscribers from November 2008 to July 2009

Keywords

Video-on-Demand, Multimedia Streaming, BitTorrent, Peer-to-Peer, Experimental Systems, Nonlinear Programming

1. INTRODUCTION

As systems such as voice, video, and data communication merge onto a single IP-network delivery platform, users are likely to benefit with a greater number of choices and more content control [10]. Consequently, cable companies have made significant investments in systems that seek to deliver high-bandwidth digital content (e.g., Video-On-Demand service for full length, high-definition movies) to consumers. Users have responded with an ever-increasing number of IPTV/cable VoD subscriptions and views.

Apart from IPTV and cable VoD, VoD on the Internet has also grown. Several companies like Amazon, Hulu, and YouTube have expanded their offerings significantly in the past year [1, 3, 4]. In this paper, we focus solely on IPTV/cable-based VoD.

Figure 1 shows the relative increase in US IPTV subscribers for a major IPTV provider from November 2008 to July 2009, with the month of November considered as the base month. Data shows an average subscriber increase of 7.5% *per month*. Figure 2 shows the number of VoD requests made per-day by all the IPTV customers between February 2009 to July 2009[1]. This Figure shows a growth in VoD requests over the five-month period. There are approximately 75% more requests made per-day in July than in February.

[1]Both graphs show relative numbers or no numbers on the y-axis. We cannot provide the absolute numbers on the y-axis due to a confidentiality agreement with the content provider.

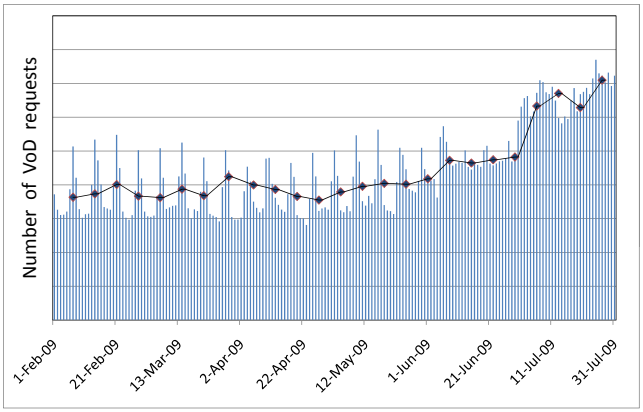


Figure 2: Number of VoD requests per day from February 2009 to July 2009

The vertical columns represent the number of VoD requests made per day, with spikes on weekends. The line with markers on the graph represents the average number of daily VoD requests for each week. The increase in subscribers and VoD requests and content pose huge challenges to IPTV providers as they seek to meet the required QoS.

IPTV service providers currently employ a unicast data delivery model. Figure 3 illustrates this data delivery model. All the VoD data is streamed from the regional or national-level video hubs to the requesting viewers. Viewers in this environment use a set-top box (STB) provided by their service provider to access IPTV content. The local community switch (DSLAM) is connected to regional video hub servers via an uplink connection, which currently has a bandwidth of around 1 Gbps [10]. This uplink becomes a bottleneck in determining the number of concurrent viewers when using unicast data delivery. The unicast delivery model is not easily scalable to thousands of clients. Further, the introduction of High-Definition (HD) format further strains the uplink, as it requires three times more bandwidth than Standard-Definition (SD).

One solution to meet the increasing demand for VoD content is to increase the bandwidth of the uplink between the community and the server hubs. This infrastructure upgrade is very expensive and could result in underutilization of link bandwidth if there are only a few concurrent active viewers at any time. A feasible solution to reduce uplink load and extend the network is to use P2P communication among set-top boxes in a community. Although set-top boxes can store viewer-selected content through resources such as DVR (Digital Video Recorder), they are essentially largely under control of the system provider. Thus, the system provider may use their resources as a way of extending the capabilities of the network. IPTV providers can reserve part of the DVR capacity and also the upload bandwidth from a household for their purpose of extending the network. This approach could exploit any unused bandwidth in the community network and any unused capacity in the user set-top boxes.

The other likely bottleneck that can occur at the regional server hubs is the server disk bandwidth. Although disk arrays can help to improve aggregate bandwidth, their effective throughput may nevertheless be limited by disk allocation and workload characteristics. Hence to satisfy the real-time constraint of video data delivery, service providers may have to employ several servers to meet the request demand, incurring substantial costs for deployment, power, cooling, and space. By instead offloading data transfer to the community P2P network, the disk bandwidth problem is also alleviated.

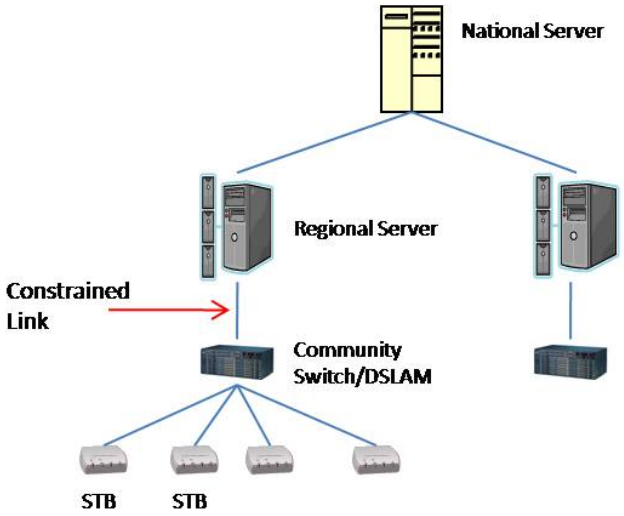


Figure 3: IPTV architecture

Previous work by Choe et al., called Toast, has modified the BitTorrent protocol to enable peers (set-top boxes) to stream data to each other, while still using the origin VoD servers to provide data chunks when no peer could provide the data in a timely fashion [11]. Toast modified the BitTorrent piece selection policy and evaluated various peering systems including one in which peers continue to serve data even after they have completed viewing their streaming media. The experimental evaluation of Toast showed that the P2P system can offload up to 70-90% of data demand from the origin server for a single media stream, thus validating the use of P2P communication for streaming. However, this model was not applied to a full VoD library, nor can it be trivially extended as such.

This paper demonstrates challenges and solutions to deploying P2P data delivery for video content delivery of a large number of movies in a fixed-provider environment. First, porting a traditional P2P model into a environment that is largely centrally-controlled would underutilize the available resources. If end nodes are entirely responsible for making decisions about which content to store based on their own on-demand viewing, the system will not be increasing its capabilities whenever end nodes are idle. This is a substantial issue in situations like home VoD that typically see idleness for most of the day followed by a brief period of high activity in the evening and early night. Instead, the system should try to preload desired data onto the set-top boxes and thus make all set-top boxes active participants in the P2P network even if they are not consuming P2P resources. Figure 4 shows the average number of requests from STB's during the entire 24 hour period [9]. We observe a low-load period between 2–8 AM where the graph reaches its lowest points. This provides the service provider an excellent opportunity to pre-seed data onto set-top boxes with minimal disruption to the viewers.

Second, the issue of set-top box capacity becomes a concern when considering all the content available from the provider (which can reach thousands of movies). The total capacity among all the set-top boxes in a community is far less than the capacity needed to store all the movies, so only a subset of the total content can be stored in the community. Further, if the movies are encoded in High-Definition, capacity becomes an even greater constraint.

Third, bandwidth is closely tied with capacity as both must be considered jointly to maximize overall streaming efficiency. This

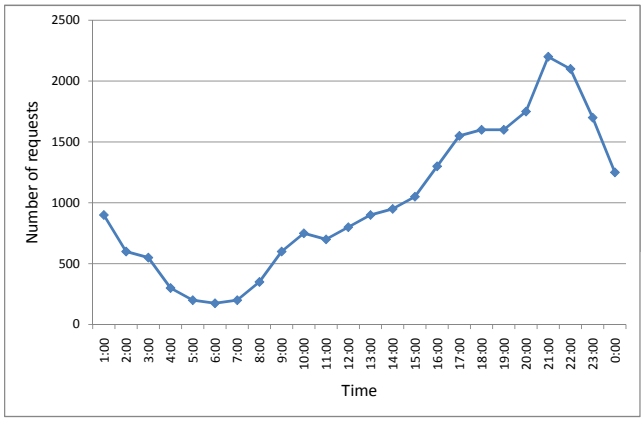


Figure 4: Average number of VoD requests during a 24-hour period [9]

is a concern as all networks currently exhibit substantial asymmetry between the uplink and downlink bitrates at any given end node (typically about an order of magnitude). If each end node has an uplink of 1 Mbps, it would require six sources simultaneously contributing data to satisfy a single request for a HD-quality streaming movie encoded at 6 Mbps; each of these sources would be fully saturating its uplink. This issue now relates back to capacity, because offloading a substantial amount of streaming traffic onto the P2P network requires that enough copies of the data exist in the network to account for the asymmetry in uplink and downlink rates. This requires replication, which reduces the effective aggregate capacity in the network and makes the subset of content that can be stored in the network an even smaller fraction of the total.

Because the P2P network can only store a subset of the data content available, any preloading solution must intelligently choose which data to load and replicate. An effective decision would require an understanding of the popularity of different pieces of content, as the greatest benefits would arise from replicating the most popular content. At the same time, the high demand for these most popular videos would require extensive replication to be able to support as many simultaneous requests as are likely to arrive.

This paper presents and prototypes intelligent capacity allocation policies using mathematical models that seek to maximize the streaming availability and efficiency within the community network. Putting together capacity and bandwidth constraints, the mathematical models are used to build up a "presence matrix" of pieces of content by carefully considering capacity constraints (making sure that no given node is assigned too much data) and bandwidth constraints (making sure that no node's uplink is overly strained by heavily replicating the most popular pieces of content and distributing the popular content across different sets of nodes so that no node is particularly more demanded than others) while optimizing for an objective function. The current objective function is to minimize the aggregate bandwidth demand at the servers. Real-time data delivery is guaranteed by building this on top of the Toast infrastructure, which falls back to the origin server if no peer can deliver the requested data within the expected real-time constraint [11]. Experimental results obtained via simulations show that using our intelligent capacity allocation, we are able to reduce central server load by 50%. The optimized allocation scheme outperforms a popularity-aware heuristic solution (weighted random allocation). Optimized allocation is also quite robust in the face of unexpected popularity shifts.

The rest of the paper is structured as follows: Section 2 provides background on IPTV and the BitTorrent protocol. Section 3 describes our system setup and the optimization algorithm. Experiments and results are discussed in Section 4 while related work is described in Section 5. Section 6 concludes the paper.

2. BACKGROUND

2.1 IPTV characteristics

In an IPTV environment, customers are provided with a set-top box (STB) to connect to the network and be able to access the IPTV service. Almost all STBs have an in-built hard-drive to use the STB as a DVR. Hard drive capacity of the STBs exceed 100 GB and with storage getting cheaper every year, this number will increase rapidly as time progresses [19]. These STBs are always on, since customers generally do not power down the STBs by unplugging the power cord from the electrical wall socket. Thus there is no "churn" in the system and the STB resources are always readily available. IPTV service providers must decide how best to exploit the resources provided by these STBs to enhance their services, thus maintaining consumer satisfication and profitability. Hence it is important to study user viewing behavior to help decide the right content that needs to be pre-seeded on the STBs. To this end, we analyzed the viewing habits of users all who are served by the same regional hub. Characteristics influencing viewing behavior include recency of content and certain external factors like the occurrence of major events.

One of the interesting trends we observe from our analysis of actual IPTV logs is that between any two consecutive weeks, the titles of videos viewed overlap by as much as 47%. Although the number of views of these overlapped videos reduces in the second week, they still make up significant numbers. Analysis shows that recently introduced content (for example, movies added to the content library at the beginning of the week) are 52% more likely to be viewed than movies that have been part of the content library for a longer time. We observed that six of the ten most popular movies (including all of the top five) for the week were added to the content library at the beginning of the week. Amazon's VoD service also showed the same trend where the seven of the top ten movies were recently introduced content [1]. Apart from movies, TV shows for which episodes were aired in the previous week tend to be viewed more on the day after or in the following week. Hence if a certain community is consistently viewing a particular TV show in high numbers, then it is easily predictable that they will continue to watch the same show and mostly the recently aired episodes.

External factors like the death of a celebrity or the winning of major awards can also contribute to viewing patterns. For example, after the death of Michael Jackson, videos featuring the artist increased in popularity. CNET UK reported that the week after Michael Jackson's death, 25 of the top 30 music videos on iTunes belonged to Michael Jackson [2]. In the previous week, there were no videos of the artist among the top sales. The other example of the influence of an external factor is the wining of major awards by a movie. Thus in the week following the Oscar awards, award winning movies are far likely to be viewed than movies that are not. Our data analysis showed that viewing of *Vicky Cristina Barcelona*, a previously less popular movie, increased by 600% in the week following its receiving an Oscar award.

2.2 BitTorrent

BitTorrent has become the most popular file distribution protocol on the Internet [13]. This is primarily due to its efficiency and speed in transferring files. Previous P2P systems usually consisted primarily of a method to search for and locate files shared on the

network. Once found, a peer simply requested the file from another peer, which transferred it using HTTP or a similar protocol. These systems were primarily differentiated by their methods of locating content, but were all similar with respect to their transfer methods. BitTorrent on the other hand ignores the search problem. Instead, it relies on web sites or other common distribution methods to distribute small files called *torrent* files (sometimes called "dot torrent" files, due to their filename extension), each of which is essentially a descriptor of a file or group of files to be downloaded. Each file to be distributed has its own torrent file, and the group of clients downloading a particular torrent is called a *swarm*. Each swarm is independent and self-contained, but individual clients may participate in more than one swarm at a time. The swarm is managed by a simple network server called a *tracker*, which is responsible for keeping track of all clients in the swarm, and informing clients about each other. The tracker does not upload or download any file data. Clients that have the entire file are called *seeds* and downloading clients which do not yet have the whole file are called *peers* or *leechers*. Peers become seeds once they have the whole file, and there are no distinctions between the seeds. (In particular, there are no differences between the original seed run by the original distributor of the file and other clients that have become seeds and are still participating in the swarm.) In most cases, once they have the entire file, clients will continue to participate as seeds until the user closes them.

One of the major innovations in the BitTorrent approach is that each file is split into a number of small pieces, and these pieces are transferred out of order. This means that peers that have different pieces of the file can exchange them, and that a peer can download different pieces from several other peers at once. In fact, since transfers are made at a granularity even smaller than the piece size, even a single piece can be downloaded from several peers at once. This can greatly increase the speed at which a file is transferred compared to simply downloading all of it from a single peer.

3. SYSTEM DESIGN

3.1 Overview

Our system is similar to Toast: a hybrid of a modified BitTorrent and a simple unicast VoD system [11]. We extend Toast to support multiple streams instead of a single movie stream. Each client can be pre-seeded with content up to the storage capacity limit of the client. The decision to pre-seed clients with content is based on an optimization formulation as described in the next section. The actual pre-seeding can be done during a daily low-load phases as shown in Figure 4. When data is actually demanded, the VoD server ensures there is no loss in quality-of-service by streaming pieces to clients if the P2P network cannot deliver the data at all or within the desired time. As in Toast, the goal of the P2P network is to offload data traffic from the servers.

Unlike standard BitTorrent, this streaming-oriented P2P system has an "in-order" piece-picking policy, by which clients fetch pieces from their peers in the order that the movie is being viewed. Similar to Toast, the piece-picking policy implements the feature of "giving up" and not selecting pieces that are too close to the current stream position to download on time. For example, if a missing piece that represents 4 seconds of video will be needed in less than 4 seconds, and the remote client's upload rate is less than the video bitrate, then the download cannot finish before the piece is needed, so the picker will skip it and choose a piece further ahead. The amount of this lookahead is estimated based on the length of the piece, the client's upload rate, and the video bitrate.

Since our clients (set-top boxes) are under the control of the content provider, we assume that users cannot modify the set-top box to either change the contents or unfairly download content while not uploading to other clients. No client is a free-loader, and all of them contribute fairly to the system. We thus further modify the client uploading protocol by removing the choking/unchoking mechanism, tit-for-tat fairness algorithm, and "rarest first" piece-picking policy, as all of these are designed to improve aggregate behavior of the swarm by increasing the likelihood of reciprocal sharing.

3.2 Pre-seeding Optimization Formulation

As mentioned, the clients are pre-seeded with content to assist in VoD streaming. However it is important to do intelligent pre-seeding so that we maximize the requests served by the P2P network and reduce the load on the uplink and the central server. The decision on what content is pre-seeded at which client is determined by a non-liner programming formulation. The formulation seeks to minimize the number of requests that are served by the regional server, thus ensuring optimal usage of the P2P network. The Non-Linear Programming (NLP) formulation is motivated by previous work done by Tawarmalani et.al, who analyzed object allocation in a network of caches for sharing web content [32].

We model the setup as having N equally sized objects and M caches. Let $i \in \{1, ..., N\}$ represent the individual objects and $j \in \{1, ...M\}$ represent the individual peers (the terms nodes, peers, STBs, and clients are used interchangeably in this paper) in the system. We assume that the peers have identical capacities, each with the ability to hold C pre-seeded objects. We also assume that requests arrive at the system at rate of λ, following a Poisson process. A given request is for object i with an exogenous probability P_i. This represents the expected popularity of the object compared to the other available objects.

There are two stages involved in this problem context. The objects are pre-seeded in the first stage. They are then distributed in the second stage depending on the type of request. In this paper, we do not seek to find the optimal allocation across both stages. Such optimization would involve determining the optimal delivery mechanism subject to the optimal allocation, which in turn depends on the delivery mechanism. Instead, we fix the second stage as adopting a fairly straightforward mechanism and focus on the optimal pre-seeding problem in the first.

Our decision variable in the first stage is X_{ij}, which is an indicator to represent that object i is pre-seeded at node j. Our second stage mechanism is as follows. If an object is requested at a particular node, the object search is executed in a sequential manner, beginning with the node indexed 1. If one of the searched peers holds the object (i.e., $X_{ij} = 1$) and is capable of serving the request (i.e., its uplink bandwidth is not occupied because of serving another request to some peer), the search is stopped and the object is served. If none of the peers are able to serve the object, then the object is procured from the origin server. We are interested in minimizing these requests to the origin server.

For characterizing the second stage, we introduce two endogenous variables: Y_{ij} is the probability that object i is served by node j and F_j is the probability that node j is free to serve an object. In order to define F_j, we let R_n represent the probability that there are n active requests in the system at the same time. We make an additional assumption that the uplink of any peer is capable of serving only one peer at any given time. A node can however download data from muliple peers simultaneously. We also assume that it takes an exogenous t periods to service the object request.

Given this model, the first stage optimization problem can be characterized as follows:

$$\min_{X_{ij}} \quad \sum_i P_i \prod_j (1 - Y_{ij}) \tag{1}$$

$$\text{s.t.} \quad F_j = \sum_n R_n \left(1 - \sum_i (P_i Y_{ij})\right)^n \tag{2}$$

$$Y_{ij} = X_{ij} F_j \prod_{j'<j} (1 - X_{ij'} F_{j'}) \tag{3}$$

$$\sum_i X_{ij} \leq C \tag{4}$$

The objective function, Equation (1) seeks to minimize the probability that a given request must be served by the central server and the request is not served by any of the peers (product of $(1 - Y_{ij})$ terms). Equation (2) expresses whether node j is free to serve any request. The node will be free if it does not serve any object over every possible number of requests in the system. Equation (3) indicates that node j will serve a request for object i if object i is present at node j, node j is free, and no lower-numbered node can serve the object. Constraint (4) limits the number of pre-seeded objects at a node to be C.

We can linearize Equation (3) to be as follows:

$$Y_{ij} = X_{ij} F_j \left(1 - \sum_{j'<j} Y_{ij'}\right) \tag{5}$$

The linearization step in proved using induction in Appendix A.

Exploiting the assumption of Poisson request arrival, R_n is:

$$R_n = e^{-\lambda t} \frac{(\lambda t)^n}{n!} \tag{6}$$

Factoring this into Equation (2) and simplifying gives us

$$\begin{aligned} F_j &= \sum_n e^{-\lambda t} \frac{(\lambda t)^n}{n!} \left(1 - \sum_i (P_i Y_{ij})\right)^n \\ &= e^{-\lambda t} \sum_n \frac{\left(\lambda t \left(1 - \sum_i (P_i Y_{ij})\right)\right)^n}{n!} \end{aligned} \tag{7}$$

Although n will be bounded by the number of possible streams that can be viewed in a peer-to-peer network (likely 1–2 per STB), this number will be quite large and can be approximated as infinite. The summation in Equation (7) thus corresponds to a Taylor series expansion of the exponential function, simplifying as:

$$\begin{aligned} F_j &= e^{-\lambda t} e^{\lambda t (1 - \sum_i (P_i Y_{ij}))} \\ &= e^{-\lambda t \sum_i (P_i Y_{ij})} \end{aligned} \tag{8}$$

Although sequential scanning of nodes seems to contradict the necessity of load balance, the optimization formulation can select the X_{ij} matrix in such a way as to maintain load balance despite this ordering.

The above formulation assumes that a client is not free any time for the entire duration that it is serving any request. This is technically not true if the client is serving a request that originates at the same node; however, handling this case separately adds further complexity to the already non-linear problem. Extending the formulation to account for these cases is a subject of future work. Section 4 shows that this restriction on the optimization model does not affect the results significantly. While the formulation does provide the optimal solution, there may be instances where other optimal solutions may exist for the same input, with each optimal solution generating identical (or near identical) objective function values.

3.3 Discussion

In this section, we show using numerical examples how the formulation behaves under different conditions of load which aids in further understanding of the formulation while also proving its correctness. The results shown here assume a pre-seeding capacity equivalent to 2 streams per node.

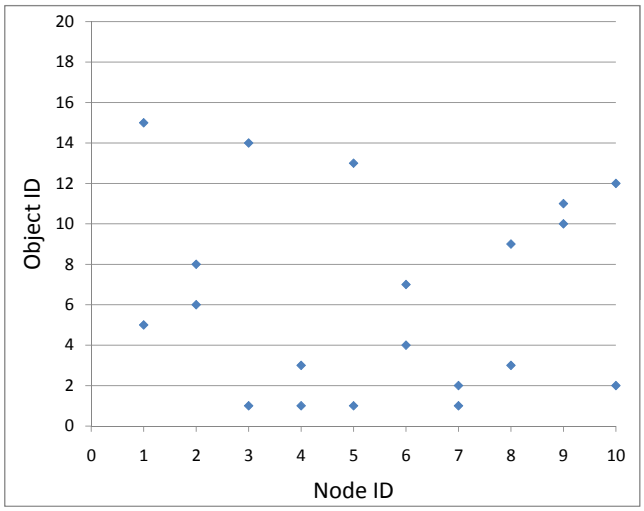
When the load on the system increases, that is, the number of active viewers in the community is high, all the STB's will be busy serving one another. There is very little free bandwidth in the network. Therefore to ensure that most requests are served from within the network, the most popular objects need to be replicated to a higher degree than those that are less popular. Since the probability of popular objects being requested is high, the high replication of such objects will ensure availability of the objects and thus most of the requests can be served by the P2P network. On the other hand, when the load on the system is low, the system will likely have excess bandwidth capacity even after serving the most popular objects, so it may use some of the STBs to replicate and pre-seed less popular objects as well.

Load is dependent on the product of arrival rate λ and service time t; the former will be higher as request rate increases, while the latter will be higher whenever the asymmetry between client uplink rate and video bitrate is high (as in modern ADSL and cable modem deployments).

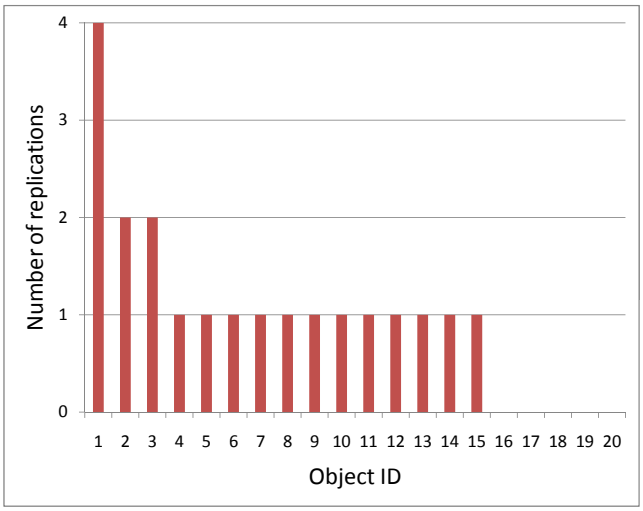
The difference between a low-load case and a high-load case is shown in Figure 5 ($\lambda t = 20$) and Figure 6 ($\lambda t = 100$). In these numerical examples, the number of caches/STBs was set to 10, each with a capacity to store 2 objects, and the total number of objects in the system was set to 20. The object popularity was distributed according to a Zipf distribution with $\alpha = 1$. The lower-numbered objects have higher popularity than the higher-numbered ones.

In Figure 5, λt is low, implying a low system load. Figure 5(a) shows a scatter plot of which objects are stored at which nodes (X_{ij}), while Figure 5(b) shows the number of replicated copies of each object. Objects 1–3 require multiple copies, with the maximum being 4 copies. Objects numbered all the way to 15 are stored at least somewhere in the peer-to-peer network. The number of replicas decreases with object number, since higher object numbers imply less popular objects. The scatter plot shows that more popular objects are either co-located with other objects that are far less popular (e.g., node 3) or that are nearly as popular but are replicated (e.g., nodes 4 or 7). The solver compensates for the fact that sequential search is more likely to target node 1 by storing less popular objects at node 1 so that it does not become overloaded. The objective function solution for this value of λt was 55.7%, indicating that 44.3% of all requests are satisfied by the peers.

In contrast, Figure 6 shows a high-load situation. The replication capacity is now focused on the most popular objects, with object 1 now getting 7 copies. Objects ranked 8 or lower are not stored at any peer. The large number of replicas of objects 1 and 2 means that their load contribution at any individual node is not particularly high; thus, the formulation does not need to use the low-numbered nodes only for storing less popular objects. Further, the formulation nearly always colocates objects 1 and 2. The objective function solution returned by the solver was also poorer at 79.5%, meaning that only 20.5% of the requests could be offloaded onto peers. Although the fraction of requests served by peers is just less than half

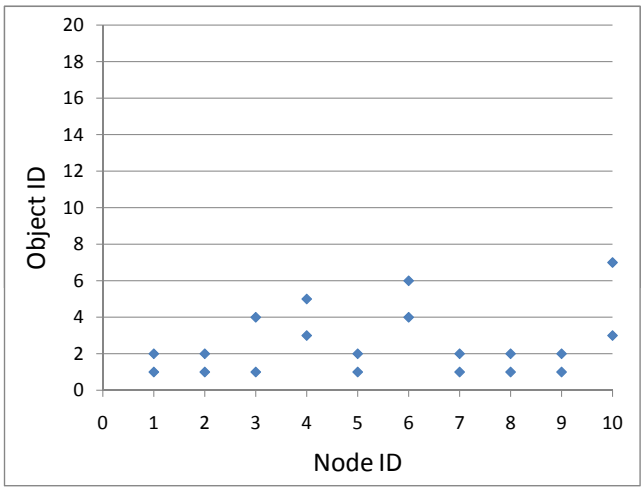


(a) Scatter plot of objects vs. cache

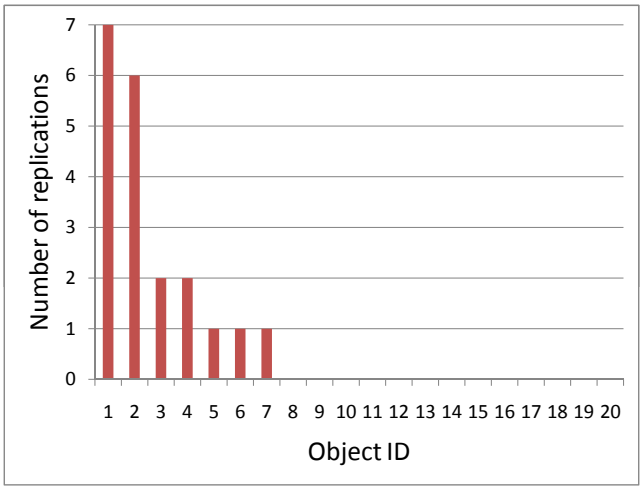


(b) Number of replicas of each object

Figure 5: Formulation output for low-load case



(a) Scatter plot of objects vs. cache



(b) Number of replicas of each object

Figure 6: Formulation output for high-load case

of the low-load case, the absolute load reduction is actually higher than in the low-load case since there is 5 times as much load. Thus, the optimization formulation effectively utilizes the P2P network to target both low-load and high-load situations.

4. EXPERIMENTAL EVALUATION

We use simulations to test the amount of server offload that is achieved by our formulation in a realistic IPTV environment. We developed a simulator for our scenario by extending the BitTorrent simulator of Bharambe et al. [5]. The original simulator simulates a BitTorrent swarm's data plane and the various BitTorrent mechanisms like choking/unchoking, rarest-first piece selection and tit-for-tat. The underlying network protocol is TCP/IP. The control packets of TCP/IP are not part of the simulation, however, as they add a very negligible overhead as compared to the size of data transferred. We also do not consider network latency as all the peers are located within the same community. Latency is very small when compared to the time required for data transfer and has no significant impact [11]. We removed the choking/unchoking mechanism as well as the tit-for-tat fairness enforcer from the simulator. We also modified the simulator to support multiple file swarms, in-order streaming piece delivery, and the use of an origin server to provide data when the peers cannot provide it within the real-time viewing constraint. Each client can serve at most one request at any given time but can download from multiple peers at the same time. We assume that all nodes can connect to each other and exchange presence bit-vectors after each transfer. In our simulations, each movie stream has a total size of 1 GB, which is the average size of a movie encoded in standard definition. The size of each data chunk is equivalent to 10 seconds of replay with a streaming rate of 2 Mbps (SD encoding). The upload rate per client is 1 Mbps and the download rate at 22 Mbps. The bandwidth model reflects the AT&T ISP network setup [23]. Each node can be preseeded with content equivalent to the size of two whole movie streams in addition to the streams the user may be viewing. This is consis-

tent with VoD service, as most VoD rentals are over a period of 24 hours and hence are saved onto the DVRs to enable faster replay, rewind, or forward functions when the video is viewed again within the rental period. We simulate a total of 40 clients and 120 movie streams. The movie streams are assigned popularity probabilities based on the Zipf probability distribution with $\alpha = 1$ [8]. The optimization problem is solved using BARON [33]. BARON (Branch-And-Reduce Optimization Navigator) is a solver for non-convex optimization problems. We ran the solver on a Sun Ultra 40 workstation equipped with two 3.0 GHz dual-core AMD Opteron processors and 12 GB RAM. Although though the system has 4 processor cores, the solver uses just one core for solving the problem. Given the non-linear nature of the formulation, a problem of the given size generates 112,800 non-linear entries and can take the solver several days (even weeks) to return the global optimal solution. This solve time is not feasible in our IPTV environment and hence we use the MINLP approximations that are estimated by the solver in a relatively quick time (several hours). In many instances the approximate solution may be the global optimal result. However, it may take the solver a very long time to confirm that such a solution is the global optimal. Also, users cannot conclude that the approximate solution is indeed the optimal solution until the solver run is complete.

Our experimental evaluation tests the pre-seeded VoD system by having the peers generate requests for the set of streams. The request sequences are generated randomly using a Zipf parameter of 1.0 for the base results (Section 4.4 considers the impact of other request popularity distributions.). The results reported compare our optimal formulation against two other seeding strategies: uniform random, by which any stream is equally likely to be pre-seeded at any node; and weighted-random, by which streams are randomly pre-seeded to nodes weighted by the popularity of the streams (e.g., if one object is twice as popular as another, it will be twice as likely to be seeded at each node). We also consider the results of the optimized pre-seeding based on whether or not data retrieval is allowed to upload to other clients while also serving pre-seeded data to the same client or not. For simplicity, the mathematical formulation assumes that uploading would not be allowed while also serving pre-seeded data to the same client. However, this constraint need not exist in the real system since serving local data does not actually consume uplink bandwidth. The case where a peer cannot upload while serving itself is called the "self-serve-no-upload" constraint. Henceforth we will refer to the results of peer-to-peer delivery using the optimization formulation and the optimization formulation with the "self-serve-no-upload" constraint as the optimal case and the optimal constrained case, respectively. In all the graphs we refer to the two cases as "Optimized" (unconstrained) and "Opt (self-serve)" (constrained) respectively. We also run the solver for two different time lengths – 2.5 hours and 4 hours and record both the solutions. The results for each is referred by appending the solve time to the terms mentioned above (eg: Optimized-4, Optimized-2.5 etc).

4.1 Full load results

In our first set of simulations, we assume all clients are viewing a movie and hence are actively requesting data from either their local cache, from their peers, or from the central server. All the clients are present in the network through the entire simulation run. There is no churn and no failure rate. We ran simulations for five different sets of requests and then average the result from the runs. Figure 7 shows the simulation results when all clients are concurrently viewing a movie. The graph shows the percentage reduction in server load due to the different pre-seeding strategies. The percentage reduction is calculated as the amount of data served by the P2P network when compared to a unicast distribution of content by the server. The x-axis represents the different allocation strategies while the y-axis represents the percentage reduction in server load. The optimized-4 case performs better than all the other storage strategies with a percentage reduction in server load at 50% while the optimized constrained case in the same time limit shows a percentage reduction of 48%. The difference between the optimal case and the constrained case is just 2%. This shows that the difference between allowing a peer to serve another peer or not to while also serving itself is not substantial, so this is not a significant limitation of the formulation. The server load reduction for the optimized-2.5 case is 49% and 46% for the constrained case, which are both nearly the same as optimized-4. The weighted-random allocation strategy lags behind all the optimized cases, with reduction in server load at 40.5%. Although weighted random does consider object popularity when assigning data, it does not explicitly consider the constraint on uplink banwdidth, so it is likely that certain links are overloaded because multiple popular objects were allocated to the same node. Finally, the uniform random case behaves the worst among all with only 15% reduction in server load. This behavior is expected, as this case does not consider object popularity at all.

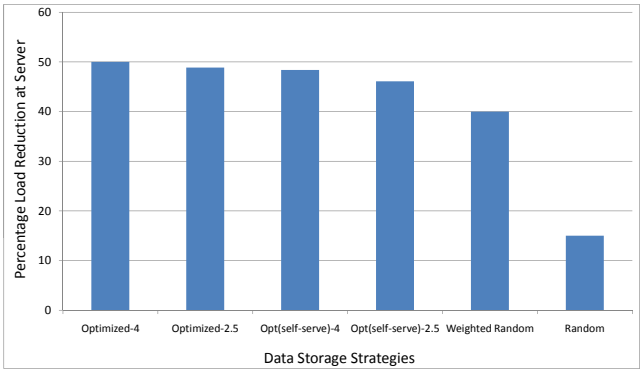


Figure 7: Percentage reduction in server load due to different pre-seeding strategies

4.2 Reduced duty-cycle results

We test the server load reduction of the different allocation schemes when the number of clients actively viewing movies is a percent of the total number of nodes. Figure 8 shows the results for five different pre-seeding algorithms (optimized-4, optimized-4 constrained, optimized-2.5, optimized-2.5 constrained and weighted-random) when the number of clients viewing movies is at 70%, 50%, and 30% of the total system. In each scenario, the other clients that are not watching any movie are still a part of the P2P network and serve any peer requests they may receive. The x-axis indicates the percentage of nodes that are watching a movie while the y-axis represents the percentage reduction in server load as before. In all three scenarios, the optimal and the optimal constrained case perform better than weighted-random. When 70% of the nodes are viewing a movie, the optimized-4 case reduces server load by 67%, the optimized-2.5 case reduces server load by 65% while the reduction due to weighted-random is 63%. Note that as we decrease the number of active viewers, the difference in the server load reduction between the optimal cases and the constrained cases reduces to zero. This arises because the constraint of not being able to serve another client while also serving oneself becomes

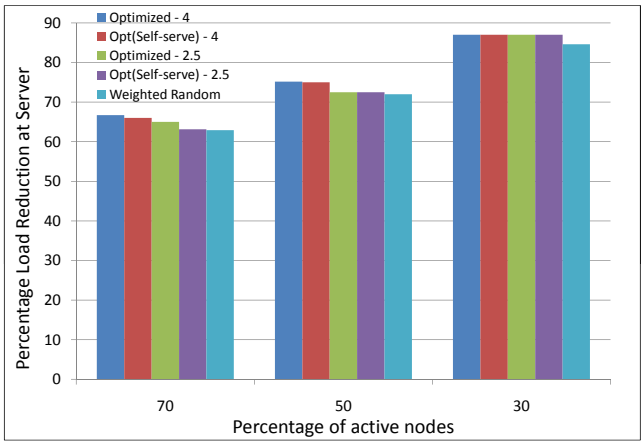


Figure 8: Percentage reduction in server load at different duty-cycles

less important when there are other free clients available. In all cases, our formulation outperforms the weighted-random heuristic.

4.3 Load-balancing

Load balancing is a key consideration for the optimal performance of any replication scheme, including this P2P network. Allowing even a few clients to be substantially more loaded than the others can noticeably degrade the performance of the network. Hence, we need to ensure that the objects are evenly distributed among the clients to ensure that majority of the requests do not go to just a few clients. As described in Section 3.2, the MINLP formulation ensures load-balancing among the clients in the network by explicitly considering the freeness at each node.

To check the load-balancing capability of the network, we record the amount of data served by each peer to other peers in the network. Figure 9 shows the average number of chunks served by each peer during all the five runs in the optimized-4 case. The y-axis represents the number of data chunks served by the different peers listed on the x-axis. We observe that most of the peers serve a near identical amount of data (around 90–110 chunks). The standard deviation of the load on the peers was 30 chunks. Node 36 served the least (62 chunks), while Node 5 served the most (174 chunks).

Figure 10 shows the number of chunks served by each node under the weighted-random pre-seeding strategy. Load balance is much poorer, with a standard deviation of 38 chunks delivered. Node 23 served the least amount of data at 31 chunks, while Node 16 served the most (186 chunks). Load-balance is poorer here because weighted-random only considers object popularity, not node freeness. Finally, Figure 11 shows the average number of chunks served by each peer during all the five runs for the optimized-2.5 case. We observe that load balancing in this case is not as good as the above two cases. The standard deviation of the load on the peers was 40 chunks. Node 3 served the least (67 chunks), while Node 29 served the most (189 chunks). Thus it does seem that running for the optimization formulation for a longer duration helps to reduce load imbalances by better spreading the data.

4.4 Robustness of the system

As mentioned in Section 2, external events can change the viewing patterns of users and substantially disturb the popularity pattern of videos. To test the robustness of our system against such externalities, we run two separate set of experiments simulating exter-

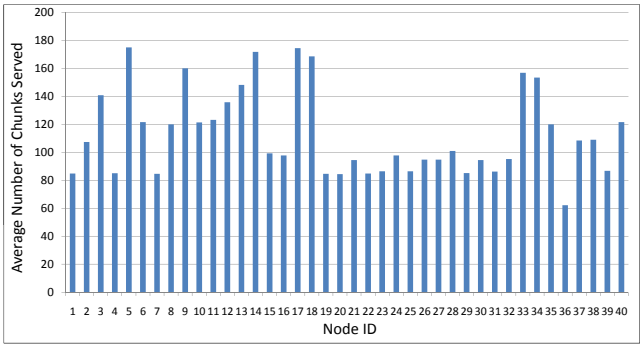


Figure 9: Distribution of load among peers for pre-seeding based on optimization solution at 4 hours

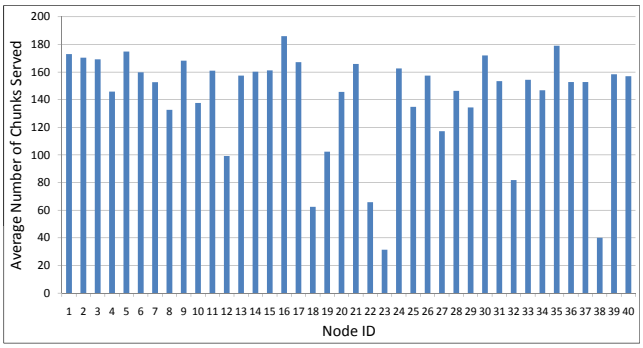


Figure 10: Distribution of load among peers for weighted-random pre-seeding

nal event occurrences against the different pre-seeding strategies and measure the reduction in server load. To ensure fairness when comparing the different Zipf values, we choose a smaller set of experimental parameters (25 nodes and 50 movie streams) so that we can run the solver until it reached a globally optimal result. The problem size generates 30,000 non-linear entries and returns the global optimal solution in 2.25 hours.

In the first set of experiments, we use the same pre-seeding strategy as in the previous tests (assuming a Zipf popularity model with $\alpha = 1$) but then test system performance using a request sequence that has a different Zipf α value, ranging from 0.75–1.5. Figure 12 shows the reduction in server load for request sequences for different Zipf α values for four cases: Actual Optimized (for which the optimization is performed using a popularity model with the same α value as the requests), Optimized, Constrained optimized, and Weighted random (all of which are generated using the pre-seeding resulting from a Zipf popularity model with $\alpha = 1$).

As we increase the α values (> 1.0), we see an improvement in the results. This can be explained by the fact that for higher α values, few objects have a high degree of popularity and hence most requests are for these few objects. Given the nature of the requests, the pieces for these popular objects are now more readily available in the network and performance is likely to improve. The results of the optimal pre-seeding generated for $\alpha = 1$ quite closely approach the results of the actual optimal pre-seeding, with less than 2% difference. In contrast, weighted-random lags more substantially since higher Zipf values lead to more severe object hot-spots and consequently encounter more load imbalance.

In contrast, when we decrease the α value to 0.75, we observe a degradation in performance as compared to our original runs. The

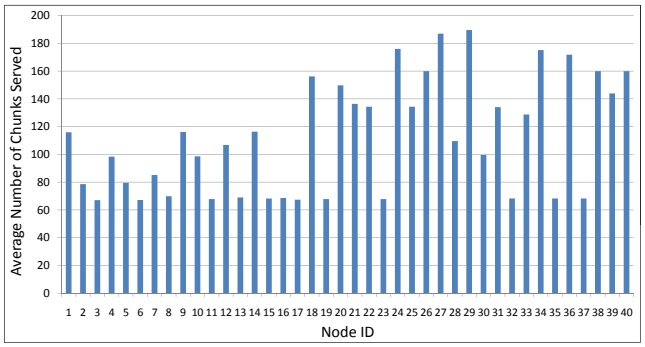


Figure 11: Distribution of load among peers for pre-seeding based on optimization solution at 2.5 hours

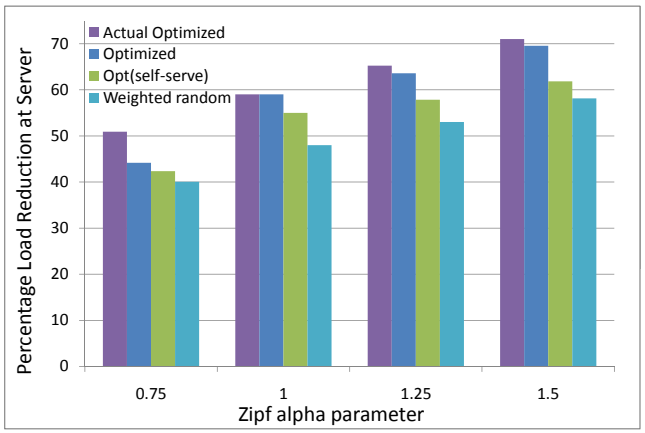


Figure 12: Percentage reduction in server load when Zipf parameter of request sequence (on X-axis) does not match Zipf parameter predicted for pre-seeding strategy (1.0)

reduction in server load in the optimal case was 44%. The difference between the server load reduction due to the optimal formulation and the weighted-random heuristic is much smaller(4%) when α is 0.75 as compared to the higher values of α as much of the data is now served by the central server in both cases. The difference in percentage reduction of load between the optimal case using the incorrectly-predicted Zipf parameter and the case where the actual Zipf parameter was used to pre-seed is 6% when α is 0.75: this is slightly higher than when using higher Zipf parameters, but still fairly low.

In the second set of experiments, we choose 10% of the objects uniformly at random and increase their popularity by 5x. The pre-seeding (both optimal and weighted-random) are generated using the originally predicted probabilities, but the actual test request sequences are generated using the new probabilities. We ran six different sets of simulations and the results were averaged over all the runs. The average reduction in server load for the optimal case was 48% while the reduction for the weighted-random case was 44%. These results show that the P2P network can still help in reducing the central server load in the face of externalities.

We repeated the second experiment with our original data size of 40 nodes and 120 objects and observed the robustness of the system. We choose 5% of the objects uniformly at random and increase their popularity by 5x. Like above, the pre-seeding (both optimal and weighted-random) are generated using the originally predicted probabilities, but the actual test request sequences are generated using the new probabilities. We ran five different sets of simulations and the results were averaged over all the runs. The average reduction in server load for the optimized-4 case was 48% while the reduction for the optimized-2.5 case and the weighted-random case were 47% and 42% respectively.

5. RELATED WORK

Video-on-Demand (VoD) has long been a research goal for system architecture, networking, and audio/video coding researchers, and hundreds of systems and solutions have been developed in these areas. A common way of implementing a VoD server is to use unicast and send each client a copy of the media, using one of several protocols designed for this purpose (e.g., RTP and RTSP [28, 29]). However, this unicast approach is inefficient with hundreds or thousands of clients. By taking advantage of the fact that the same files are requested by many of the clients, many techniques have been developed using multicast for nearly on-demand viewing, or using multiple unicast or multicast streams to reduce server load while still providing true on-demand service. Such schemes include patching, staggered broadcasting, hierarchical multicast stream merging, adaptive piggybacking, and periodic broadcast protocols [7, 15, 17, 18, 36]. However, IP multicast is rarely seen on the Internet or even intra-ISP networks, so these solutions have not had much impact.

Many newer techniques for video data delivery are based on some form of peer-to-peer or overlay multicast technology, all with their own protocols to manage peer communication and organization [14]. The first such system uses linear chains of clients to achieve functionality similar to IP multicast and uses these chains to implement a generalized batching technique for on-demand video [30]. GloVE combines these chaining and batching techniques, allowing multiple streams of data between different clients, but relies on IP multicast to make these streams efficient [16].

5.1 Peer-assisted VoD

CoopNet, PALS, PROP, Toast and Zebroid provide on-demand streaming using P2P networks and are more closely related to our work. Each of these systems seeks to support an infrastructure-based system with P2P networks and thus achieve scalability and reliability. CoopNet provides both live and on-demand streaming using a multicast tree rooted at the server and divides the streaming media content into multiple sub-streams using *multiple description coding (MDC)* to provide robustness [25]. When CoopNet is used for on-demand streaming, the P2P network is used only when the server is overloaded. The server is required to keep track of the peers and the content held by them, and redirect requests to peers when it is overloaded. (Clients in our system do this by themselves, preferring to get content from peers rather than the server). PALS uses layered-encoding to allow receivers to fetch data from multiple sources, including either other peers or servers [27]. The receiver chooses sender peers in such a way as to optimize throughput and quality of service. PALS distinguishes between senders and receivers and thus only uses peers as senders when they have the complete data of the stream. PROP is designed for intranets which deploy a proxy server [20]. At any time, the requesting client receives data from either a peer or the proxy server. If data is not available in the peers or the proxy, the proxy server requests the missing data from the media server.

As mentioned in the introduction, Toast uses BitTorrent to provide VoD to peers in a WAN [11]. To guarantee QoS, the P2P network is supported by a server. Toast explores various piece-picking policies and different seeding models tested at different client upload rates and constrained capacity. The system was eval-

uated with a single movie with no constraints placed on the piece-popularity, as this is most similar to the existing BitTorrent file swarm model [35]. However, using individual swarms for videos would mismanage and/or underutilize the potential resources available in a network of set-top boxes.

Zebroid pre-stripes content across peers in an ISP network to speed up P2P content delivery in an IPTV environment and is most related to our work [9]. Similar to our model, Zebroid uses a limited upload bandwidth model and stripes content during idle hours. Using simulations and through a prototype implementation, they show that Zebroid is effective in reducing load on the VoD server. Content data is divided into chunks and each chunk is further divided into stripes. Hence at any time, several peers are required to serve a single chunk of content. They use erasure codes to deal with peer failure, which is very low in a IPTV environment [9]. Zebroid does not perform any smart prefetching. Suh et al. propose a push-to-peer VoD system where data is proactively pushed from the content provider to peers [31]. They investigate data placement strategies and propose a distributed load balancing strategy for selecting peers. Deterministic and stochastic demand models are handled by a code-based scheme. They do not consider a server to support the P2P network [31].

BiToS is a pure P2P VoD service that uses BitTorrent for content distribution and balances rarest-first piece selection with a need for real-time delivery [37]. BiToS does not include any backing servers to guarantee real-time delivery. As a result, the measured results for all variants indicate that at least 5% of pieces are not received in time, degrading quality of service.

Other works have provided analysis and simulations of proposed peer-to-peer VoD systems backed by servers, in which the peer-to-peer network is used to reduce load from the servers. Cui et al. propose oStream, a system using overlay multicast trees to stream most of the video from peers instead of the server [14]. They offer extensive analysis and some simulation results. Huang et al. describe a peer-assisted VoD service with different fetching policies based on mathematical models of client needs and the capabilities of the server and P2P network [22]. They provide the results of a discrete-event simulation model based on real-world traces of accesses to a video to show the potential of such an approach. They focus on a single-video only and use the Internet as their distribution medium. Chen et al. describe and simulate a system that employs topology-aware algorithms and provides economic incentives to peers that provide data, thus providing additional encouragement to contribute resources and coordinating delivery to reduce overall network traffic [10].

Other P2P VoD systems have explored minimizing cross-AS and cross-ISP traffic by exploiting available P2P bandwidth and storage [24, 6, 21]. None of these consider intelligent pre-seeding.

5.2 Other Related Work

Numerous web caching systems and content distribution networks (CDNs) (e.g., Akamai) have sought to offload the delivery of streaming video content onto geographically-distributed servers. Peer-to-peer systems represent an alternative approach to this problem. P2P has advantages in dynamically provisioning resources as demand rises since each requesting node must also be a provider. If the peer nodes are actually controlled by a system administrator (e.g., cable company or PCs on a LAN), they can be set to remain as background seeds long after their viewing is complete, thus offloading work from remote network servers without requiring additional infrastructure support through a CDN.

A related problem to on-demand video streaming is that of live streaming. Live streaming accomplishes a similar goal (distributing video or other content to large number of people) but does not have the requirement that the user be able to skip to arbitrary points in the stream (for example, in the future). However, this is the primary advantage that BitTorrent exploits: the ability of clients to exchange pieces from anywhere in the file. BitTorrent-based systems for live streaming have been proposed, implemented, and analyzed. Such systems typically introduce a slight delay in playback to allow for limited exchange of a small window of content pieces that have already been published but not yet distributed to all peers[26][34]. Alternative peer-to-peer approaches include overlay-based multicast networks (e.g., [12]). These systems also provide the benefits of peer-to-peer in terms of demand-driven scalability, but BitTorrent-based systems benefit from BitTorrent's more richly-connected peering, use of multiple simultaneous peers at each peer, and tit-for-tat incentive system.

6. CONCLUSION

This paper presents and evaluates a system for intelligent pre-seeding of data in a ISP community network to reduce the load on the uplink between a cable ISP community and the regional hub for VoD services using peer-to-peer technologies. The system extends the ISP network by using consumer set-top boxes (STBs) to exploit any unused bandwidth and storage for streaming video.

A non-linear programming formulation is presented that, based on a popularity distribution of objects and the bandwidth and capacity constraints of the STBs, decides the number of copies of each object that need to be replicated in the P2P network to ensure optimal usage of the P2P network. We show that the expected object popularity distribution can be predicted by analyzing IPTV VoD subscription logs. The formulation also decides at which nodes in the network the replicated objects should be stored to optimize peer-to-peer data delivery opportunities.

We test the results of the formulation against a simulator for a modified BitTorrent peer-to-peer data delivery system. This framework supports streaming data delivery rather than just downloads by biasing the piece-picking policy toward immediately needed pieces, and also removes BitTorrent incentive mechanisms such as tit-for-tat and choking. The BitTorrent client is adapted to communicate with a traditional VoD server when the desired piece of data cannot be delivered by the peers in a timely fashion. Results from simulations show that using the formulation decision for pre-seeding can help in efficiently offloading server load. Our mathematical model generates results that are more efficient than even a popularity-aware weighted random preloading strategy, and the model results are also able to provide more efficient load-balancing among the peers. The reduction in server load due to our formulation was 50%, while that due to a popularity-aware weighted random was 40%. Our model for preseeding is also robust in the face of various popularity shifts studied. Thus our model combined with the modified BitTorrent client greatly improves the utilization of client resources and network bandwidth while also reducing server and network infrastructure requirements.

Acknowledgments

The authors thank Robin Chen (AT&T) for his help and comments on the paper. The authors also thank the anonymous reviewers for their input.

7. REFERENCES

[1] Amazon.com VoD service. www.amazon.com.

[2] CNET. http://asia.cnet.com/crave/2009/06/30/40-of-the-top-100-itunes-chart-positions-are-taken-by-michael-jackson.

[3] Hulu. www.hulu.com.

[4] YouTube. www.youtube.com.

[5] A. Bharambe, C. Herley, and V. Padmanabhan. Analyzing and improving a bittorrent network's performance mechanisms. In *Proceedings of IEEE Infocom*, 2006.

[6] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in BitTorrent via biased neighbor selection. *Proc. of IEEE ICDCSâ @ Y06*, 2006.

[7] S. Carter and D. Long. Improving video-on-demand server efficiency through stream tapping. In *Proc. International Conference on Computer Communications and Networks*, pages 200–207, 1997.

[8] Y. Chen, Y. Huang, R. Jana, H. Jiang, M. Rabinovich, J. Rahe, B. Wei, and Z. Xiao. Towards capacity and profit optimization of video-on-demand services in a peer-assisted IPTV platform. *Multimedia Systems*, 15(1):19–32, 2009.

[9] Y. Chen, R. Jana, D. Stern, B. Wei, M. Yang, and H. Sun. Zebroid: using IPTV data to support peer-assisted VoD content delivery. In *Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video*, pages 115–120. ACM New York, NY, USA, 2009.

[10] Y.-F. Chen et al. When is P2P Technology Beneficial for IPTV Services. In *Proceedings of the 17th International Workshop on Network and Operating System Support for Digital Audio and Video*, May 2007.

[11] Y. Choe, D. Schuff, J. Dyaberi, and V. Pai. Improving VoD server efficiency with bittorrent. In *Proceedings of the 15th international conference on Multimedia*, pages 117–126. ACM New York, NY, USA, 2007.

[12] Y. Chu, S. G. Rao, and H. Zhang. A case for end system multicast (keynote address). In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA, 2000. ACM Press.

[13] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, volume 6, 2003.

[14] Y. Cui, B. Li, and K. Nahrstedt. oStream: asynchronous streaming multicast in application-layer overlay networks. *IEEE Journal on Selected Areas in Communications*, 22(1):91 – 106, 2004.

[15] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *MULTIMEDIA '94: Proceedings of the second ACM international conference on Multimedia*, pages 15–23, New York, NY, USA, 1994. ACM Press.

[16] L. de Pinho, E. Ishikawa, and C. de Amorim. GloVE: A distributed environment for scalable video-on-demand systems. *Int. J. High Perform. Comput. Appl. (USA)*, 17(2):147 – 61, Summer 2003.

[17] D. Eager, M. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *Knowledge and Data Engineering, IEEE Transactions on*, 13(5):742–757, Sept.-Oct. 2001.

[18] L. Golubchik, J. C. S. Lui, and R. Muntz. Reducing i/o demand in video-on-demand storage servers. In *SIGMETRICS '95/PERFORMANCE '95: Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 25–36, New York, NY, USA, 1995. ACM Press.

[19] E. Grochowski and R. Halem. Technological impact of magnetic hard disk drives on storage systems. *IBM Systems Journal*, 42(2):338–346, 2003.

[20] L. Guo, S. Chen, S. Ren, X. Chen, and S. Jiang. PROP: a scalable and reliable P2P assisted proxy streaming system. In *Proceedings. 24th International Conference on Distributed Computing Systems, 2004.*, pages 778–786, 2004.

[21] C. Huang, J. Li, and K. Ross. Can internet video-on-demand be profitable? In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 133–144. ACM New York, NY, USA, 2007.

[22] C. Huang, J. Li, and K. Ross. Peer-Assisted VoD: Making Internet Video Distribution Cheap. *Proceedings of Sixth Internaltional Workshop on Peer-to-Peer Systems*, 2007.

[23] Y. Huang, Y. Chen, R. Jana, H. Jiang, M. Rabinovich, A. Reibman, B. Wei, and Z. Xiao. Capacity analysis of MediaGrid: a P2P IPTV platform for fiber to the node (FTTN) networks. *IEEE Journal on Selected Areas in Communications*, 25(1):131–139, 2007.

[24] V. Janardhan and H. Schulzrinne. Peer assisted VoD for set-top box based IP network. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, pages 335–339. ACM New York, NY, USA, 2007.

[25] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 177–186, 2002.

[26] http://www.ppstream.com/. Last checked July 21, 2009.

[27] R. Rejaie and A. Ortega. PALS: Peer-to-Peer Adapative Layered Streaming. In *Proceedings of the 13th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 153–161, June 2003.

[28] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF RFC 1889, January 1996.

[29] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). IETF RFC 2326, April 1998.

[30] S. Sheu, K. Hua, and W. Tavanapong. Chaining: A generalized batching technique for video-on-demand systems. In *Procedings of IEEE International Conference on Multimedia Computing and Systems*, pages 110–117, 1997.

[31] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello. Push-to-Peer Video-on-Demand system: design and evaluation. *IEEE Journal on Selected Areas in Communications*, 25(9):1706, 2007.

[32] M. Tawarmalani, K. Kannan, and P. De. Allocating Objects in a Network of Caches: Centralized and Decentralized Analyses. *Management Science*, 55(1):132–147, 2009.

[33] M. Tawarmalani and N. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563–591, 2004.

[34] S. Tewari and L. Kleinrock. Analytical Model for BitTorrent-based Live Video Streaming. In *Proceedings of the IEEE NIME 2007 Workshop*, January 2007.

[35] C. Thompson. The BitTorrent Effect. *Wired Magazine*, January 2005.

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

[37] A. Vlavianos, M. Iliofotou, and M. Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–6, April 2006.

APPENDIX

A. LINEARIZATION PROOF

We prove using induction that Equation (3) in Section 3.2 can be linearized to Equation (5).

Equation (3) is as follows:

$$Y_{ij} = X_{ij}F_j \prod_{j'<j}(1 - X_{ij'}F_{j'})$$

and Equation (5) says:

$$Y_{ij} = X_{ij}F_j(1 - \sum_{j'<j} Y_{ij'})$$

When $j = 1$, Equation (3) becomes:

$$Y_{i,1} = X_{i,1}F_1 \tag{A.1}$$

which is trivially equal to Equation (5). When $j = 2$, Equation (3) gives:

$$Y_{i,2} = X_{i,2}F_2(1 - X_{i,1}F_1)$$

Plugging in Equation (A.1) gives:

$$\begin{aligned} Y_{i,2} &= X_{i,2}F_2(1 - Y_{i,1}) \\ &= X_{i,2}F_2(1 - \sum_{j'<2} Y_{i,j'}) \end{aligned} \tag{A.2}$$

which is of the form specified in Equation 5.

Having proven the base case, assume that Equation (3) can be linearized to Equation (5) for $j = z$:

$$\begin{aligned} Y_{i,z} &= X_{i,z}F_z \prod_{j'<z}(1 - X_{ij'}F_{j'}) \\ &= X_{i,z}F_z(1 - \sum_{j'<z} Y_{i,j'}) \end{aligned}$$

This implies the following two properties:

$$\prod_{j'<z}(1 - X_{ij'}F_{j'}) = (1 - \sum_{j'<z} Y_{i,j'}) \tag{A.3}$$

$$X_{i,z}F_z = \frac{Y_{i,z}}{1 - \sum_{j'<z} Y_{i,j'}} \tag{A.4}$$

Using Equation (3) for $j = z + 1$ and plugging in the above equations, we find that:

$$\begin{aligned} Y_{i,z+1} &= X_{i,z+1}F_{z+1} \prod_{j'<z+1}(1 - X_{i,j'}F_{j'}) \\ &= X_{i,z+1}F_{z+1}(1 - X_{i,z}F_z) \prod_{j'<z}(1 - X_{i,j'}F_{j'}) \\ &= X_{i,z+1}F_{z+1}(1 - \frac{Y_{i,z}}{1 - \sum_{j'<z} Y_{i,j'}})(1 - \sum_{j'<z} Y_{i,j'}) \\ &= X_{i,z+1}F_{z+1}((1 - \sum_{j'<z} Y_{i,j'}) - Y_{i,z}) \\ &= X_{i,z+1}F_{z+1}(1 - \sum_{j'<z+1} Y_{i,j'}) \end{aligned} \tag{A.5}$$

This completes the proof by induction.