Achievable Catalog Size in Peer-to-Peer Video-on-Demand Systems*

Yacine Boufkhad† Fabien Mathieu‡ Fabien de Montgolfier† Diego Perino‡ Laurent Viennot§

Abstract We analyze a system where n set-top boxes with same upload and storage capacities collaborate to serve r videos simultaneously (a typical value is $r = n$). We give upper and lower bounds on the catalog size of the system, i.e. the maximal number of distinct videos that can be stored in such a system so that any demand of at most r videos can be served. Besides r/n, the catalog size is constrained by the storage capacity, the upload capacity, and the maximum number of simultaneous connections a box can open. We show that the achievable catalog size drastically increases when the upload capacity of the boxes becomes strictly greater than the playback rate of videos.

1 Introduction

Video on Demand (VoD) is the next challenge in Content Distribution over the Internet. Bandwidth requirements for serving quality content to a large number of customers are very high: VoD operators have to operate a large number of servers with high-speed network links, and other expensive resources.

An innovative approach consists in taking advantage of peer-to-peer algorithms. Currently, home Internet bandwidth access is relatively scarce with respect to video playback rate, and thus peers can only alleviate the load of servers, which are still mandatory. However, server-free architectures become possible as the access bandwidth increases, for example by the dint of new technologies such as fiber to the home. Most recent work in that trend study the problem of *single video distribution*, where peers viewing the same video collaborate [6, 4, 2]. The challenge is then to reduce as much as possible startup and seek delays.

Another issue is to manage a distributed catalog of videos. We indeed consider the problem of fully decentralizing the server in a set of entities with storage and communicating capacities, called *boxes*. A challenging task then resides in offering to the user the largest possible catalog. This is where a peer-to-peer approach becomes powerful as the storage capacity of the system increases with the number of boxes operated for the server part. A key problem to solve is then *video allocation*: how to store a maximum number of videos on n boxes so that any request of at most r videos can be satisfied. Notice that such an allocation scheme must be combined with a scheduling algorithm to decide which boxes handle each request.

*Supported by Project-team *GANG* between INRIA and LIAFA.

†Université Paris Diderot, Paris, France

‡Orange Labs, Issy-les-Moulineaux, France

§INRIA Paris Rocquencourt, France

An interesting target for operating such a fully decentralized VoD system is to build a distributed server using a set of boxes on the company's side for serving $r \gg n$ customers. Relying on a single video distribution algorithm between peers, the problem is then to serve at most r distinct requests (one video source per single video distribution swarm). We may thus consider the particular case where requests are pairwise distinct.

Another interesting target resides in the set-top boxes operated by an Internet Service Providers (ISP) on the customer's side. As these boxes may contain a hard drive, this mass storage capacity, combined with the upload power of the set-top boxes, may be used both as a cooperative video viewing system and as a distributed server. In that case, using a single video distribution scheme reduces the upload of participating boxes to serve as sources. We will then typically consider $r \leq n$ requests (assuming that each box either watch a video or is idle).

Using the storage capacity of set-top boxes of a P2P VoD system as a cache is proposed in [1], but that still relies on servers for sourcing the system. Our approach mostly follows the track opened by Push-to-Peer [7] where set-top boxes are really used as sources of the system (after a *push* phase has been completed). However, Push-To-Peer combines both a queuing model and an allocation scheme tailored for boxes with upload capacity smaller than playback rate. In particular, the catalog size achieved in [7] do not increase with n when the number of simultaneous connections is constrained. We focus on the allocation problem for boxes with upload capacity greater or equal to the playback rate and study how the catalog size may increase with n.

Contribution In this paper, we derive upper and lower bounds on the catalog size of a distributed VoD system with regards to box capacities. We assume that each box has storage space equivalent to d videos and upload capacity of u video streams split among c connections at most (see Table 1 for a complete list of the parameters used). A trivial upper bound on the catalog size is dn. Note that for serving r requests it is necessary that $un \geq r$ (that is $u \geq 1$ when $r = n$). We thus define $b = \frac{un}{r}$ the *upload provisioning*.

When the upload capacity is scarce (i.e. $b \approx 1$), we show that the catalog size is bounded by $m = (d - u)c\frac{n}{r} + un$. That holds even if all requests are pairwise distinct. On the other hand, we propose a simple cyclic scheme that achieves a catalog size of $dc\frac{n}{r}$ and enables incremental arrivals of requests. Moreover, this scheme can be modified to achieve the optimal $(d - u)c\frac{n}{r} + un$ catalog size when all requests are distinct. ($c\frac{n}{r}$ can be viewed as

n	Number of boxes for serving videos
m	Catalog size (# of videos stored)
r	Maximum # of simultaneous video requests
d	Storage capacity of a box, in # of videos
u	Upload capacity of a box, in # of full video streams
b	Upload provisioning w.r.t. requests ($b = \frac{un}{r}$).
c	# of simultaneous upload connections per box
s	# of stripes of videos (a video can be viewed by downloading its s stripes simultaneously)
q	Number of distinct videos in a request ($q \leq r$)
ℓ	Single video schemes threshold (see Section 5.3).

Table 1: Key parameters

a bound on the number of connections for downloading a video).

In the case where $b > 1$, we propose random allocation schemes that offer a catalog size of $\Omega(dn)$ up to some logarithmic factor. This shows that the catalog size of the system may increase almost proportionally to its storage capacity as soon as the upload provisioning b is greater than the playback rate. More precisely, we first show that a random allocation of $O(\log n)$ copies per video allows to offer a catalog size of $\Omega(dn/\log n)$ and to answer a set of r distinct requests with high probability. We then use expander graph techniques to derive a scheme that can always serve any set of r distinct requests with catalog size $\Omega(dn/\log d)$.

These results are then generalized in various ways to the general case where some videos can be multiply requested. First, the videos can be copied according to their popularity under the realistic assumption that the popularity of videos follows a power law distribution. In that case, the lower bounds obtained for distinct requests then still hold up to a $1/\log r$ factor. Another possibility it to use multicast (if available) at the cost of some startup delay. More generally, we can rely on an efficient single video distribution scheme to enhance the random allocation scheme and still achieve a catalog size of $\Omega(dn/\log n)$. Beside the theoretical analysis, we validate the random allocation approach through simulations showing experimentally that the system answers efficiently random and even adversarial requests. Our results are summarized in Table 2.

2 Model

We consider a set of n boxes. Each box has storage capacity of d videos and upload capacity equivalent to u video streams (for instance if $u = 1$ a box can upload exactly one stream). We thus assume that all boxes have same constant upload capacity and that all videos are of equal bit-rate. Additionally, we suppose that each box is always available. These boxes are used to serve video requests incoming incrementally. As new requests arrive and old requests terminate, the number of simultaneous requests at a given time is supposed to remain bounded by r. For each request, a *download allocation* algorithm designates the set of boxes that will collectively upload the requested video. This algorithm can be *fully incremental* when the designated boxes remain the same until the end of the request, or *weakly static* when the designated boxes remain the same until arrival of the next request, or *dynamic* when the designated boxes may change due to complex storage allocation schemes. Of course, with regard to connection stability, the former is the best. We propose only fully incremental or weakly static schemes. Our fully incremental schemes will be distributed as long as it is possible to learn and probe the boxes storing a given video. On the other hand, our weakly static schemes may require some centralized computation. Our upper bounds hold for all download allocation types as long as connections have to remain stable during short periods of time.

To enable collective upload of a video, each video may be divided in s equal size *stripes* using some balanced encoding scheme. The video can then be viewed by downloading simultaneously the s stripes at rate $1/s$ (playback rate is normalized to 1). A very simple way of achieving striping consists in splitting the video file in a sequence of small packets. Stripe i is then made of the packets with number equal to i modulo s.

We suppose that a box may open at most c connections for uploading simultaneously video data. The main reason is that we assume that a downloader may only open a limited number of connections to ensure a low startup time and manageable protocols. A balanced scheme must then result in a bounded number of upload connections per box. Another reason is that the total goodput decreases if too many connections are involved [3]. We additionally assume that the connections of a box equally share its bandwidth as it would typically be the case for simultaneous TCP connections.

We first give upper and lower bounds for a system with sparce capacity. We then focus on the problem where the boxes constitute a distributed server used for serving distinct requests, as discussed in the introduction. Finally, Section 5.3 deals with the problem where set-top boxes both serve and view videos (in that case we will assume $r = n$).

3 Scarce upload capacity ($b \approx 1$)

3.1 Full striping

As stated in [7], there exists a simple optimal scheme for video allocation when the number of simultaneous connections is unbounded: full striping. Assume each video can be split in n stripes. A system with n boxes can then store $m = dn$ videos by allocating to each box one stripe of rate $1/n$ for each video. Any request for r videos will then result in a demand of r stripes per box. They can always be served as long as $u \geq r/n$. Notice that dn is a trivial upper bound on the number of videos that can be stored in the system. However, we show that a system with scarce upload capacity and limited number of simultaneous connections cannot offer as many videos as it can store.

3.2 Cyclic allocation

Theorem 1 *In the special case where $b = 1$ (i.e. $un = r$), it possible to offer $dc\frac{n}{r}$ videos. Moreover, it is possible to offer $(d-u)c\frac{n}{r} + un$ videos when requests are distinct.*

Proof: Choose a coding scheme with $s = cn/r$ stripes (for the sake of simplicity we assume that cn/r and n/s are integers). For each $0 \leq i < s$, store Stripe i on the n/s boxes with number j such that $j = i$ modulo s. Any demand for $r \leq n$ videos can then be satisfied: demanded video number j can be downloaded from the boxes with number $j, j+1, \ldots, j+s-1$ (modulo n). Each box then have to serve at most $rs/n = us = c$ demands of upload $1/s$. Note that we can achieve a slightly better bound of $d(s+1)$ if the connection from a box to itself is not accounted in the simultaneous connection constraint. This download allocation scheme is fully incremental.

When a request consists in r different videos, a catalog size of $(d-u)c\frac{n}{r} + un$ videos in the system can be achieved as follows. Store $(d-u)c\frac{n}{r}$ videos according to the previous scheme plus un videos "uniquely" stored in the following sense: stripe i of video j is stored on box number $i+j$ modulo n. A request for $r = un$ videos is satisfied by allocating first the demands for uniquely stored videos. Each remaining video v can then be served by the s boxes storing a uniquely stored video which is not part of the request. If ever this uniquely stored video is then requested, v has to be viewed from another set of boxes with free capacity. (For that purpose, a DHT or a tracker could maintain which uniquely stored videos are not currently demanded.) The download allocation scheme is thus weakly static. □

Note how using striping allows to increase catalog size in these schemes.

3.3 Upper bound

Theorem 2 *In the case where $un = r$, the number m of videos offered by the system is at most $m = (d-u)c\frac{n}{r} + un$.*

Proof: Consider a set-top box a storing data from i different videos. The number of videos that are not stored in a is at most $u(n-1)$, otherwise a request for $un = r$ videos not stored in a would fail because of a lack of upload capacity. Therefore $m \leq i + u(n-1)$. If $i \leq (d-u)c\frac{n}{r} + u$, then we get $m \leq (d-u)c\frac{n}{r} + un$. Now consider $i = (d-u)c\frac{n}{r} + 1 + j$ for some integer $j \geq u-1$ and suppose $m \geq (d-u)c\frac{n}{r} + un + 1 = i + un - j$. Ask for the j videos with fewest data on a plus $un-j$ videos not on a. If these j videos represent less than an amount u of data on a, the system will not have the capacity to serve these r videos as box a will upload strictly less than $u = r/n$ videos, a contradiction. They must thus represent not less than u. Moreover, the $(d-u)c\frac{n}{r} + 1$ videos with larger portions of data share a storage space of at most $d-u$. The j videos with fewest data on a thus occupy less than $r/(nc)$, implying $j > \frac{un}{r}c = c$. Consider then a request with c videos among these j videos with fewest data on a plus $n-c$ videos excluding the $(d-u)c\frac{n}{r} + 1$ videos with largest portion of data on a. As a can upload on c connections at most and uploads less than $r/(nc)$ per connection, a will upload less than r/n and the system will not have the capacity to serve these r videos. This brings again a contradiction. We thus conclude with $m \leq (d-u)c\frac{n}{r} + un$.

The above argumentation assumes that the connections of a do not change over time. However, the result still holds with dynamic connections. Consider a synchronous request of r videos (all viewers start simultaneously). Focus on any time window $[t_1; t_2]$ where the connections of a remain stable. Let $t = t_2 - t_1$ denote its duration as a fraction of the average video duration ($t < 1$). We can then apply the static arguments on the portions of video data in the time window $[t_1; t_2]$. If d' is the storage space of a dedicated to this time window over all videos, we then similarly deduce that the number of videos in the system is at most $(d' - ut)\frac{c}{t}\frac{n}{r} + un$. Note that some time window uses a storage space $d' \leq td$ (otherwise we would get a contradiction by summing over storage spaces dedicated to all time windows). We thus obtain the same bound. □

Note that we get a similar bound in a system where downloaders are constrained by a maximum number c' of simultaneous connections. If all peers open at most c' download connections, then some box has $c'\frac{r}{n}$ upload connections at most. With the same arguments, we then get $m \leq (d-u)c' + un$.

4 Pairwise distinct requests

Here is discussed the case where each video may be requested by only one user (or one swarm of users). The extension to requests with multiplicity shall be discussed in Section 5.

4.1 Random allocation

We now prove some lower bounds based on random allocation of copies of video stripes on boxes. We assume that each video is split in $s < c\frac{n}{r}$ stripes and that the upload provisioning is slightly greater than 1:

$$u \geq \frac{r}{n} + \frac{1}{s}, \text{ i.e. } b \geq 1 + \frac{1}{c-1} \quad \left(\text{for } s = (c-1)\frac{n}{r}\right)$$

Theorem 3 *With $u \geq r/n + 1/s$ and $c \geq us$, it is possible to store $\Omega(dn/\log n)$ videos in the system and to satisfy any demand of r distinct videos with high probability.*

Proof: For each stripe of a video, select $\beta \log nrs$ boxes uniformly at random and copy the stripe to each of them (β is a constant made explicit later on). With high probability, $\Omega(dn/\log n)$ videos can be stored in the system without exceeding the storage capacity of any box ($r \leq un$ implies $rs = O(n)$).

Consider a request for r distinct videos. Each box can upload $us \geq sr/n + 1$ stripes simultaneously. The allocation of video distribution is made greedily. Consider the problem of finding a box for downloading a given stripe after $S < rs$ stripes have already been allocated. The number of boxes with exceeded capacity is at most $S/us < r/u$. The new stripe cannot be allocated if its $\beta \log(ns)$ copies fall into these boxes. This happens with probability $\left(\frac{r/u}{n}\right)^{\beta \log nrs} \leq (nrs)^{-\beta \log b} \leq \frac{1}{nrs}$ for $\beta = 1/\log b$. (Note that $\beta \leq 1/\log(1 + n/(rs))$). All videos can be served if no stripe allocation fails. Any subset of r pairwise distinct videos can thus be downloaded simultaneously with probability at least $1 - \frac{1}{n}$. □

Notice that the above download allocation scheme is fully incremental: former requests are never reallocated and it has very low probability of blocking a demand as long as the number of demands is less than r.

4.2 Expander graph allocation

We now give a procedure to allocate $\Omega(dn/\log d)$ videos in the system so that any request for r distinct videos can always be satisfied.

Theorem 4 *For $ds = \Omega(\log n)$ and $u \geq \frac{r}{n} + 1/s$, it is possible to store $\Omega(dn/\log d)$ videos in the system so that any request for r distinct videos can always be satisfied.*

Proof: Use an expander graph like construction (as sketched below) to find a bipartite graph between video stripes and boxes verifying the following cardinality property: for all subset S of at most rs stripes, the set $B(S)$ of boxes containing these stripes verifies $|B(S)| \geq |S|/(us)$. By Hall's theorem, there always exists a matching between the stripes of any subset of r videos and the boxes such that each box has degree at most us (upload capacities are thus respected). Such a matching can indeed be computed running a maximal flow algorithm. (Hall's theorem can be seen as a min-cut max-flow theorem.)

We just give a sketch of a randomized construction of an expander graph with the desired cardinality property. See [5] for deterministic expander graph construction. We consider a number of copies $k > 1$ and some constant $\beta > 0$. Store βdns stripes in the system by copying each stripe on k boxes chosen uniformly at random. This defines the allocation bipartite graph $G_{\beta,k}$. As $ds = \Omega(\log n)$, we can find $\beta < \frac{1}{k}$ such that no box has degree more than ds with non zero probability (using Chernoff's bounds for example). Consider any subset S of $|S| = i \leq rs$ stripes. We have $Pr[|B(S)| < p] < \binom{n}{p}\left(\frac{p}{n}\right)^{ki}$. Using $\binom{n}{p} \leq (ne/p)^p$, plugging $p = i/(us)$, setting $un = br$, and considering all subsets S of at most rs stripes, we get the following bound on the probability that $G_{\beta,k}$ has not the desired cardinality property: $\sum_{i=1}^{rs}\binom{\beta dns}{i}\binom{n}{i/(us)}\left(\frac{i}{uns}\right)^{ki} \leq \sum_{i=1}^{rs}\left(\frac{\beta d(br/u)s\,e}{i}\right)^{i}\left(\frac{brse}{i}\right)^{i/(us)}\left(\frac{i}{brs}\right)^{ik} \leq \sum_{i=1}^{rs}\left(\frac{\beta bde^{1+1/(us)}/u}{b^{(k-1/(us)-1)}}\right)^{i}$ for $k \geq 1 + 1/(us)$ since $i \leq rs$. It is less than 1 for $k > 1 + \frac{1}{us} + \frac{\log 2\beta de^2/u}{\log b}$. There thus exists a bipartite allocation graph having the desired cardinality property without exceeding any box storage capacity. (With a logarithmic number of trials, such a bipartite graph can be found with high probability.) □

Note that once a bipartite graph with the cardinality property has been decided for the video allocation, any request for at most r distinct videos can always be satisfied. The download allocation scheme is weakly static: when the set of requests changes, a new matching for connections has to be found (typically running a maximal flow algorithm starting from the previous matching). This may require to reconfigure many connections. We test in Section 6 how requests can be inserted fully incrementally in similar bipartite graphs (without any reconfiguration).

5 Managing videos multiply requested

Let us now suppose that the same video can be watched simultaneously by many users. As this context is specially interesting for the set-top box setting (box on the customer's side), we consider $r = n$ requests (each box views at most one video) for a total of $q \leq r$ distinct videos watched. However, our results could be extended to the case where a box may view up to $i \geq 1$ videos simultaneously (implying $r = in$). Note that the full striping and the cyclic allocation scheme (with dc videos) still apply when there may be multiple requests for the same video.

5.1 Power law video popularity

The lower bounds of Section 4 still hold up to a $1/\log r$ factor if the multiplicity of requests follows a power law distribution. More precisely, assume that for every video we know a bound i on the maximum number of simultaneous viewing it may reach. We then say it is a *type i* video. For $1 \leq i \leq r$, let p_i be the proportion of videos of type i. We postulate that popularity follows a power law with parameter γ, given by $p_i = i^{-\gamma}/\sum_{i=1}^{r} i^{-\gamma}$.

Consider a catalog of size $m'/\log r$ where m' is the catalog size that can be achieved by some allocation scheme (e.g. one of the schemes proposed in Theorems 3 and 4). We then constitute a redundant catalog where each video of type i is duplicated i times. The catalog size is then $\frac{m'}{\log r}\sum_{i=1}^{r} ip_i$. As $\sum_{i=1}^{r} ip_i = \frac{\sum_{i=1}^{r} i^{1-\gamma}}{\sum_{i=1}^{r} i^{-\gamma}} \leq \log r$ when $\gamma > 2$, this redundant catalog may be allocated using a distinct request scheme. Requests for the same video are then allocated to different instances of the video in the catalog.

5.2 Multicast

In the case where the boxes, on the customer's side, belong to the same DSLAM (or some specific subnetwork), *multicast* may be used: a stripe uploaded from a box can be downloaded simultaneously by many other ones. However, this requires the viewers to be synchronized. This can be obtained by splitting the videos in small duration windows. The price to pay is that all the users watching the same video progress in parallel in their window, and so a user may wait a window duration before starting playback. (A simple solution to reduce startup time is that all users store the first window of all videos [7]). Then if we use both striping and windowing, the problem of multiple requests trivially reduces to the problem of pairwise distinct requests.

5.3 Using single video distribution schemes

The problem of *single video distribution* has already been addressed (e.g. [6, 4, 2]). If a set p peers watching the same video receive data from other peers at a global rate k, the minimum bandwidth they must allocate to the video is $p - k$, leaving a remaining upload capacity of at most $(u-1)p + k$. If this bound is reached, the distribution scheme is called *perfect*. An instance of perfect distribution scheme is *multiple chaining*: in a chain of users, the most advanced user, i.e. the one watching the furthest, downloads the video as if it were a single watcher. The second

most advanced user downloads directly from the the first one, and so on... We obtain a "perfect" scheme by using k chains.

However, perfect schemes need tight schedulers, and for large number of users, more robust schemes must be used. Such schemes usually have average download and upload requirements of $1+\epsilon$ per peer (we neglect ϵ in the sequel), leaving a remaining overall upload capacity of $(u-1)p$.

We model this distinction between perfect and robust schemes by introducing a critical size $\ell \geq 1$: if $p \leq \ell$, we assume the scheme is perfect (remaining capacity: $(u-1)p+k$); otherwise, a robust scheme is used (remaining capacity: $(u-1)p$). Of course we may set $\ell = 1$ if we suppose that perfection does not exist. In that case, using the scheme of Section 4.2, we can achieve the same catalog size with multiplicity in requests when $u \geq 2+1/s$. Theorem 5 generalizes this result for any ℓ.

Theorem 5 *With $u \geq 1+1/s+1/\ell$ and $c \geq us$, it is possible to offer $\Omega(dn/\log n)$ videos and to satisfy any demand of $r = n$ videos with high probability.*

The proof is similar to that of Theorem 3, except that the single distribution scheme capacity requirements have to be taken into account.

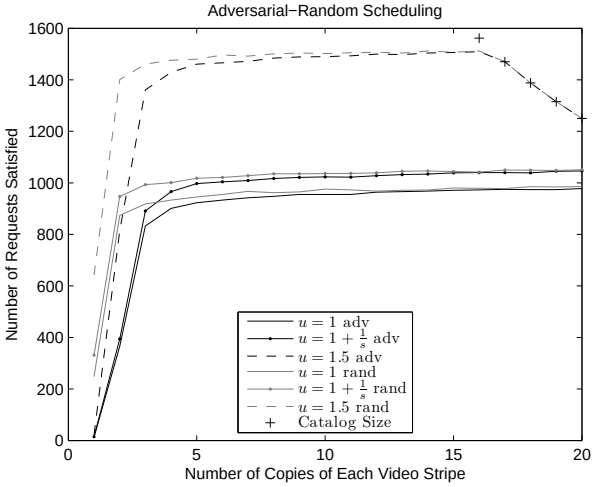
6 Simulations

In this section, we use a simulator to study the performances of a regular allocation scheme. Each video is divided in s stripes replicated k times. To equally fill up boxes and equally copy stripes, we build a regular random bipartite graph rather than the purely random allocation used for theoretical lower bounds. The reason is that it allows to fully utilize the storage capacity of boxes compared to a random allocation that is inherently unbalanced. The mks stripes are thus placed according to a random permutation in the nds available storage slots (we assume $mk = nd$).

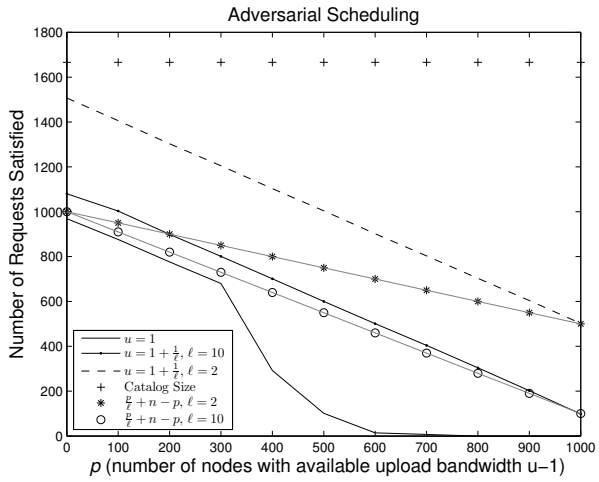
Once the allocation is made, the simulator tries to fulfill video requests until a request fails. We do not consider dynamic reallocations as requests arrive, although it could probably increase the number of requests the system can handle. We indeed use the following fully incremental scheme. Once a given video is requested, the system takes the corresponding stripes from the nodes with the more available upload bandwidth. The request scheduler can be *random* or *adversarial*. In the former, videos are randomly selected among the ones not selected yet. In the latter, the scheduler chooses the video for which the system will select a node with minimal remaining upload capacity.

We simulate a $n = 1000$ nodes system, with a storage capacity per node of $d = 25$, and a number of stripes per video of $s = 15$.

Pairwise distinct requests Following Section 4, we first analyze a scenario where all requests are distinct. We consider three different node upload capacity scenarios: $u = 1$ (scarce capacity), $u = 1+\frac{1}{s}$ (extra-stripe capacity) and $u = 1.5$. The targeted number of requests is n, but being able to perform more than n requests also makes sense if we assume that some nodes may want to watch more than one video. We vary the number of copies per video from $k = 1$ to $k = 20$ (the catalog size is then $m = \frac{nd}{k}$). Figure 1(a) shows the minimum number of request satisfied for each scenario among several runs.



(a) Distinct requests scenario



(b) Multiple requests scenario ($k = 15$)

Figure 1: Requests scenarios ($n = 1000$, $d = 25$, $s = 15$, $c \geq us$).

First, we can notice that *random* and *adversarial* requests give similar results except when the number of copies is very small. This means that if the catalog is redundant enough, the system will be unaffected by the request patterns. Figure 1(a) shows that the system performs quite well with distinct requests: with $u = 1$, it serves 95% of n requests if $k = 7$, and 98% if $k = 20$. However, it never satisfies 100% of n requests. If we increase the upload capacity up to $u = 1+\frac{1}{s}$, then n requests can be satisfied as long as $k \geq 5$. This means that a bandwidth overhead of $\frac{1}{s}$ and a redundancy of 5 provide a catalog robust to distinct requests. By increasing again the upload capacity to $u = 1.5$, less copies are needed to satisfy all the n requests and the system answers up to almost 1500 demands.

Capacity Constraints	Achievable lower bound	Theoretical upper bound
$b = 1$	$m = (d-u)c\frac{n}{r} + un$ (distinct requests), $m = dc\frac{n}{r}$ (any requests)	$m \leq (d-u)c\frac{n}{r} + un$
$b \geq 1 + \frac{1}{c-1}$	$m = \Omega(dn/\log n)$ for distinct requests with low failure probability, or $m = \Omega(dn/\log d)$ for distinct requests with an expander graph (Theorems 3 and 4 with $s = (c-1)\frac{n}{r}$)	$m \leq dn$
$u \geq 1 + \frac{1}{s} + \frac{1}{\ell}$, $r = n$	$m = \Omega(dn/\log n)$ for any multiset of requests with low failure probability (Theorem 5).	
$u \geq 2 + \frac{1}{s}$, $\quad r = n$	$m = \Omega(dn/\log d)$ for any multiset of requests (Theorems 4 using efficient single video distribution).	

Table 2: Results reminder

These results also show that the system almost fully exploits the available bandwidth except when there are too few copies per video or when the catalog size is lower than un.

Multiple requests We take multiple requests into account by altering node upload capacities: we assume that p nodes ($p \leq n$) have an available bandwidth for distinct requests $u' = u - 1$, since 1 is devoted to a single video distribution scheme (similarly to Section 5.3). The number of distinct requests with ℓ or more viewers is then p/ℓ at most. The number of distinct requests singly viewed or treated through chaining (of less than ℓ boxes) is at most $n - p$. We thus deduce that the system can manage n requests with multiplicity, if it can handle $\frac{p}{\ell} + n - p$ distinct requests with p nodes having altered upload bandwidth.

Figure 1(b) shows the number of distinct requests the system can handle with a redundancy set to $k = 15$ and adversarial request scheduling. We consider three node upload capacity scenarios: $u = 1$, $u = 1.1$ and $u = 1.5$. The $\frac{p}{\ell} + n - p$ curves are also plotted for $\ell = 2$ and $\ell = 10$ ($\ell = 2$ correspond to no chaining, only practical single video schemes and $\ell = 10$ corresponds to optimal single video schemes for less than 10 simultaneous viewers). Scenarios above those curves can manage n requests, with multiplicity, or more.

The scenario with scarce upload capacity ($u = 1$) cannot satisfy n requests. It can only be close to the limit for $p \leq 300$, $\ell = 10$, and gives poor performances for $p > 300$. However, if we increase the upload capacity of $\frac{1}{\ell}$, the result is a concave function of p that is always above the limit requested to satisfy n requests with multiplicity. As $u = 1 + \frac{1}{\ell}$ is the capacity needed to handle n requests if $p = n$, and $u = 1 + \frac{1}{s}$ is the capacity needed if $p = 0$, a capacity of $1 + \max(\frac{1}{\ell}, \frac{1}{s})$ seems sufficient in practice compared to the $1 + \frac{1}{s} + \frac{1}{\ell}$ bound suggested by Theorem 5.

7 Conclusion

In this paper, we gave several trade-offs between bandwidth and storage capacity. Main results are summarized in Table 2. One should retain that when the available bandwidth is close to the minimal bandwidth needed to fetch requests, the catalog is sparse with respect to available capacity, and multiple requests of a same video are difficult to handle. Things get better if the bandwidth is over-provisioned. In the set-top box setting, if the available bandwidth is more than twice the playback rate, the size of the catalog can be almost proportional to the available storage capacity. Moreover, a high catalog size can also be achieved with available bandwidth 50 % percent greater than playback rate and high probability of satisfying requests. Note that using striping reduces the required upload provisioning. Interesting future work resides in reducing the gap between upper and lower bounds. For example, can we get tighter upper bounds with multiplicity in requests?

References

[1] M. S. Allen, B. Y. Zhao, and R. Wolski. Deploying video-on-demand services on cable networks. In *ICDCS '07*.

[2] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez. Exploring VoD in P2P swarming systems. In *INFOCOM '07*.

[3] F. Baccelli, D. Hong, and Z. Liu. Fixed point methods for the simulation of the sharing of a local loop by a large number of interacting TCP connections. In *Proc. ITC Specialist Conference on Local Loop, Barcelona, Spain*, 2001.

[4] B. Cheng, X. Liu, Z. Zhang, and H. Jin. A measurement study of Peer-to-Peer Video-on-Demand system. In *IPTPS '07*.

[5] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bull. AMS*, 43:439–561, 2006.

[6] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? In *SIGCOMM '07*, pages 133–144.

[7] K. Suh, C. Diot, J. F. 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, 2006.