Push-to-Peer Video-on-Demand System: Design and Evaluation

Kyoungwon Suh, Christophe Diot, Jim Kurose, Laurent Massoulié, Christoph Neumann, Don Towsley, Matteo Varvello

***Abstract*—We propose Push-to-Peer, a peer-to-peer system to cooperatively stream video. The main departure from previous work is that content is proactively pushed to peers, and persistently stored before the actual peer-to-peer transfers. The initial content placement increases content availability and improves the use of peer uplink bandwidth.**

Our specific contributions are: (i) content placement and associated pull policies that allow the optimal use of uplink bandwidth; (ii) performance analysis of such policies in controlled environments such as DSL networks under ISP control; (iii) a distributed load balancing strategy for selection of serving peers.

***Index Terms*—Peer-to-Peer networks, Video on Demand service, push service, rateless coding, randomized peer selection**

I. Introduction

OVER the past five years, there has been considerable research in the use of peer-to-peer networks for distributing both live [6], [28], [21], [20] and stored [7], [2] video. In such systems, peer interest plays the central role in content transmission and storage - a peer *pulls* content only if the content is of interest. Once pulled content has been stored locally, the peer may then in turn distribute this content to yet other self-interested peers. Such a pull-based system design is natural when individual peers are autonomous and self-interested. However, when individual peers are under common control, for example in the case of residential home gateways or set-top boxes under the control of a network or content provider, a wider range of system designs becomes possible.

The use of home gateways or set-top boxes under the control of a network or content provider is motivated by the fact that the they meet much stronger reliability requirements in terms of system and bandwidth availability. In fact, many ISPs and content providers have already deployed or have plans to deploy such equipment (though they do not have peer-to-peer functionality) in subscribers' premises to offer billable content services.

Our objective is to design a reliable VOD architecture that relies on peer-to-peer transfer as a primary means to provide high-quality streaming. To the best of our knowledge, most pull-based video streaming services either provide only low-quality video (i.e., less than 500Kbps) or use hybrid approaches in which the role of P2P streaming is limited to reducing the traffic load posed on their streaming infrastructure (e.g., ZATTOO [27] and JOOST [12]). Though PPLIVE [20] has been successful in providing video streaming without much infrastructure support, it has been reported that it relies heavily on hosts in university networks [10].

In this paper, we investigate the design space of a *Push-to-Peer* Video-on-Demand (VoD) system. In such a system, video is first pushed (e.g., from a content creator) to a population of peers. This first step is performed under provider or content-owner control, and can be performed during times of low network utilization (e.g., early morning). Note that as a result of this push phase, a peer may store content that it itself has no interest in, unlike traditional pull-only peer-to-peer systems. Following the push phase, peers seeking specific content then pull content of interest from other peers, as in a traditional peer-to-peer system. The Push-to-Peer approach is well-suited to cooperative distribution of stored video among set-top boxes in a DSL network, where the set-top boxes themselves operate under provider control. We believe, however, that the Push-to-Peer approach is more generally applicable to cases in which peers are long-lived and willing to have content proactively pushed to them before video distribution among the cooperating peers begins.

In this paper, we consider the design and analysis of a Push-to-Peer system in a network of long-lived peers where upstream bandwidth and peer storage are the primary limiting resources. We consider a *controlled* environment, with a set of always-on peers, constant available bandwidth among the peers, and the possibility of centralized control, assumptions appropriate in the specific setting of a VoD system consisting of set-top boxes within a single DSLAM [11] in a DSL network. DSLAMs of providers such as France Telecom typically connect around 800 DSL users. With this number of users, if 50 GigaBytes of storage is available on each DSL gateway, it is possible to store up to 5600 DVD-quality movies under a single DSLAM. This number of movies would be scaled down by a suitable factor for the replicated placement schemes we advocate later.

We begin by describing an idealized policy for placing video data at the peers during the push phase - full striping - and its consequent pull policy for downloading video. We also consider the practical case in which the number of peers from which a peer can download is bounded, and propose a code-based scheme to handle this constraint. We demonstrate that these two placement policies are optimal among policies that

Manuscript received March 16; revised August 22, 2007. This work was supported in part by the National Science Foundation under CNS 0519998, ANI 0240487, and CNS 0626874. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation. This work was done while K. Suh was with UMASS, Amherst.

K. Suh is with Illinois State University, Normal, IL, US.

J. Kurose and D. Towsley are with University of Massachusetts, Amherst, MA, US.

C. Diot, L. Massoulié and C. Neumann are with Thomson Research Lab, Paris, France.

M. Varvello is with Eurecom, Sophia-Antipolis, France.

Digital Object Identifier 10.1109/JSAC.2007.071209.

make use of the same amount of storage per movie, in that they maximize the demand that the system can sustain.

We analyze the performance of these policies (in terms of blocking under a no-wait blocking model, and delay under a model in which blocked requests are queued until they can be served). Our performance models can be used not only to quantitatively analyze system performance but also to dimension systems so that a given level of user performance is realized - an important consideration if Push-to-Peer is provided as a billable service by the network provider. We also consider the case of prefix caching at the peers.

The remainder of this paper is structured as follows. In Section II, we describe the controlled DSLAM setting, and the push and pull phases in more detail. We also summarize some of the important differences between the Push-to-Peer and traditional peer-to-peer approaches for VoD. In Section III, we describe two policies for placing video data at the peers during the push phase. In Section IV we analyze the performance of the previous schemes under both a blocked-calls-lost and blocked-calls-queued model. We apply those analytical results to address prefix sizing problem. In Section V we propose a distributed job placement algorithm and investigate its performance. Section VI discusses related work. Section VII concludes this paper.

II. Network Setting and Push-to-Peer Operation

In this section, we describe the network setting for the Push-to-Peer architecture and overview the push and pull phases of operation. We also describe our video playback model, in terms of user requirements and performance metrics.

We will describe the Push-to-Peer architecture in the context of a number of always-on set-top boxes (STBs) or Residential Home Gateways (RHGs) that collectively sit below a DSLAM in a DSL network and cooperatively distribute video amongst themselves.

The Push-to-Peer system comprises a content server, a control server, and boxes at the user premises. The content server, located in the content provider's premises, pushes content to the boxes during the *push* phase, as described below. A control server is also located in the content provider's premise; it provides a directory service to boxes in addition to management and control functionalities. The always-on STBs or RHGs reside at the customer premises. Although there are important technological and commercial differences between STBs and RHGs, we will refer to these devices generically as *boxes* in the remainder of this paper, since the crucial capabilities - the ability to download, upload, and store video under provider control - are common to both STBs and RHGs.

Content distribution proceeds in two phases in our Push-to-Peer system.

- **Push Phase.** During the push phase, the content server pushes content to each of the boxes. We envision this occurring periodically, when bandwidth is plentiful (e.g., in the early AM hours), or in background, low priority mode. After pushing content to the peers, the content server then disconnects (i.e., does not provide additional content), until the next push phase. A crucial issue for the push phase is that of data placement: what portions of which videos should be placed on which boxes; we address this problem in Section 3.
- **Pull Phase.** In the pull phase, boxes respond to user commands to play content. Since a box typically does not have all of the needed content at the end of the push phase, it will need to retrieve missing content from its peers. While it is possible for the boxes to proactively push content among themselves (not in response to user commands) we do not consider that possibility here. We assume that a user will watch only one video at a time.

We make the following assumptions about the DSL network, and the boxes at the user premises:

- **Upstream and downstream bandwidth.** We assume that the upstream bandwidth from each box to the DSLAM is a constrained resource, most likely smaller than the video encoding/playback rate [1]. We assume that when a peer uploads video to N different peers, the upstream bandwidth is equally shared among those peers. We also assume that video is transferred reliably, either using FEC or ARQ. We assume that the downstream bandwidth is sufficiently large that it is never the bottleneck when a peer downloads video from other (possibly many other) peers (instead, the upstream bandwidths at those other peers are collectively the limiting resource). We thus also assume that the downstream bandwidth is larger than the video encoding/playback rate.
- **Peer storage.** We assume that boxes have hard-disks that can store content that is pushed to the box during the push phase. This content can then be uploaded to other peers upon request, during the pull phase. The disk may also store movie prefixes, that are used locally at the box to decrease startup delay, as discussed in Section 4. We note that when a box needs to pull video from other boxes for playout, this video must also be stored in a local playout buffer, but we do not consider the (relatively small) requirements of this playout buffer here.
- **Peer homogeneity.** We assume that all peers have the same upstream link bandwidth and the same amount of hard disk storage.

Each movie is chopped into *windows* of contiguous data of size W. A full window needs to be available to the user before it can be played back. However a user can play such a window once it is available, without waiting for subsequent data. The window size is a tunable parameter: the smaller the window size, the smaller the startup delay for video playback. Since the window is a unit of *random access* to a video, the window also allows us to support VCR operations such as jump forward and jump backward. A viewer only needs to wait until a single window to which a jump operation is made is fully available. Each window is further divided into smaller data blocks that are stored onto distinct boxes.

[1]For example, AT&T lightspeed network and Verizon FiOS allocate up to 1Mbps and 2Mbps upload bandwidth to each home respectively. The video encoding rate for high-definition (HD) video uses 6 Mbps bandwidth and the rate for standard-definition (SD) uses 2Mbps bandwidth [5]. Clearly, the aggregate upstream bandwidth of peers may be smaller than the aggregate downloading bandwidth needed to support high-quality p2p video streaming service. More importantly, the maximum upstream bandwidth could be reliably achieved only when the traffic is sent *locally* among the nodes connected by a same switch such as DSLAM [5].

In the sequel we will consider two modes of operation. In the first mode, when a new request to play a movie cannot be served at an aggregate speed that matches the encoding / playback speed, the request is dropped. In the second mode, the request is enqueued, until a sufficient amount of bandwidth to play back the video becomes available.

We refer to the first approach as the *blocking model*, and to the second as the *waiting model*. Depending on which model we consider, we measure system performance using either the request blocking rate or the mean startup delay.

III. Data placement and pull policies

In this section we first propose the full-striping data placement and code-based data placement schemes. In contrast to full striping, the latter allows a box to download a video from a small number of boxes. This is useful when the number of simultaneous connections that a box can support is constrained. We then state and prove optimality properties of both schemes, in terms of the demands they can accommodate. We consider both deterministic and stochastic models of demands.

VCR operations such as jump forward, jump backward, and pause can be supported by both schemes though we assume sequential access when those schemes and their corresponding demand models are presented. In architectural perspective, it is a simple modification [2]. Since the only resource constraint that our demand models pose is the downloading rate, it does not matter whether the window for a video is requested in order in terms of time (i.e., sequential access) or they are requested out of order (i.e., controlled random access required by VCR operations).

We don't consider the full striping to be a practical scheme, since it is not resilient to box failures. We present it to obtain the benchmark performance bound of the push-to-peer system, which is meant to be compared to the performance of the code-based scheme.

In the remainder of the paper we assume that there are M boxes and that each window of a video is of size W.

A. Full Striping scheme

A full striping scheme stripes each window of a movie over all M boxes. Specifically, every window is divided into M blocks, each of size W/M, and each block is pushed to only one box. Consequently, each box stores a *distinct* block of a window. A full window is reconstructed at a particular box by concurrently downloading $M-1$ distinct blocks for the window from the other $M-1$ boxes. Hence a single movie download request generates $M-1$ sub-requests, each targeted at a particular box.

A box serves admitted sub-requests according to the Processor Sharing (PS) policy, forwarding its blocks of the requested video to requesting boxes. PS is an adequate model of fair sharing between concurrent TCP connections, when there is no round-trip time bias and the bottleneck is indeed the upstream bandwidth.

We further impose a limit on the number of sub-requests that a box can serve simultaneously. Specifically, to be able to retrieve the video at a rate of R_{enc}, one should receive blocks from each of the $M-1$ target boxes at rate at least R_{enc}/M, where R_{enc} is the video encoding/playback rate. Hence we should limit the number of concurrent sub-requests being served by each box to at most $K_{max} := \lfloor B_{up}M/R_{enc} \rfloor$, where B_{up} is the upstream bandwidth of each box. We envision two approaches for handling new video download requests that are blocked because one of the $M-1$ required boxes is already serving K_{max} distinct sub-requests. In the first approach, we simply drop the new request. In the second approach, each of the $M-1$ sub-requests generated by the new request is managed independently at each target box. If there are fewer than K_{max} concurrent jobs at the target box, then the sub-request enters service directly. Otherwise, it is placed in a FIFO queue local to the serving box, and waits till it can start service.

B. Code-based placement

We describe a modification of full striping, namely code-based placement, under which the maximum number of simultaneous connections that a box can serve is bounded by y, for some $y < M-1$. This scheme applies rateless coding [16], [17]. A rateless code such as the LT code [16] can generate an infinite number of so-called coded symbols by combining the k source symbols of the original content. These k source symbols can be reconstructed with high probability from any set of $(1+\epsilon) * k$ distinct coded symbols. In practice, the overhead parameter ϵ falls in $[0.03, 0.05]$, depending on the code that we use [4], [17].

The code-based scheme we propose divides each window into k source symbols[3], and generates $Ck = (M(1+\epsilon)/(y+1))k$ coded symbols. We call C the expansion ratio, where $C > 1$. For each window, the Ck symbols are evenly distributed to all M boxes such that each box keeps $Ck/M = (1+\epsilon)k/(y+1)$ distinct symbols. A viewer can reconstruct a window of a movie by concurrently downloading any Cky/M distinct symbols from an arbitrary set of y boxes out of $(M-1)$ boxes.

The code-based scheme is similar to full striping in the sense that distinct (coded) symbols are striped to all M boxes. However, unlike full striping, only y boxes are needed to download the video.

We now define the pull strategy used for the code-based scheme. The maximum number, K'_{max}, of sub-requests that can be concurrently processed on each box to ensure delay-free playback now reads $K'_{max} = \lfloor (y+1)B_{up}/R_{enc} \rfloor$. Under the blocking model, a new request is dropped, unless there are y boxes currently handling less than K'_{max} sub-requests. In that case, the new request creates y sub-requests that directly enter service at the y boxes currently handling the smallest

[2]The following modification should be made: (1) Each box requires additional memory space of size equal to the size of a single movie. The additional memory caches a full copy of the video that a box is currently watching. (2) The window that a jump operation is made to by the box is downloaded if a copy of the window has not been cached in the additional memory space. Otherwise, the box continues to download all windows of the video in order until it has a full copy of the video in the additional memory.

[3]With rateless codes the greater the value of k, the greater is the probability of reconstructing content with small overhead [4]. Consequently, the symbol size should be as small as possible, and therefore in our case symbol size should be equal to packet size (i.e. MTU).

number of jobs. Under the waiting model, each box has a queue from which it selects sub-requests to serve. Each new movie download request generates $M-1$ sub-requests that are sent to all other boxes. Upon receipt at a receiving box, each sub-request either enters service directly, if there are less than K'_{max} sub-requests currently served by that box. Otherwise it is placed in a FIFO queue specific to the box. Once a total of y sub-requests have entered service, all other $M-1-y$ sub-requests are deleted. Thus each request eventually generates only y sub-requests.

C. Deterministic demands

We first consider a model where the demand is specified by the maximum number of concurrent viewings, N_j, of each movie j, that the system is expected to face at any given time. The quality of a placement strategy is then evaluated by determining the demand profiles it can handle. Here a demand profile is $\{N_j\}$ such that no additional request can be served. The demand profiles $\{N_j\}$ can be thought of as describing the maximum demand that can be handled at a busy hour, or during a flash crowd event.

We first consider full striping. One has the following.

Proposition 1: Under full striping, a sufficient condition for a demand profile $\{N_j\}_{j=1,\ldots,J}$ to be sustained is

$$\sum_{j=1}^{J} N_j R_{enc}/M \leq B_{up}. \tag{1}$$

Under any scheme which stores a single copy of each movie, a necessary condition for a demand profile $\{N_j\}_{j=1,\ldots,J}$ to be sustained is

$$\sum_{j=1}^{J} N_j R_{enc}(1-1/M) \leq MB_{up}. \tag{2}$$

Proof: To see the sufficiency of (1), we note that each particular viewing request is broken into $M-1$ sub-requests, mapped to $M-1$ boxes, and each such sub-request requires a rate of $(1/M)R_{enc}$ in order to allow delay-free playback. Thus the rate demand on a particular box is at most $\sum_{j=1}^{J} N_j R_{enc}/M$. It can therefore be met under Condition (1).

To establish the second part, let $A_{j,m}$ denote the amount of memory dedicated to movie j on box m. By assumption, $\sum_{m=1}^{M} A_{j,m} = T_j R_{enc}$. Let T_j denote the length of movie j in seconds. Consider a random assignment of movie requests to boxes, under the constraint that no two requests come from the same box. Then for a given box m, the rate at which it must handle sub-requests is, on average, given by

$$\sum_{j=1}^{J} \frac{A_{j,m}}{T_j} N_j (1-1/M).$$

Indeed, each request to view movie j has probability $(1-1/M)$ of coming from another box, in which case it creates a sub-request to box m, that must be served at rate $A_{j,m}/T_j$ to allow delay-free playback. Summing this expression over m yields the average total service rate for the system. This also coincides with the left-hand side of (2), which by assumption is strictly larger than the total uplink capacity MB_{up}. Thus there must exist specific assignments of viewing requests N_j to boxes m that are infeasible, for otherwise the average service rate would not exceed the total uplink bandwidth. ■

Note that the conditions (1) and (2) cover the cases where the ratio $\sum_j N_j R_{enc}/(MB_{up})$ is respectively less than one and greater than $M/(M-1)$. The intermediate range is of vanishing length when M is large. In this sense we can claim that full striping is asymptotically optimal among policies that store only one copy of each movie.

Let us now turn to code-based placement. Again, we assume that the amount of storage dedicated to each movie j is $CT_j R_{enc}$, for some common factor $C > 1$. We then have

Proposition 2: Under coding, a sufficient condition for the demand profile $\{N_j\}_{j=1,\ldots,J}$ to be sustained is

$$R_{enc}\left[\frac{C}{1+\epsilon} + \sum_{j=1}^{J} N_j \left(1 - \frac{C}{M(1+\epsilon)}\right)\right] \leq MB_{up}. \tag{3}$$

Under any scheme which stores at most $CT_j R_{enc}$ for movie j, a necessary condition for demand profile $\{N_j\}_{j=1,\ldots,J}$ to be sustained is

$$R_{enc} \sum_{j=1}^{J} N_j (1 - C/M) \leq MB_{up}. \tag{4}$$

Proof: Each request to view movie j generates y sub-requests, where y is related to C via $C = M(1+\epsilon)/(y+1)$. Delay-free playback is possible if each sub-request can be served at a rate of $R_{enc}/(y+1)$. A balanced assignment of sub-requests to boxes can ensure that each box deals with at most

$$\left\lceil \frac{1}{M} \sum_{j=1}^{J} N_j y \right\rceil \leq 1 + \frac{1}{M} \sum_{j=1}^{J} N_j y$$

sub-requests. Thus demand is feasible if

$$\frac{R_{enc}}{y+1}\left[1 + \frac{1}{M} \sum_{j=1}^{J} N_j y\right] \leq B_{up}.$$

Replacing y by its expression $(1+\epsilon)M/C - 1$ in the above inequality yields Condition (3).

To establish the second part, consider a random assignment of viewing requests to boxes. Let $A_{j,m}$ denote again the amount of storage dedicated to movie j on box m. Because of the constraint $\sum_m A_{j,m} = CT_j R_{enc}$, each request to view movie j has on average a fraction C/M of the required data stored locally on the box, and thus requires on average a service rate of $R_{enc}(1 - C/M)$. Thus, the left-hand side of (4) represents the average service rate required for delay-free playback of all requests. When (4) is not satisfied, this is larger than the total uplink bandwidth MB_{up}, and necessarily there exist specific assignments of viewing requests to boxes that cannot be satisfied, for otherwise the average service rate needed would not exceed the available uplink bandwidth. ■

D. Stochastic models of demand

Let us introduce the following stochastic model for demand. Requests for movie j arrive according to a Poisson process with rate ν_j. Each request originates from box m with probability $1/M$, for all $m \in \{1, \ldots, M\}$. This last assumption can be interpreted as follows. We assume that no knowledge is available regarding which user is more likely to request which movie. We make this assumption for simplicity. In practice we expect to have information about user preferences, either communicated explicitly by users to the system, or inferred from past usage behaviour.

We note that, although the Poisson arrival model is standard in queueing theory, it is not entirely realistic in the present set-up. In particular one would not expect the same movie to be viewed several times from the same box, or several movies to be viewed simultaneously from the same box. However, the model allows to gain insight in the design challenges of placement schemes. This is illustrated further in the next section where we discuss prefix caching strategies.

1) Optimality of full striping: Denote by L_j the size of movie j in bytes, and by $A_{j,m}$ the amount of memory in bytes dedicated to movie j on box m. Then the average size of a download request for movie j is $L_j - (1/M)\sum_{m=1}^{M} A_{j,m}$.

We shall assume that a single copy of each movie is stored in the system, which can be translated into the constraint $\sum_{m=1}^{M} A_{j,m} = L_j$. It is natural to ask whether under such constraints, there exists a placement strategy that is optimal with respect to the demand rates ν_j that it can accommodate. The following shows that full striping is such an optimal placement strategy:

Proposition 3: Assume that a single copy of each movie is stored in the whole system. Then under full striping data placement, and for the waiting model, the system is stable (i.e., download times do not increase unboundedly) whenever

$$\sum_{j=1}^{J} \nu_j L_j \,(1 - 1/M) < M * B_{up}. \tag{5}$$

Moreover, for any other placement strategy specified by the $A_{j,m}$, the set of rates ν_j accommodated without rejection is strictly smaller than that under full striping.

Proof: Note that for any placement policy in which movies are stored only once, the work arrival rate at a given box m is given by

$$\rho(m) := (1 - 1/M)\sum_{j=1}^{J} \nu_j A_{j,m}. \tag{6}$$

Under full striping, one has $A_{j,m} = L_j/M$. Thus condition (5) is equivalent to the condition that the work arrival rate $\rho(m)$ is less than the service rate B_{up} of box m. This condition does not depend on m, and is thus necessary and sufficient for stability of the whole system.

Consider now a different placement strategy, for which there exists a pair (j^*, m^*) such that $A_{j^*,m^*} > L_{j^*}/M$. For any demand rates ν_j, $j = 1, \ldots, J$, assume that there exists a pull strategy that can stabilize the system under such demand. Then necessarily, for all $m \in \{1, \ldots, M\}$, one has $\rho(m) < B_{up}$. Summing these inequalities one obtains (5), hence such demand can also be handled under full striping.

Consider now a particular demand vector where $\nu_j = 0$ for all $j \neq j^*$, and

$$\nu_{j^*}(1 - 1/M)L_{j^*} = MB_{up} - \epsilon,$$

for some small $\epsilon > 0$. Clearly this verifies (5). However, the load placed on box m^* is precisely

$$\rho(m^*) = (1 - 1/M)\nu_{j^*} A_{j^*,m^*}.$$

By our choice of (j^*, m^*), we thus have that

$$\rho(m^*) > (1 - 1/M)\nu_{j^*} L_{j^*}/M.$$

Thus for small enough ϵ, one must have $\rho(m^*) > B_{up}$. Therefore, this box is in overload and the system cannot cope with such demands, while full striping can. ■

As shown in [23], this result can be further extended to non Poisson arrivals, and strengthened by showing that at any given time, the average work to be done at a particular box is minimal under full striping.

2) Near-optimality of the code-based scheme: We assume additional storage is used per movie as described before. Specifically, we assume that a total storage capacity of $C * L_j$ is devoted to movie j, where $C = M * (1 + \epsilon)/(y + 1)$ is the expansion ratio introduced in the previous section. The solution based on encoding assumes that for movie j, a total quantity of $A_{j,m} \equiv C * L_j/M$ data is stored on each individual box m. This data consists of symbols, such that for any collection of $y + 1 = M/C$ boxes, each movie can be reconstructed from the joint collections of symbols from all these $y + 1$ boxes. We then have the following proposition:

Proposition 4: By using the pull strategy described in Section III-B, the system is stable whenever the Poisson arrival rates ν_j verify

$$\sum_{j=1}^{J} \nu_j L_j \,[1 - C/((1 + \epsilon)M)] < M * B_{up}. \tag{7}$$

Moreover, any scheme that uses $C * L_j$ storage for movie j cannot cope with demand rates ν_j, unless the following condition

$$\sum_{j=1}^{J} \nu_j L_j \,[1 - C/M] < M * B_{up} \tag{8}$$

holds.

The proof relies on standard Lyapunov function techniques, using as a Lyapunov function the unfinished work in the system. It is omitted in the present document for brevity. We only note that the average amount of data that needs to be downloaded for a request for movie j is $L_j(1 - C/M)$ when the overall storage devoted to movie j is CL_j, and hence the left-hand side of (8) is indeed the rate at which work enters the system, while the right-hand side is an upper bound on the service capacity of the system. Thus with the assumed total storage per movie, Condition (8) is indeed necessary to ensure the existence of a pull strategy for which the system is

stable. The code-based scheme is indeed nearly optimal, since Conditions (8) and (7) coincide when the overhead parameter ϵ tends to zero.

IV. Performance analysis

A. Blocking model

We now propose simple models to predict the blocking probability of the system when requests are dropped (blocking model) when resources are not available.

We first consider full striping. In the actual system, the number of requests in progress varies from box to box, because a requesting box does not place a request on itself. Also, the overall service speed varies between $(M-1)B_{up}$ and MB_{up} depending on the system state: when a single video download takes place, it proceeds at speed $(M-1)B_{up}$, while an overall service rate of MB_{up} is achieved when sub-requests are served on all boxes.

However we consider simplified dynamics, where the number of sub-requests is the same on each box, and the total service capacity is also constant. Specifically, we consider a total service capacity of $B_{total} = MB_{up}$ and assume this is shared evenly among active downloads. The total amount of data that needs to be downloaded for the playback of movie j is then taken to be $L_j(1-1/M)$. We assume movie j download requests arrive according to a Poisson process with rate ν_j, and a maximum number of concurrent downloads of $K_{max} = \lfloor B_{total}/[R_{enc}(1-1/M)] \rfloor$. These simplified dynamics correspond to the classical M/G/1/K/PS model, the blocking probability of which is given by (see e.g. [14])

$$P_b^I := \frac{(1-\rho)\rho^{K_{max}}}{[(1-\rho^{K_{max}+1})]} \tag{9}$$

where

$$\rho = \frac{\sum_{j=1}^{J} \nu_j L_j (1-1/M)}{B_{total}}. \tag{10}$$

To model the performance under coding we make similar simplifying assumptions. We again assume that each box handles the same number of sub-requests, so that the system state is captured by the total number of movie download requests. However we account for the fact that each movie request is served by a maximum of y boxes, by taking the total service rate, when there are n movie requests, as the minimum of B_{total} and $n\tilde{B}$ where $\tilde{B} := yB_{up}$.

Under such simplifying assumptions, the system state evolves as a birth and death process on $\{0, \ldots, K_{max}\}$, where $K_{max} = \lfloor B_{total}/[R_{enc}y/(y+1)] \rfloor$. The birth rate equals $\nu = \sum_j \nu_j$ in all states except K_{max}, and the death rate in state n is [4]

$$\frac{\min(n * \tilde{B}, B_{total})}{\overline{\sigma}}$$

where $\overline{\sigma}$ is the average job size,

[4]We would indeed have a Markovian birth and death process if job sizes were exponentially distributed, and with mean $\overline{\sigma}$. Insensitivity results on Processor Sharing systems, see e.g. [13] guarantee that the rejection probability is insensitive to the actual service time distribution and justify formula (11) for the case of mixtures of deterministic service time distributions.

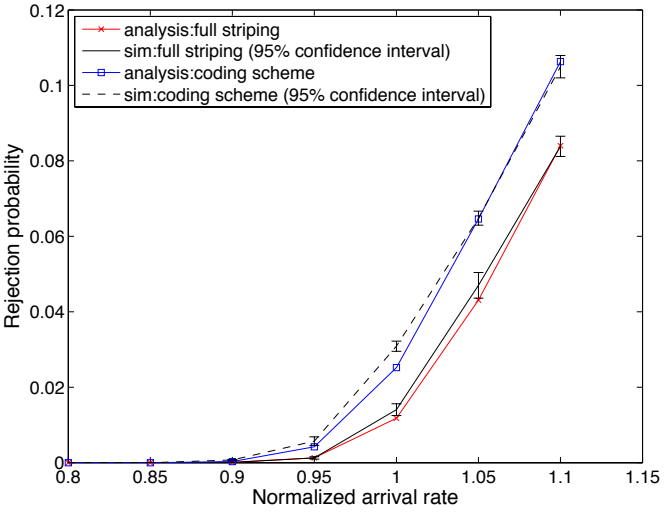

Fig. 1. Rejection probability with $M = 128$, Video encoding rate $R_{enc} = 2Mbps$, Upstream bandwidth $B_{up} = 1Mbps$, Size of Video $L = 2Gbytes$, Coding overhead $\epsilon = 0.05$, and Maximum number of simultaneous incoming connection $y = 31$.

$$\overline{\sigma} = (1+\epsilon)\sum_j (\nu_j/\nu) L_j (y/(y+1)).$$

For this system the blocking probability, which coincides with the steady state probability of being in state K_{max}, is

$$P_b^{II} := \frac{\rho_1^k/k!\ \rho_0^{K_{max}-k}}{\sum_{i=0}^{k-1} \frac{\rho_1^i}{i!} + \frac{\rho_1^k}{k!}\frac{1-\rho_0^{K_{max}-k+1}}{1-\rho_0}} \tag{11}$$

where we have introduced the notations

$$\begin{aligned} \rho_0 &:= \tfrac{\nu\overline{\sigma}}{B_{total}},\ \rho_1 := \rho_0 \tfrac{B_{total}}{B}, \\ k &:= \lfloor \tfrac{B_{total}}{B} \rfloor. \end{aligned}$$

The derivation is a simple exercise, and is omitted for brevity.

We plot the rejection rate for the proposed data placement schemes in Figure 1. In the simulation, we assume that the arrival of user requests is a Poisson process and that the probability that the request originates from a specific box is $1/M$. Note that the code-based scheme pushes 4 copies of the video to 128 boxes collectively. On the other hand, the full striping scheme pushes only 1 copy of the video. The x-axis indicates the normalized arrival rate of user requests and the y-axis indicates the rejection probability of user requests.

The rejection rates do not differ much between the two schemes. Perhaps surprisingly, the full striping scheme consistently outperforms the code-based scheme, even though the last scheme benefits from larger amounts of data stored on each box. This is explained by the fact that there is 5% coding overhead for the code-based scheme and the full striping scheme allows viewers to take advantage of the bandwidth from all 128 boxes regardless of the number of served viewers, while the code-based scheme constrains the number of boxes that are concurrently used to 31.

B. Full striping: waiting model

In this section we consider the performance of the system under full striping and when requests are allowed to queue up for resources. As for blocking, we make simplifying assumptions to define a tractable performance model. Specifically, we again assume that all boxes handle the same numbers of sub-requests. Thus, an incoming movie request is accepted on all boxes, in which case it gets a fair share of the overall system upstream bandwidth, provided there are fewer than K_{max} jobs in the system. Otherwise, the job is put in a single FIFO queue. Again K_{max} is determined to ensure that effective download rate is at least playback rate R_{enc}.

We call this system the FIFO+PS service system. While its performance is well understood under the assumptions of Poisson job arrivals and exponential service times, to our knowledge its performance has not been analysed previously when the assumption of exponential service times is relaxed. One of our contributions is to provide such an analysis, in a heavy traffic regime.

Notations are as follows. Service capacity is normalised to 1. Jobs arrive at instants of a Poisson process with intensity ν_ℓ. Jobs are i.i.d. with some fixed distribution; we denote by σ a typical job service time. K_{max} still denotes the maximum number of jobs that can be served concurrently. The index ℓ is introduced to set the stage for the heavy traffic analysis. Denoting by $\rho_\ell := \nu_\ell \mathbf{E}(\sigma)$ the traffic intensity, we shall assume that, as ℓ tends to infinity, the load approaches 1 from below:

$$\rho_\ell < 1,\ \ell \geq 1; \quad \lim_{\ell\to\infty} \rho_\ell = 1.$$

We shall further assume some scaling behaviour for parameter K_{max}, namely the existence of a positive number m such that:

$$\lim_{\ell\to\infty} (1-\rho_\ell) K_{max} = m.$$

We then have the following result, the proof of which can be found in [23]:

Theorem 1: Assume that the service time distribution is a finite mixture of Exponential distributions. Denote by Z^ℓ the number of jobs in steady state in the ℓ-th system. One then has the following convergence, for all $t > 0$:

$$\lim_{\ell\to\infty} \mathbf{P}\left(Z^\ell > \frac{t}{1-\rho_\ell}\right) = \begin{cases} e^{-m-2(t-m)\overline{\sigma}^2/\overline{\sigma^2}} \\ \text{if } t > m, \\ e^{-t} \text{ if } t \leq m. \end{cases} \tag{12}$$

Furthermore, denoting by W^ℓ the waiting time of a job in steady state in the ℓ-th system, one has the following convergence, for all $t \geq 0$:

$$\lim_{\ell\to\infty} \mathbf{P}\left((1-\rho_\ell) W^\ell > t\right) = e^{-m-2t\overline{\sigma}/\overline{\sigma^2}}. \tag{13}$$

In particular the probability of not waiting satisfies

$$\lim_{\ell\to\infty} \mathbf{P}(W^\ell = 0) = 1 - e^{-m}. \tag{14}$$

Remark 1: Although we have established the theorem only for the case of service times that are mixtures of exponential distributions, we expect it to hold more generally, and in particular to apply to the present setup where service time distributions are concentrated on a finite set of values.

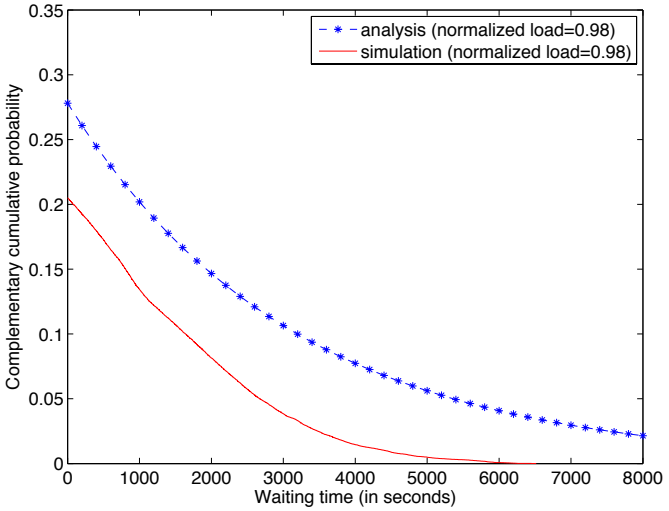

Fig. 2. Waiting time distribution for full striping with $M = 128$, Video encoding rate $R_{enc} = 2Mbps$, Upstream bandwidth $B_{up} = 1Mbps$, Size of Video $L = 2Gbytes$ and a normalized load of $\rho = 0.98$

We now indicate how to use this result. For given system parameters, we approximate the distribution of the waiting time of an arbitrary job as follows:

$$\mathbf{P}(W^\ell > t) \approx e^{-(1-\rho_\ell)[K_{max} + 2t\overline{\sigma}/\overline{\sigma^2}]}. \tag{15}$$

We plot the waiting time distribution for full striping, obtained by simulation, and the corresponding analytical prediction from Equation (15) in Figure 2. While the shapes are similar, the match is not perfect. We observed better matches when instead of deterministic service times, we used exponential service times in simulations. We suspect that the heavy traffic approximation becomes accurate only at very high loads when service time distributions are not exponential.

C. Application: sizing prefixes

We now show how to use the previous results to further optimize content placement assuming extra storage is available. We again assume there are J movies, all encoded at a constant bit rate R_{enc}, and denote by L_j the size of movie j.

For movie j, we assume that a prefix of size P_j is stored locally on each box. This ensures that each user can play back the first $t_j := P_j/R_{enc}$ seconds of movie j without downloading extra content. We further assume that encoded symbols are created and placed on each box so that for each movie j, its remainder can be reconstructed from the symbols present at any $y+1$ boxes.

Let D denote the memory space available on each box. The above described placement strategy will be feasible provided the following constraint is satisfied:

$$\sum_{j=1}^{J} P_j + (L_j - P_j)/(y+1) \leq D. \tag{16}$$

Denote by ν_j the rate of requests for movie j. The amount of movie j that needs to be downloaded for the playback is then

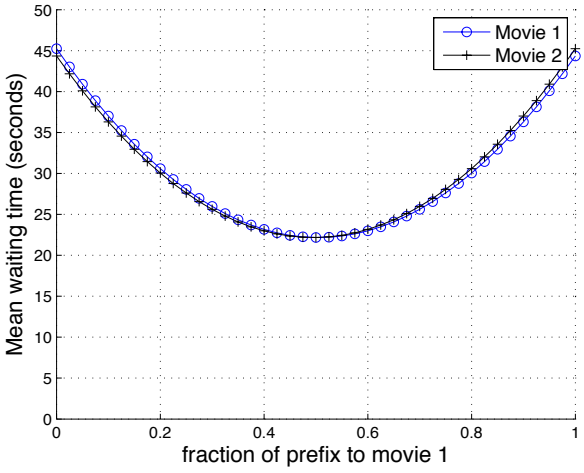

Fig. 3. Waiting time against prefixes balanced popularity ($\nu_1 = \nu_2 = 0.99$, $R_{enc} = B_{total} = 1$, $L_1 = L_2 = 1$, $P_1 + P_2 = 1$, $y \gg 1$)

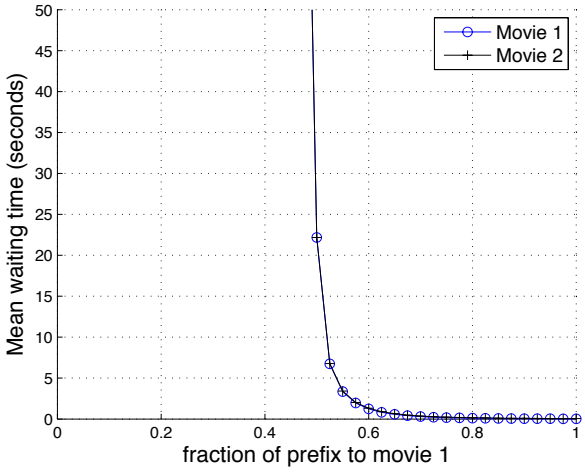

Fig. 4. Waiting time against prefixes distinct popularity ($\nu_1 = 0.99 * 4/3$, $\nu_2 = 0.99 * 2/3$, $R_{enc} = B_{total} = 1$, $L_1 = L_2 = 1$, $P_1 + P_2 = 1$, $y \gg 1$)

$$\sigma_j = \frac{y}{y+1}(L_j - P_j). \tag{17}$$

Indeed, the prefix of size P_j is stored locally, as well as a fraction $1/(y+1)$ of the remainder of the movie. The normalised load on the system is thus:

$$\rho = \frac{\sum_{j=1}^{J} \nu_j \sigma_j}{B_{total}}. \tag{18}$$

1) Blocking probabilities: We first consider performance under blocking. The maximum number of concurrent jobs is $K_{max} = \lfloor B_{total}/[R_{enc}y/(y+1)] \rfloor$. The blocking probability is given by (9) in the particular case where $y+1 = M$, that is to say under full striping. This probability is then minimized by making the load as small as possible.

Using linear programming, one can easily see that, to minimize the load ρ as given by (18) and (17) under memory constraints (16) one should aim to cache locally the most popular movies in full.

2) Waiting times: We now assume that requests are queued and scheduled according to FIFO rather than dropped when the number of concurrent requests in service equals K_{max}.

The evaluations (15) give us an approximation of the distribution of the delay W between request initiation and download beginning. The actual delay can be reduced because playback can start t_j seconds before download starts.

This yields the following expression for the average delay $\bar{D}_j$ experienced by requests for movie j:

$$\begin{aligned} \bar{D}_j &= E\left[\max(0, W - t_j)\right] \\ &= \int_{t_j}^{\infty} (x - t_j) \frac{2\overline{\sigma}(1-\rho)}{\overline{\sigma^2}} e^{-(1-\rho)[K_{max} + 2x\overline{\sigma}/\overline{\sigma^2}]} dx. \end{aligned}$$

We thus obtain the formula

$$\bar{D}_j = \frac{\overline{\sigma^2}}{2\overline{\sigma}(1-\rho)} e^{-(1-\rho)[K_{max} + 2t_j\overline{\sigma}/\overline{\sigma^2}]}. \tag{19}$$

We use a simple example to illustrate how a fixed amount of memory in a box can be optimally allocated to preload prefixes of movies depending on their relative popularities. Figures 3 and 4 show plots of the mean waiting times $\overline{D}_j$ obtained from Formula (19). In each case, there are two movies, and there is a fixed amount of memory that can be used for prefixes of either or both movies. In Figure 3, the popularities of both movies are same. In this case, the figure indicates that both movies should get prefixes of equal sizes. Note that for equal popularities, varying prefixes does not change the normalized load ρ. Also, it does not affect the average service time $\bar{\sigma}$. It would appear then that one movie would benefit from having a larger prefix. This is however not the case, because unbalanced prefixes lead to a large variance in the service times and thus a large second moment $\overline{\sigma^2}$.

In Figure 4, movie 1 is twice as popular as movie 2. The figure indicates that it is beneficial to both movies to allocate the prefix memory to movie 1. By storing large prefixes for movie 1, we reduce the system load ρ, and this is the leading effect.

V. Randomized Job Placement

In the previous sections we considered the case where all boxes are centrally coordinated. With such an assumption the job placement strategies, i.e. the decision where to place and serve the sub-requests of a job, are optimal. However, in practice a centralized system does not scale in the number of boxes. In this section we therefore propose a distributed load balancing strategy for the selection of serving peers. Although we only consider the case that upstream bandwidth is not variable here for the interest of space, we also consider the case that upstream bandwidth is dynamically changing in our technical report [23]. More specifically, we propose resource overbooking scheme to hedge against upstream bandwidth diversity and dynamic job migration scheme to address upstream bandwidth fluctuations. Details can be found in [23].

The strategy we consider for initial job placement is as follows. When a download request is generated, d distinct boxes are randomly chosen from the overall collection of M boxes. The load, measured in terms of fair bandwidth share that a new job would get, is measured on all probed boxes. Finally, sub-requests are placed on the y least loaded boxes

among the d probed boxes, provided that each of the y sub-requests gets a sufficiently large fair bandwidth share, i.e. larger than or equal to $(y/(y+1))R_{enc}$ with our previous notation. If any of the least loaded boxes cannot guarantee such a fair share, then the request is dropped.

We assume as before that each box has a fixed overall upstream bandwidth of B_{up}. Thus the maximum number of sub-requests on each box is $K_{max} = \lfloor B_{up}/[y/(y+1)R_{enc}] \rfloor$.

Many results are available on the performance of related randomized load balancing schemes. If we assume requests arrive according to a Poisson process with rate $\lambda * M/y$, no rejection ($K_{max} = \infty$), $y = 1$ (requests generate a single sub-job), and exponential job size distribution, we have exactly the model analyzed by Vvedenskaya et al. [26] (see also Eager et al. [8] and Mitzenmacher et al. [19]). For this system they show that, in the large M limit, in steady state the fraction ϕ_i of all M boxes that contain at least i jobs is given by

$$\phi_i = \rho^{\frac{d^i-1}{d-1}},$$

where ρ is the normalized load on each box.

The system we consider differs by the fact that there are several sub-jobs, and by the possibility of job rejection. It is however amenable to a similar analysis. We now determine fixed point equations that characterize the fraction of boxes holding a given number of sub-jobs in equilibrium. We do not claim the derivation is rigorous, but instead validate it by simulations.

The heuristic derivation proceeds as follows. Fix $i \in \{0, \ldots, K_{max}\}$. For a new request, denote by $X_{<i}$ (respectively X_i, $X_{>i\&<K_{max}}$ and $X_{K_{max}}$) the number of sampled boxes with less than i jobs (respectively i, more than i and less than K_{max}, and K_{max}). The vector of these four quantities follows a multinomial distribution with parameters d and $(p_{<i}, p_i, p_{>i\&<K_{max}}, p_{K_{max}})$, where

$$p_{<i} := \sum_{j<i} p_j,\ p_{>i\&<K_{max}} := \sum_{i<j<K_{max}} p_j.$$

Denote by $F_i(u, v, w, z)$ the probability that this multinomial distribution puts a job on the vector (u, v, w, z). Its dependence on the parameters p_j is not made explicit to simplify notation. Denote by G_i the expected number of boxes which previously had i jobs and receive a new one from such a new request. This can be written as

$$G_i = \sum_{u,v,w,z} F_i(u, v, w, z) \min(v, \max(0, y-u))\ \mathbf{1}_{z \le d-y}.$$

Indeed the factor $\mathbf{1}_{z<d-y}$ retains only terms in the summation where all sub-jobs can get enough bandwidth, and the term $\min(v, \max(0, y-u))$ counts the number of least loaded boxes that currently have i jobs. We obtain the following heuristic differential equation using the notations:

$$\frac{d}{dt} M p_i = M(\lambda/y)(G_{i-1} - G_i) - \mu M(p_i - p_{i+1}).$$

The rationale is that new boxes with i jobs appear at rate $M\lambda/yG_{i-1}$ because of extra jobs being placed on boxes previously holding $i-1$ jobs, and also at rate $\mu M p_{i+1}$ because of departures from boxes previously holding $i+1$ jobs. The rationale for the departure rates is similar.

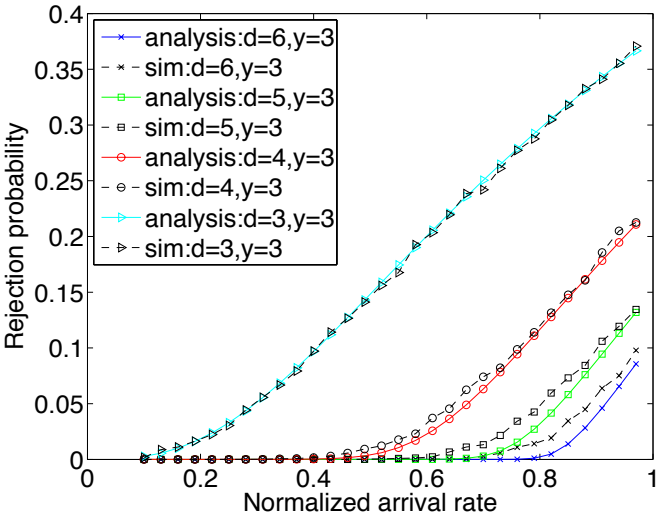

Fig. 5. Numerical solutions and simulation results for rejection probability using the proposed load balancing scheme

The fixed point equation for p_i is then obtained by setting the left-hand side of the previous equation to zero. Introduce now the notation

$$\lambda_i := \frac{\lambda}{y} \frac{G_i}{p_i}.$$

The fixed point equations may then be written as

$$p_{i+1}\mu = \lambda_i p_i,\ i = 0, \ldots, K_{max} - 1.$$

Since $\sum_{i=0}^{K_{max}} p_i = 1$, we obtain in turn

$$p_0 = \frac{1}{1 + \sum_{i=1}^{K_{max}} [(\prod_{j=0}^{i-1} \lambda_j)/\mu^i]} \quad (20)$$

$$p_n = \frac{\prod_{j=0}^{n-1} \lambda_j}{\mu^{n-1}} p_0,\ n = 0, \ldots, K_{max}. \quad (21)$$

Note that the parameters λ_i in the right-hand sides of these expressions depend on the p_i's themselves. The fixed point equations (20), and (21) cannot be solved explicitly. However we obtain a numerical approximation by applying iteratively the function specified by (20), and (21) on an initial guess for the p_i's. We observed fast numerical convergence of the iterations in our experiments. Once the parameters p_j are determined, the rejection probability is determined according to the formula

$$p_{reject} = \sum_{i=d-y+1}^{d} \binom{d}{i} p_{K_{max}}^i (1 - p_{K_{max}})^{d-i}.$$

Figure 5 shows the numerical solutions and simulation results we obtain for distinct choices of parameters (y, d) for varying normalized load $\rho = \lambda/\mu$, and setting K_{max} to 3. Here, the simulation results is obtained using $M = 50$ boxes. The numerical solutions match reasonably well the simulation results. We believe that the fixed point equations we just described are accurate in the large M limit.

More importantly, we observe that even at normalised loads close to 100%, the rejection probabilities remain small: below

15% when only two additional boxes are probed and down to 10% when three additional boxes are probed.

VI. Related work

Peer-to-peer networks for streaming video on the Internet have generated a lot of interest recently [6], [28], [20], [25]. However, most of the efforts have focused on efficient tree and mesh construction, assuming the upstream bandwidths of peers are larger than video playback rate. Under this assumption p2p systems can scale to support arbitrarily large numbers of clients. In contrast, we can cope with uplink bandwidths smaller than video playback rate, a condition that holds in most access networks, particularly DSL. More recently, Dana et al. [7] and Tewari et al. [24] proposed BitTorrent-based live streaming service under the same assumption of limited upstream bandwidth. In both proposals, the upstream bandwidth limitation is overcome by the assistance of server-based stream delivery in their proposed systems. However, the Push-to-Peer system does not rely on content servers except in the push phase.

Load balancing strategies have also been investigated in the context of job scheduling in distributed systems and more general bins and balls problems [8], [18], [19]. To the best of our knowledge, all of the proposed load balancing schemes are targeted to balance loads of independent jobs. On the contrary, we address the problem of balancing the load imposed by sub-requests from a job, that should be co-scheduled ideally. More recently, load balancing in the case of bulk arrivals of jobs has been investigated by Adler et al. [1], however, the balancing decision is made per job rather than per set of jobs arriving together. Our proposed scheme collectively balances all sub-requests for a job.

Another related area of work is the data placement and pull scheme for video streaming services. Several methods have been proposed in the literature [15], [3], [21], [22]. Particularly, random duplicated assignment strategy of data blocks and mirroring are proposed for VoD servers by Korst [15] and Bolosky et al. [3] respectively to address the problem of disk failure. However, we use a code-based placement that addresses the problem of box failures. The prefix prefetching schemes for p2p video streaming [21], [22] require upstream bandwidth of a peer to be larger than video playback rate, an assumption we do not make.

Rateless coding schemes have been proposed by [4], [17], [16]. While these works discuss how to use the codes to download files using multicast/broadcast transmissions [4] or using peer-to-peer networks [17], none of these works address the usage of coding for video streaming or video-on-demand. Other work proposed the use of network coding to accelerate file download in peer-to-peer networks [9] or to ameliorate VoD for p2p [2]. Because of the push-phase, our approach does not require that peers serve content that they downloaded previously from other peers. Therefore network coding is not needed in our context.

Our work is different from a streaming service provided by multiple servers in the following aspects. First, unlike multiple streaming servers, the boxes are also clients in our case. The implication is that by placing more data, the bandwidth requirements become lower, which is not the case in the multi-server streaming. Secondly, in case of a multi-server streaming service, a client is redirected to a set of streaming servers, which are under complete control of the provider and well-connected through high-speed networks. However, in our work, the client chooses a set of best peer nodes in distributed fashion, i.e., using the proposed randomized peer selection algorithm.

VII. Conclusion and future work

We proposed Push-to-Peer, a novel peer-to-peer approach to cooperatively stream video using push and on-demand pull of video contents. We showed the theoretical upper performance bounds that are achieved if all resources of all peers are perfectly pooled, and present the placement (namely full-striping and code-based scheme) and pull policies that achieve those bounds. However, perfect pooling is only possible with global knowledge of system state, which in practice is not feasible. Therefore, we proposed and analysed a randomized job placement algorithm. We are currently developing a prototype system.

We plan to do a more systematic analysis of placement schemes that take into account movie popularity. The non-uniform size of prefixes preloaded for different movies makes the use of processor sharing scheduling less effective, because the deadline for downloading a window is determined by the size of preloaded prefix. To address this issue, we plan to adopt Earliest Deadline First (EDF) scheduling policies developed for multiprocessors.

VIII. Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments.

References

[1] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen. Parallel randomized load balancing. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 238–247, 1995.

[2] S. Annapureddy, C. Gkantsidis, and P. Rodriguez. Providing Video-on-Demand using Peer-to-Peer networks. In *Proc. Internet Protocol TeleVision (IPTV) workshop*, 2006.

[3] W. Bolosky, R. Fitzgerald, and J. Douceur. Distributed schedule management in the Tiger video fileserver. In *Proc. ACM SOSP*, 1997.

[4] J. W. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE J-SAC, Special Issue on Network Support for Multicast Communication*, 20(8), 2002.

[5] Y. Chen, Y. Huang, R. Jana, H. Jiang, M. Rabinovich, B. Wei, and Z. Xiao. When is P2P technology beneficial for IPTV services? In *Proc. of ACM NOSSDAV*, 2007.

[6] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *Proc. ACM SIGCOMM*, 2001.

[7] C. Dana, D. Li, D. Harrison, and C. Chuah. BASS: BitTorrent assisted streaming system for video-on-demand. In *Proc. IEEE MMSP*, 2005.

[8] D. Eager, E. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Trans. software engineering*, SE-12(5):662–675, 1986.

[9] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proc. IEEE INFOCOM*, 2005.

[10] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Trans. Multimedia*, 2007.

[11] International engineering consortium. Digital subscriber line access multiplexer (DSLAM): Definition and overview. *http://www.iec.org/online/tutorials/dslam*.

[12] http://www.joost.com.

[13] F. Kelly. *Reversibility and stochastic networks*. Wiley, New York, 1979.
[14] L. Kleinrock. *Queueing systems Volume 1:Theory*. Wiley Interscience, 1975.
[15] J. Korst. Random duplicated assignment: An alternative striping in video servers. In *Proc. ACM Multimedia*, 1997.
[16] M. Luby. LT Codes. In *Proc. IEEE Symposium on foundation of computer science (FOCS)*. IEEE Computer Society, 2002.
[17] P. Maymounkov and D. Mazieres. Rateless codes and big downloads. In *Proc. the International Workshop on Peer-to-Peer Systems*, February 2003.
[18] R. Mirchandaney, D. Towsley, and J. Stankovic. Adaptive load sharing in heterogeneous distributed systems. *Journal of parallel and distributed computing*, 9:331–346, 1990.
[19] M. Mitzenmacher, A. Richa, and R. Sitaraman. The power of two random choices: A survey of the techniques and results. *Handbook of Randomized Computing*, 1:255–305, July 2001.
[20] http://www.pplive.com.
[21] A. Sharma, A. Bestavros, and I.Matta. dPAM: a distributed prefetching protocol for scalable asynchronous multicast in P2P systems. In *Proc. IEEE INFOCOM*, 2005.
[22] S.Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *Proc. IEEE INFOCOM*, 1999.
[23] K. Suh, C. Diot, J. Kurose, L. Massoulié, C. Neumann, D. Towsley, and M. Varvello. Push-to-peer video-on-demand system: design and evaluation. Technical Report CR-PRL-2006-11-0001, Thomson Research, November 2006.
[24] S. Tewari and L. Kleinrock. Analytical model for BitTorrent-based live video streaming. In *Proc. IEEE NIME Workshop*, 2007.
[25] A. Vlavianos, M. Iliofotou, and M. Faloutsos. BiToS: Enhancing BitTorrent for supporting streaming applications. In *Proc. IEEE Global Internet Symposium*, 2006.
[26] N. Vvedenskaya, R. Dobrushin, and F. Karpelevich. Queueing system with selection of the shortest of two queues: an asymptotic approach. *Problems of information transmission*, 1996.
[27] http://www.zattoo.com.
[28] X. Zhang, J. Liu, B. Li, and T. Yum. CoolStreaming/DONet: A data-driven overlay network for live media streaming. In *Proc. IEEE INFOCOM*, 2005.

Kyoungwon Suh received B.S. and M.S. degrees in Computer Engineering from Seoul National University in Korea in 1991 and 1993, respectively. He continued his studies in the Department of Computer Science at Rutgers University in NJ, where he earned a M.S. degree in 2000. In 2007, he received his Ph.D. degree in Computer Science from University of Massachusetts at Amherst. He is currently an assistant professor in Illinois State University, Normal, IL. His research interests include peer-to-peer networks, overlay networks, network monitoring and measurement, and video streaming services. He is a member of ACM.

Jim Kurose received a B.A. in physics from Wesleyan University and a Ph.D. degree in computer science from Columbia University. He is currently Distinguished University Professor (and past chairman) in the Department of Computer Science at the University of Massachusetts. He has been a Visiting Scientist at IBM Research, INRIA, Institut EURECOM, the University of Paris, LIP6, and Thomson Research Labs. His research interests include network protocols and architecture, network measurement, sensor networks, multimedia communication, and modeling and performance evaluation. He has served as Editor-in-Chief of the IEEE Transactions on Communications and was the founding Editor-in-Chief of the IEEE/ACM Transactions on Networking. He has been active in the program committees for IEEE Infocom, ACM SIGCOMM, and ACM SIGMETRICS conferences for a number of years, and has served as Technical Program Co-Chair for these conferences. He is a Fellow of the IEEE and ACM.

Christophe Diot received a Ph.D. degree in Computer Science from INP Grenoble in 1991. He was with INRIA Sophia-Antipolis (Oct. 93-Sep.98), Sprint (Oct. 98-Apr. 03), and Intel Research in Cambridge, UK (May 03-Sep. 05). He joined Thomson in October 2005 to start and manage the Paris Research Lab (http://thlab.net). Diot has been a pioneer in multicast communication, DiffServ, Internet measurements, and more recently Pocket Switched Networks. Diot's research activities now focus on advanced p2p communication services. Diot is the Thomson Scientific Council chair and an ACM fellow.

Laurent Massoulié is a researcher at the Thomson research Lab in Paris. His recent work focuses on design and evaluation of peer-to-peer systems for Video-on-Demand and Live Streaming applications. He also works on distributed schemes for congestion control and overlay network management. He is an Associate Editor of IEEE/ACM Transactions on Networking, and the recipient of best paper awards at IEEE INFOCOM 99 and ACM SIGMETRICS 2005 conferences.

Christoph Neumann received his MSc from the University of Karlsruhe (Germany) and his MEng from INPG in Grenoble (France). During his Ph.D. studies at the University Joseph Fourier (Grenoble, France) he worked at INRIA in Grenoble. He obtained his Ph.D. in 2005. He is currently a member of technical staff at Thomson Corporate Research, Paris, France. His research interests include peer-to-peer systems, large scale content distribution and Forward Error Correction codes. He is a member of ACM and IEEE.

Don Towsley holds a B.A. in Physics (1971) and a Ph.D. in Computer Science (1975) from University of Texas. He is currently a Distinguished Professor at the University of Massachusetts in the Department of Computer Science. He has held visiting positions at IBM T.J. Watson Research Center, Yorktown Heights, NY; Laboratoire MASI, Paris, France; INRIA, Sophia-Antipolis, France; AT&T Labs - Research, Florham Park, NJ; and Microsoft Research Lab, Cambridge, UK. His research interests include networks and performance evaluation. He currently serves as Editor-in-Chief of IEEE/ACM Transactions on Networking, and on the editorial boards of Journal of the ACM and IEEE Journal on Selected Areas in Communications and has previously served on numerous other editorial boards. He was Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE '92 conference and the Performance 2002 conference. He is a member of ACM and ORSA, and Chair of IFIP Working Group 7.3. He has received the 2007 IEEE Koji Kobayashi Award, the 2007 ACM SIGMETRICS Achievement Award, the 1998 IEEE Communications Society William Bennett Best Paper Award, and numerous conference/workshop best paper awards. Last, he has been elected Fellow of both the ACM and IEEE.

Matteo Varvello is a Ph.D. candidate in the joint program between Thomson Paris Research Lab and Eurecom Institut. In 2006, he earned his MEng in Networking Engineering from Polytechnic of Turin. In 2006, He received his Post grad Diploma in Telecommunications and his MS degree in Networking and Distributed Systems from Institut Eurecom and Ecole Polytechnique, respectively. He spent six month in Thomson Paris Research Lab as a research intern in 2006. His current research interests focus on the study and development of a peer-to-peer architecture for Networked Virtual Environments.