Tradeoffs in Cloud and Peer-assisted Content Delivery Systems

Niklas Carlsson*, György Dán†, Derek Eager‡, Anirban Mahanti§
*Linköping University, Linköping, Sweden, niklas.carlsson@liu.se
†KTH Royal Institute of Technology, Stockholm, Sweden, gyuri@ee.kth.se
‡University of Saskatchewan, Saskatoon, Canada, eager@cs.usask.ca
§NICTA, Sydney, Australia, anirban.mahanti@nicta.com.au

***Abstract*—With the proliferation of cloud services, cloud-based systems can become a cost-effective means of on-demand content delivery. In order to make best use of the available cloud bandwidth and storage resources, content distributors need to have a good understanding of the tradeoffs between various system design choices. In this work we consider a peer-assisted content delivery system that aims to provide guaranteed average download rate to its customers. We show that bandwidth demand peaks for contents with moderate popularity, and identify these contents as candidates for cloud-based service. We then consider dynamic content bundling (inflation) and cross-swarm seeding, which were recently proposed to improve download performance, and evaluate their impact on the optimal choice of cloud service use. We find that much of the benefits from peer seeding can be achieved with careful torrent inflation, and that hybrid policies that combine bundling and peer seeding often reduce the delivery costs by 20% relative to only using seeding. Furthermore, all these peer-assisted policies reduce the number of files that would need to be pushed to the cloud. Finally, we show that careful system design is needed if locality is an important criterion when choosing cloud-based service provisioning.**

I. Introduction

Content delivery over the Internet is rapidly growing with respect to the number of users, the intensity of their use, and the diversity of contents being accessed. Therefore, it is important to design content delivery systems that effectively scale with respect to *both* the total request rate and the number of available content items.

Content popularity typically exhibits high skews. The most popular contents account for the majority of the downloads, with a long tail of niche contents which account for a non-negligible combined request rate [1]–[4]. Peer-to-peer systems such as BitTorrent can offload the origin servers very effectively when serving highly popular contents. Unfortunately, these systems are much less effective for less popular contents [5], [6], for which there may not be sufficiently many peers to facilitate offloading. Systems that want to ensure that all clients are provided timely service must therefore often resort to using significant server resources to serve the long tail of mildly-to-less popular contents.

To improve the file availability (and reduce the amount of server upload bandwidth necessary to ensure service), both bundling [7]–[10] and incentives for seeding [11], [12] have been proposed. With bundling, clients are asked to participate in the download of a larger number of contents (which the peer in some cases may not want). With seeding, peers are voluntarily uploading contents that they have completely downloaded in the past. Both approaches leverage voluntary peer contributions to improve the file availability.

This paper considers the problem of determining the most effective ways for a content provider with a large catalogue of contents with diverse popularities to serve the content to a large number of clients. More specifically, we explore the best design tradeoffs in a system that leverages both cloud and peer resources to offload the origin servers and reduce the delivery cost of the content provider. For the peer bandwidth we consider bundling, seed-based, as well as hybrid approaches. For the content provider's upload contributions, we consider the use of origin servers, cloud servers, as well as differentiated pricing based on locality of service.

Our analysis assumes a chunk-based system in which requesting clients can be served by other clients (peers), from the origin server, or from the cloud. The content provider is assumed to have a copy of every content, but can also select to push some of the content to one or more servers in the cloud. The cloud storage and bandwidth is assumed to be elastic, but comes at a cost. The provider must select which content to push to the cloud, which locations to push the content to, as well as which servers should serve which clients.

Clients (or peers) agree to help out with the content delivery, but *only* during the times during which they download content themselves. During this time period, the peers can assist the system in three ways: (i) upload pieces of the content they are currently downloading, (ii) seeding content that they have downloaded in the past, and (iii) through dynamic bundling. Both peer seeding and dynamic bundling (or torrent inflation) [8] effectively improve the availability, or in our case reduce the server bandwidth requirements. In contrast to seeding, dynamic bundling (or inflation) has the advantage that no persistent storage is required on the peers. This may be particularly attractive in privacy-aware environments.

Based on the above system assumptions, we develop a simple cost model that allow us to identify and analyze basic tradeoffs in these systems. We first derive and evaluate simple bounds and approximations of the minimum server bandwidth required to ensure that a torrent achieves a target average download rate. Second, using these expressions, we compare simple policy classes for which content to push to

NOTICE: This is the authors' version of a work that was accepted for publication in IEEE P2P 2012.
The final/official version will appear the conference proceedings and IEEE Xplore.

the cloud and provide insights regarding the importance of careful content selection. Third, we consider the best usage of the peer upload bandwidth. We analyze and compare basic policies that are designed to determine how seeding and torrent inflation should be best utilized. Finally, we take a closer look at how to best replicate the content and where to direct clients in systems where the cloud provider has a differentiated cost model and charges based on the locality of the clients that are served (e.g., to reduce its own network costs). We consider baseline policies that balance the importance of large torrent size (for self-sustainability purposes) and locality of service.

The above model allows us to evaluate which content to push to the cloud, which torrents should be inflated, and which should be served by the origin servers. The model allows us to provide insights regarding how to best use the server, cloud, and peer resources. Our results include, but are not limited to, the following contributions and insights.

- We show that bandwidth demand peaks for contents with moderate popularity, and identify these contents as candidates for cloud-based service. We find that there can be a significant penalty to pushing the wrong contents to the cloud, and note that the long tailed nature of the popularity distribution, may result in large numbers of files having similar bandwidth requirements. Therefore, a change in the cloud-related costs can significantly change the number of files that should be pushed to the cloud.
- To the best of our knowledge, this paper is the first that combines the ideas of leveraging peer storage (for seeding) and dynamic bundling. We find that much of the benefits from peer seeding can be achieved with careful torrent inflation. Hybrid policies that combine bundling and seeding have the best performance, often reducing the delivery costs by 20% relative to only using seeding.
- We determine which torrents do benefit from additional peer assistance, and provide insights to how the full catalogue of contents are best served. Interestingly, the effective use of peer assistance reduces the number of files that should be pushed to the cloud.
- We find that carefully designed locality-aware policies are important for content providers wanting to minimize their delivery costs. While simpler policies that use fixed number of cloud replicas and/or treat all peers in a swarm equally typically achieve good performance for some parts of the parameter space, they do poorly in other parts.

Overall, our analysis shows that careful system design can result in significant cost savings. We believe our insights and conclusions will be valuable for developers of future systems.

The remainder of the paper is organized as follows. Section II presents our system model. Section III considers which content to push to the cloud, such as to best balance the server bandwidth costs and cloud-related bandwidth and storage costs. Section IV considers how to best leverage the peer contributions. Section V evaluates policies that take network locality into account. Related works are discussed in Section VI, before Section VII concludes the paper.

II. System description

A. Service model

In this paper we consider a peer-assisted system with a catalogue of $N = |\mathcal{N}|$ file contents, where $\mathcal{N}$ is the set of files. For simplicity, each file is assumed to be of size L and clients only download one file at a time. The system is assumed to employ chunk-based delivery, such as in BitTorrent, with which the peers can download pieces in parallel from other peer, servers, and/or from the cloud.

The content provider's servers are assumed to have a complete copy of every content, and thus have a storage requirement of LN. A subset $\mathcal{M} \in \mathcal{N}$ of these contents can also be served from the cloud. Let $M = |\mathcal{M}|$ denote the number of files stored in the cloud. Each of these files can be pushed to one or more cloud servers. In total we assume that there are $R = |\mathcal{R}|$ cloud server locations and the content provider pushes file i to a subset $\mathcal{P}_i \in \mathcal{R}$ of these.

Clients make part of their upload bandwidth available to the system during their download (possibly since they are provided with system complying software which ensures that they do), but *only* during the download itself. We assume that the peers are homogeneous in that they are provided with the same service agreements in terms of the *maximum average download time* T. (While we do not expect peers to be completely homogeneous in practice, we note that there likely would be minimum system requirements that the peers would have to satisfy to obtain the service.) We assume that each peer is guaranteed the same maximum download time T and in exchange agrees to upload file content (on behalf of the system) while downloading content for this duration. We assume that all peers make the same fixed upload rate U available to the system. In this paper, we only consider the case in which the requested file content L is no greater than the total amount the peer can upload during its download (i.e., T is selected such that $L \leq TU$).

In our analysis we assume that, if possible, a peer uploads within the swarm for its file of interest the same amount of data that it downloads to get this file (L bytes), but that the remaining bytes ($TU - L$) that the peer can upload may be used either to offload the servers (by seeding content downloaded in the past) or to participate in the exchange of some other file content, effectively inflating the swarm size for that content.

B. Cost model

Both server bandwidth and cloud bandwidth is assumed to be elastic, and the content provider pays a fixed price per unit of bandwidth consumed. Our analysis would allow the use of any concave cost function. However, for simplicity, we assume a linear relationship between the cost of bandwidth at the servers and within the cloud. Without loss of generality, we normalize all costs to the cost of a unit of server bandwidth at the origin servers and let the cost of cloud bandwidth be a fixed factor c of this. In this paper, we consider the case when $0 \leq c < 1$. (When $1 \leq c$ it is never beneficial to serve content from the cloud.) Furthermore, we assume that the content provider pays a fixed amount C for each file stored in the cloud.

Each cloud server is assumed to be associated with a unique locality region. For simplicity, we assume that these regions are non-overlapping, and each peer is *local* to exactly one region, and considered *remote* to all other cloud servers. Our general cost model considers the fact that a cloud provider may want to charge more for uploading to remote peers than local peers. We use a *remote access* cost q (per unit of remote bandwidth) to capture this differentiation.

Our interest is in minimizing the total delivery cost for the content provider. Given the above cost model, this corresponds to minimizing the sum of all server and cloud-related costs (while leveraging peer resources):

$$\sum_{i\in\mathcal{N}} B_i^s + c\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{P}_i} B_{i,j}^c(1+qf_{i,j}) + C\sum_{i\in\mathcal{M}}|\mathcal{P}_i|, \qquad (1)$$

where B_i^s is the server bandwidth used to serve content i, $B_{i,j}^c$ is the cloud (server) bandwidth used to serve content i at cloud server site j, and $f_{i,j}$ is the fraction of that bandwidth that is used to serve peers that are remote to site j.

C. Workload model and operational constraints

For our analysis, we consider the system in steady state. We denote the request rate of file i by λ_i, and index the files in decreasing order of popularity such that $\lambda_i \geq \lambda_j, \forall i < j$. In the cases we rely on analytical evaluation, we assume that client requests for each file follow a Poisson process. While the peer arrival process changes over the lifetime of torrents, over short periods the Poisson process can be a reasonable approximation [13]. Furthermore, it should be noted that our general conclusions do not appear to depend on the Poisson assumptions, as the qualitative shape of the various bandwidth costs likely are very similar for more general workloads. For the bandwidth usage, we have validated this claim for simple diurnal request patterns.

As noted above, each peer's upload bandwidth can (in addition to its regular upload participation for the requested file) be used for either dynamic bundling (inflation) or seeding. With bundling, peers participate in the download of other file content, in which this bandwidth effectively is used to inflate the popularity of that file. To capture these inflation contributions, we recalculate the additional file sharing of these files into an (artificial) arrival intensity ϕ_i, which effectively inflates the total request volume of files from λ_i to $\lambda_i + \phi_i$. With seeding, peers act as a seeder for content that they have already downloaded. Similar to the server and cloud bandwidth, peer seeding is only used if needed. Let B_i^p denote the seed contributions of file i. Naturally, the sum of both these types of additional peer contributions cannot exceed the available peer upload bandwidth of peers. Hence, we have that:

$$\sum_{i\in\mathcal{N}} \phi_i L + \sum_{i\in\mathcal{N}} B_i^p \leq \sum_{i\in\mathcal{N}} \lambda_i(UT-L). \qquad (2)$$

To ensure service, our peer-assisted system may need to provide additional upload bandwidth to that provided by the peers currently downloading a particular file. We will call this upload contribution the *total system upload bandwidth*. With or without inflation, the total system upload bandwidth contributions $B_i(\lambda_i)$ must be contributed by either the servers (B_i^s), the cloud ($\sum_{j\in\mathcal{P}_i} B_{i,j}^c$), or seeding peers ($B_i^p$), such as to ensure that the downloading clients obtain data at the rate L/T. Naturally, all upload bandwidth contributions B_i^* are non-negative (i.e., $0 \leq B_i^*, \forall i$) and must sum to the total required upload bandwidth requirements $B_i(\lambda_i + \phi_i)$ to ensure that the (average) service guarantees are satisfied. To summarize,

$$B_i(\lambda_i+\phi_i) \leq B_i^s + \sum_{j\in\mathcal{P}_i} B_{i,j}^c + B_i^p, \forall i. \qquad (3)$$

III. Cloud contributions: Server-cloud tradeoff

Let us first consider the tradeoff between server bandwidth usage and cloud storage. In this section we take a closer look at the importance of carefully selecting which content to push to the cloud. For this purpose, we will consider the case when the cloud provider does not differentiate remote and local service and instead focus on the comparison of five basic policies to determine which content to push. For this case $q = 0$, and there is no advantage to pushing more than one copy to the cloud. The cost model in equation (1) then reduces to the following:

$$\sum_{i\in\mathcal{N}\setminus\mathcal{M}} B_i^s + c\sum_{i\in\mathcal{M}} B_i^c + C|\mathcal{M}|, \qquad (4)$$

where B_i^c is the cloud bandwidth used to serve content i.

A. Baseline policies

We define five basic replication policies for determining which contents to push to the cloud. The more advanced policies assume knowledge of the expected total system upload bandwidth requirements $B(\lambda_i)$ of each file.

- **Random (p):** To illustrate the ineffectiveness of a naive approach, we consider a policy that selects a fraction p of the files at random, and pushes these to the cloud. Here, $p = \frac{M}{N}$ is a parameter of the policy.
- **Head (p):** This policy pushes the most popular files to the cloud. Again, p determines the fraction of the total number of files to push to the cloud.
- **Tail (p):** In contrast to the *head* policy, this policy pushes the least popular files to the cloud. Again, p determines the fraction of the total number of files to push.
- **Optimized (best case):** To illustrate the potential for careful replica management, we consider the static optimal policy that always pushes exactly the set of files that are best served by the cloud, into the cloud. This corresponds to all files for which the server upload bandwidth B_i^s otherwise would exceed $C/(1-c)$. (To see this, note that this ensures that the cloud only is used whenever $B_i^s > cB_i^c + C$, where $B_i^s = B_i^c$.) This solution will minimize the server bandwidth usage requirements.
- **Worst case:** To illustrate the worst case scenario, we have a policy that always pushes the set of files that will result in the worst case cost, by pushing exactly the wrong set of files to the cloud. Naturally, this set of files is the complete mirror of the *optimized (best case)* policy.

B. Evaluation framework

For our policy evaluation we will consider a system with N files, each of varying popularity. For simplicity, we will consider a system in steady state, with known request rates λ_i, for the simple case when peers only participate in the upload of the files they request. This corresponds to the case when $U = L/T$. The more general case will be analyzed in later sections. However, the relative tradeoff between cloud storage and server bandwidth is not significantly affected.

Furthermore, focusing on the relative cost differences of the policies, rather than the absolute values, we will use an analytic approximation of the total system upload bandwidth for an example policy that could be used by the servers or the cloud (servers) to serve a single file.

While the system upload bandwidth in general can be provided by the origin servers, the cloud, or even other peers seeding the content (after previously having downloaded the content), it is important to note that typically only one such entity is needed to keep a torrent alive. Naturally, at very low request rates (where there is no other peers helping out) this node may be required to upload at rate $\lambda_i L$. In contrast, if the torrent is self-sustaining no server bandwidth is needed. It has been observed that a critical mass typically is required for torrents to become self-sustaining [6], [8].

In this paper, we define two server-based policies that attempt to upload only the minimum amount of pieces required to maintain a self-sustaining torrent, as indicated by there being at least one copy of each piece among the set of downloaders of the content.

- **Missing piece:** The server only uploads one piece at a time whenever there is at least one piece missing among the peer set.
- **Missing piece, with fewest priority:** Same as the *missing piece* policy, but the server always gives priority to the peers with the fewest pieces (with ties broken at random).

With both policies, when the server uploads pieces, it uploads at rate U, equal to the client upload rate.

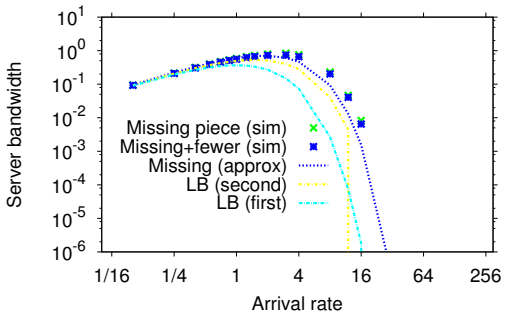
For the purpose of our evaluation, we will use an *approximation* of the total system upload bandwidth required by these policies. We consider the number of pieces lost from the torrent whenever a peer departs leaving just k remaining peers. Assume that the oldest such remaining peer has a fraction $k/(k+1)$ of the pieces, the next oldest peer has a fraction $(k-1)/(k+1)$ of the pieces, the next oldest after that has $(k-2)/(k+1)$, and so forth. Furthermore, assume that the sets of pieces held by these peers are independent. The fraction of pieces lost from the torrent would then be on average $k!/[(k+1)^k]$, and under the Poisson assumption the server upload bandwidth for these policies can be calculated as:

$$B_{missing} \approx \sum_{k=0}^{\infty} \frac{(\lambda L/U)^k}{(k+1)^k} e^{-\lambda L/U} \lambda L. \qquad (5)$$

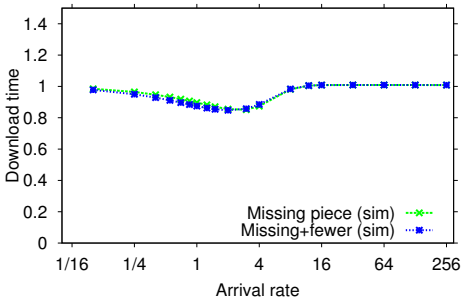
Note that the rate at which a peer departs leaving just k remaining peers is given by λ times the probability that there are k peers in the system (from flow balance).

To evaluate the accuracy of our analytic approximations and to provide some insights to the required system upload bandwidth usage as a function of the request rate we used simulations. For our simulations, we modified an existing BitTorrent simulator [14] which captures the BitTorrent dynamics, including the rarest-first policy, rate-based tit-for-tat, as well as the bandwidth constraints of the peers and the server. In accordance with our model, these simulations assume that peers do not depart before having downloaded the full copy, but leave as soon as they have completed their download. Furthermore, peers are assumed to be homogeneous with an upload capacity $U = 1$, download capacity $D = 3$, and arrive according to a Poisson process. Without loss of generality, the files size L is chosen to be 1. This corresponds to normalizing the file size, and measuring the bandwidth in units of files that can be transferred during the time it takes to upload an entire file, and the request rate corresponds to the average number of requests during the time it takes to upload a file.

Figure 1 shows the server bandwidth usage and average client download times for the above policies, as function of the file request rate. To give some insight to the potential performance of these policies, we also include two *lower bounds* for the minimum possible system upload bandwidth. The derivation of these bounds is provided in the appendix.



(a) Server (system) upload bandwidth



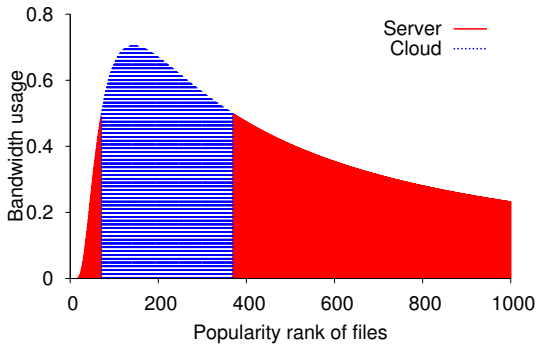
(b) Client delay

Fig. 1. Validation of server upload bandwidth approximation. (Default parameters: $D = 3$, $U = 1$, $L = 1$, and a file with 512 pieces.)

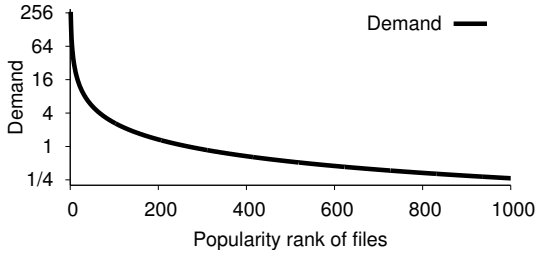
We note that the torrents become self-sustaining at request rates somewhere above $\lambda = 16$. While the *approximation* and *lower bounds* both track the general shape of the server bandwidth usage, there appears to be some room for tighter bounds or improved policies. However, as our focus is on

the general design tradeoffs of these systems, rather than the absolute server bandwidth values, the simple policies and their bandwidth approximations will suffice. It is also encouraging to see that the policy in general appears to achieve download times close to the desired download time of one.

Finally, it should be noted that the general qualitative shape of $B(\lambda)$ may be more generally true. While the approximations and bounds only are based on the Poisson assumption, the general qualitative shape is likely very similar for more general workloads (with the most bandwidth being consumed by torrents with intermediate request rates). We have validated this claim for simple diurnal request patterns.



(a) Bandwidth usage



(b) Request rates

Fig. 2. Characterization of the optimal cloud storage policy when there is no additional peer contributions. (Default scenario with $N = 1,000$ files, Zipf popularity with $\alpha = 1$, average request rate $\bar{\lambda} = 2$, cloud storage cost $C = 0.25$ and cloud bandwidth cost factor $c = 0.5$.)

C. Performance evaluation

In our experiments we consider a scenario in which the file popularities follow a Zipf distribution. We denote the average file request rate by $\bar{\lambda} = \frac{1}{N}\sum_{i=1}^{N}\lambda_i$.

Figure 2 shows the server and cloud bandwidth usage for the *optimal* cloud storage policy. For this experiment we have used our default scenario with cloud storage cost $C = 0.25$ and cloud bandwidth cost factor $c = 0.5$. The blue (lighter color) in the middle illustrates the files that should be pushed to (and served in) the cloud. The other files (red, darker color) should be served by the server. We note that some of the most popular files are self-sustaining and do not require any server (or cloud) bandwidth. In later sections, we will show how the extra peer upload bandwidth can be used to further reduce the delivery costs of the content provider.

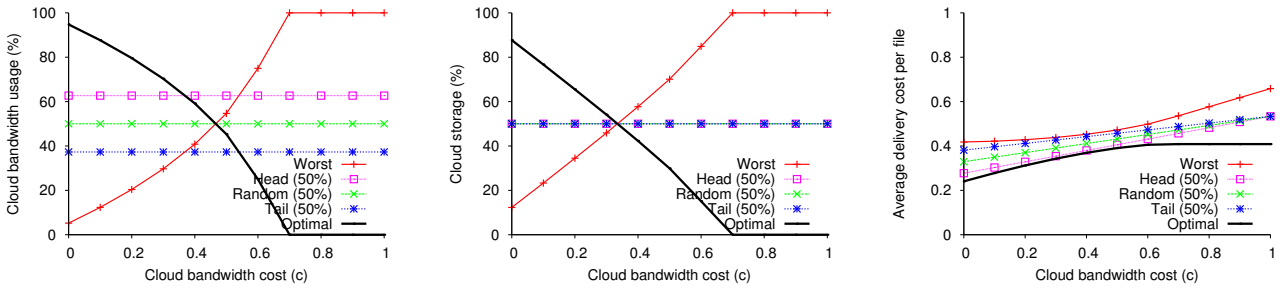
We next look at the impact of cloud storage and the file-selection policy used to push content to the cloud. Figures 3 and 4 break down the cloud bandwidth usage, cloud storage requirements, and total delivery cost, as a function of the cloud bandwidth cost c and cloud storage cost C, respectively. We note that the total system bandwidth usage is the same in all these scenarios, and only the costs and policies differ. While the three static policies (with $p = 50\%$) have a fixed allocation, the *optimal* policy carefully adjusts its cloud usage based on these costs. In all cases this policy achieves distinctly lower costs. While our default scenario (with $c = 0.5$ and $C = 0.25$) corresponds to a parameter region where the cost differences between the policies are relatively smaller, in general, there is a significant difference between the *optimal* policy and the other policies. This shows that it is important to carefully select which content to push to the cloud.

To assess the impact of the popularity and load characteristics of the content provider's file catalogue, we next consider the impact of varying the load parameters. Figure 5 shows a comparison of the delivery costs of the example policies, when varying the average request rate $\bar{\lambda}$, the popularity skew parameter α, and the number of files N. While the absolute costs are the highest for average request rates around $\bar{\lambda} = 4$ and for a flat popularity distribution, we note that the relative cost differences between the policies are the greatest when the request rates are either high, low, or the overall load is skewed (high α). We also note that our results appear relatively insensitive to the number of files.

IV. Peer contributions: Bundling and seeding

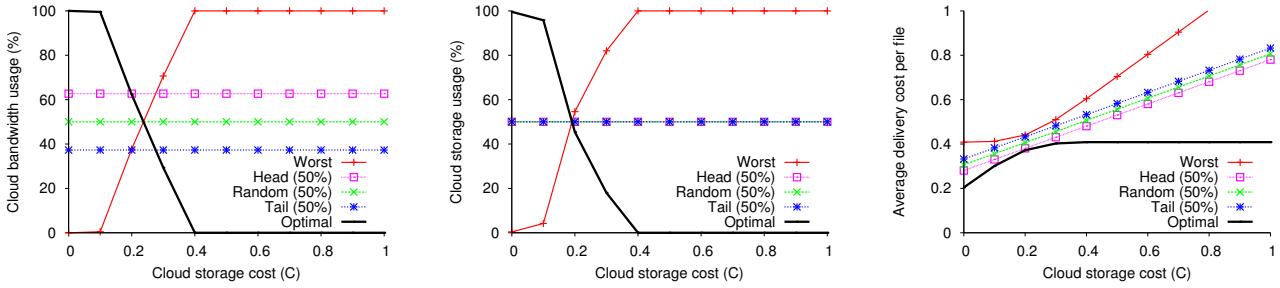
Having provided insights into how a content provider should best utilize cloud resources, we next move our attention to the peer upload bandwidth. In particular, we are interested in how the content provider should best utilize the additional upload contribution ($UT - L$) provided by each downloader. As described in Section II, this bandwidth can be used for dynamic bundling (inflation) and seeding of previously downloaded contents. We note that the first approach does not require any permanent peer storage, while the second approach does. At a high level, we consider three policies:

- **Peer seeding (only):** With seeding, the peers simply serve peers on behalf of the server when the server otherwise would need to serve the content (according to the missing piece policy, described in Section III, for example). Of course, peers can only seed contents they have downloaded in the past. For simplicity, in this section we assume that the probability that a peer has a copy is proportional to the request rate of that file content.
- **Bundling (only):** With bundling peers agree to participate in the download of some other file contents in addition to the file they requested. Allocation of the inflation bandwidth contributions of the peers are described below.
- **Hybrid:** With the hybrid policy, we use both peer seeding and bundling. The default is to use bundling; however, as soon as a peer is given a seeding opportunity, this is given priority (so as to avoid a torrent dying).



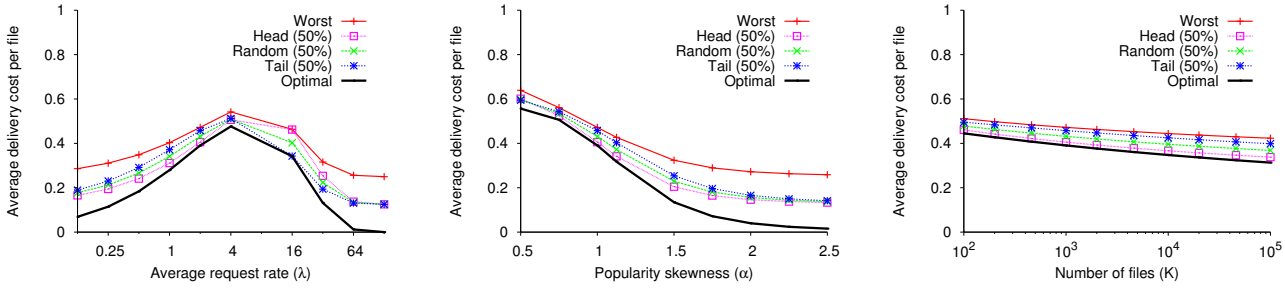
(a) Fraction bandwidth from cloud (b) Fraction stored in cloud (c) Average delivery cost

Fig. 3. Resource usage and cost comparison of example policies when varying the cloud bandwidth cost factor c. (Default scenario with $N = 1,000$ files, Zipf popularity with $\alpha = 1$, average request rate $\overline{\lambda} = 2$, and cloud storage cost $C = 0.25$.)



(a) Fraction bandwidth from cloud (b) Fraction stored in cloud (c) Average delivery cost

Fig. 4. Resource usage and cost comparison of example policies when varying the cloud bandwidth cost factor c. (Default scenario with $N = 1,000$ files, Zipf popularity with $\alpha = 1$, average request rate $\overline{\lambda} = 2$, and cloud bandwidth cost factor $c = 0.5$.)



(a) Average request rate (b) Popularity skew (c) Number of files

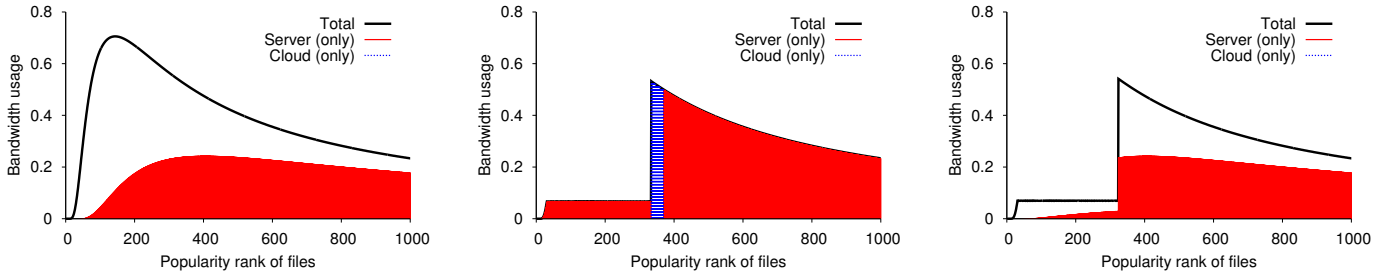
Fig. 5. Delivery cost comparisons of example policies when varying (a) the average request rate $\overline{\lambda} = 2$, (b) the popularity skew parameter α, and (c) the number of files N. (Default scenario with $N = 1,000$ files, Zipf popularity with $\alpha = 1$, average request rate $\overline{\lambda} = 2$, cloud storage cost $C = 0.25$ and cloud bandwidth cost factor $c = 0.5$.)

As for the inflation contributions (using the *bundling* and the *hybrid* policies), we consider four different sub-policies for determining which torrents to inflate.

- **Proportional:** With this policy, the inflation contributions are assigned proportionally to the current request load of each file, such that $\phi_i = \lambda_i(UT/L - 1)$.
- **Random (uniform):** With this policy, the inflation contributions are assigned uniformly among files. On average, each file hence obtains $\phi_i = \overline{\lambda}(UT/L - 1)$.
- **Basic:** With the *basic* policy we identify the set of files that would benefit from inflation (in that their delivery cost would reduce if an additional fraction $\overline{\lambda}(UT/L - 1)$ is allocated to that file). Assuming that there are $n \leq N$ such files, we then allocate $\phi_i = \frac{N}{n}\overline{\lambda}(UT/L - 1)$ of the peer inflation to these files (and none to the others).
- **Fine:** With the *fine* (grained) policy, the *basic* policy is first applied. Then a greedy search algorithm is used to incrementally improve the current inflation allocation. In each step, we identify the file with the biggest possible cost reduction and the file with the smallest possible cost increase if an inflation contribution ψ is moved from the second to the first. The value ψ is initially equal to the inflation contribution of the file with the smallest inflation contribution, but is decreased by a factor 2 each time that we do not find any further improvements, until we no longer can make improvements.

Referring to our system model, the above policies must all satisfy equations (2)-(5). For the *hybrid* policies, this requires that the total inflation $\sum_{i=1}^{N} \phi_i$ is adjusted such that it does not exceed the total upload contributions $\sum_{i=1}^{N} \lambda_i(UT/L - 1)$ minus the seed contributions $\sum_{i=1}^{N} B_i^p$. This is simply done by repeatedly solving the equations until the ϕ_i values converge.

Figure 6 shows a breakdown of the peer, cloud, and server bandwidth usage with our three baseline algorithms: *seeding*



(a) Seeding (only) (b) Bundling (only) (c) Hybrid

Fig. 6. Characterization of the optimal cloud, peer, and server usage for the three baseline policies: (a) seeding (only), (b) bundling (only) with the fine inflation allocation, and (c) hybrid with the fine inflation allocation. (Default scenario with $N = 1,000$ files, Zipf popularity with $\alpha = 1$, average request rate $\bar{\lambda} = 2$, cloud storage cost $C = 0.25$ and cloud bandwidth cost factor $c = 0.5$.)

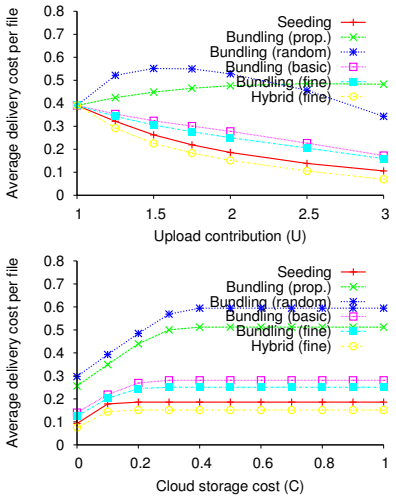


Fig. 7. Delivery cost comparisons of example policies when varying the upload rate and cloud costs. (Default scenario with $N = 1,000$ files, Zipf popularity with $\alpha = 1$, average request rate $\bar{\lambda} = 2$, cloud storage cost $C = 0.25$, cloud bandwidth cost factor $c = 0.5$, and peer upload rate $U = 2$.)

(only), *bundling (only)*, and the *hybrid* policy. Here, both the *bundling* and the *hybrid* policy use the *fine* allocation. We note that *peer seeding* does not at all affect the total system upload bandwidth required (as illustrated by the black curve in Figure 6(a)). In contrast, both the *bundling* and *hybrid* policies are able to significantly reduce this amount by inflating some of the more popular contents, such as to reduce their system upload bandwidth requirements. In addition, the *hybrid* policy is able to use peer seeding to further reduce the amount of server (or seed) bandwidth needed.

Finally, we note that all policies are able to (mostly) reduce the bandwidth requirements beyond the point where there is no benefit to push the contents to the cloud. The exception is the *bundling* policy, for which there is a small number of files for which there was not enough inflation contributions around to completely eliminate the cost benefit of the cloud. (See the blue region in Figure 6(b).)

Figure 7 shows the delivery cost comparisons of example policies when varying the upload rate and cloud costs. We note that the cloud costs must be small to help reduce the delivery costs, especially as most of these peer-based policies effectively can help reduce the need for significant server bandwidth resources. From Figure 7(a) we can also see that naive *bundling* policies may actually hurt the system, as the wrong torrents may get inflated. Careful *bundling*, however, can achieve close to the same cost reductions as *peer seeding*, while not requiring any long-term peer storage. Furthermore, we note that combining the two techniques (as with the *hybrid* policy) can further reduce the costs significantly.

While simple in nature, the *fine* inflation allocation (used by the *bundling* and *hybrid* policies) requires much more computation than the *basic* allocation. As the costs of the two approaches in general do not differ by more than 10%, the *basic* allocation may be beneficial in some contexts.

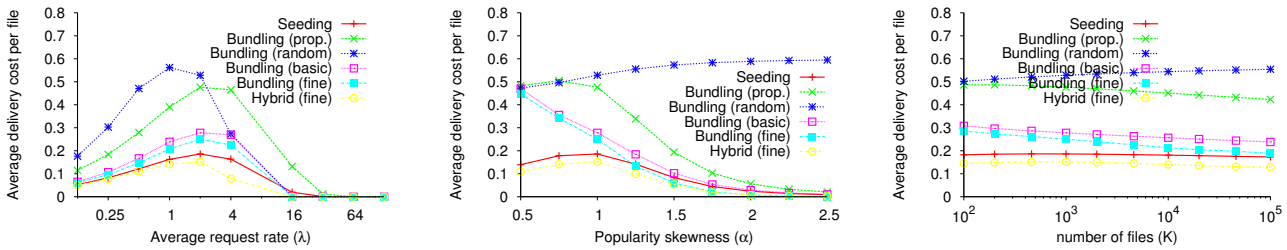
Figures 8 shows the delivery cost comparisons of example policies when varying the workload conditions. Again, these figures illustrate the importance of careful inflation selection, as well as a clear benefit to leveraging seeding when possible.

In summary, we find that much of the benefits of peer seeding can be achieved with careful torrent inflation. Hybrid policies that combine bundling and peer seeding perform best, often reducing the delivery costs by an extra 20% relative to only using seeding. Interestingly, the number of files that should be pushed to the cloud decreases as more advanced peer policies are employed.

V. Replication and Locality Awareness

Thus far we have considered the case when the content provider is charged the same bandwidth cost for all uploads, independent of the peers' locality. However, future charging models (of cloud bandwidth, for example) may also take into account the service locality. In this section we take a closer look at how the content provider can best reduce its costs in systems where it has an additional bandwidth cost for uploads to remote peers.

For this analysis, we use the full cost model, but focus on a single file (for which it has been determined that pushing the content to the cloud is beneficial). Again, the model captures the balance between large torrent sizes (for self-sustainability purposes) and locality of service. We must therefore determine how many copies should be pushed to the cloud, to which cloud replicas the content should be pushed, and which replica should be serving which peer.



(a) Average request rate (b) Popularity skew (c) Number of files

Fig. 8. Delivery cost comparisons of example policies when varying the workload conditions. (Default scenario with $N = 1,000$ files, Zipf popularity with $\alpha = 1$, average request rate $\bar{\lambda} = 2$, cloud storage cost $C = 0.25$, cloud bandwidth cost factor $c = 0.5$, and peer upload rate $U = 2$.)

We assume that the content provider has the choice to replicate the contents to any subset $\mathcal{P}_i \in \mathcal{R}$ of the $R = |\mathcal{R}|$ possible replica sites. As earlier, there is a storage cost C associated with each replica. However, in addition to the original bandwidth cost c there is an additional *remote access* cost q for any bandwidth that is associated with uploads to peers outside the region of that replica server.

While all analysis in this section applies to torrents with inflation, for simplicity, we consider the case when bundling is not utilized. Requests for the representative content i from the clients of each group j are assumed to be Poisson at rate $\lambda_{i,j}$, with the servers/groups indexed from 1 to R in non-increasing order of the group request rates. Ignoring the scaling c of the cloud bandwidth cost (as only cloud servers are considered), the total delivery cost for file i can be calculated as

$$\sum_{j \in \mathcal{P}_i} B_{i,j}(1 + q f_{i,j}) + |\mathcal{P}_i| C, \tag{6}$$

where $B_{i,j}$ is the replica upload bandwidth usage for file i at replica j, and $f_{i,j}$ is the fraction of this bandwidth that is to clients outside the local region of replica site j. Naturally, $B_i(\lambda_i) \leq \sum_{j=1}^{R} B_{i,j}$, where $\lambda_i = \sum_{j=1}^{R} \lambda_{i,j}$.

A. Baseline policies

To gain insights into the characteristics of good locality-aware system policies, as well as to garner an understanding of what aspects of such protocols may provide the biggest dividend, we next consider several baseline policies.

Local: The content is replicated to all R locations, and peers are served only by the local server and peers. With this policy there will be one swarm per replica site. Naturally, we have $B_{i,j} = B(\lambda_{i,j})$ and $f_{i,j} = 0, \forall j$.

Central: All peers are served using a single large swarm, with the replica with the highest request rate acting as the single helper. For this policy, we have $B_{i,1} = B_i(\lambda_i)$, and $B_{i,j} = 0$ for all other j $(2 \leq j \leq R)$. Furthermore, the remote access probability can be calculated as $f_{i,1} = \frac{1}{\lambda_i} \sum_{j=2}^{R} \lambda_{i,j}$.

All, locality aware: All peers are served using a single large swarm. However, in contrast to the *central* policy, the content is replicated to all R replica sites and whenever content must be served (to ensure self-sustainability) one of the replicas with local peers are selected to make the upload(s). For this policy, we have $\sum_{j \in \mathcal{R}} B_{i,j} = B(\lambda_i)$, and $f_{i,j} = 0, \forall j$.

Single, locality aware: Similar to the *central* policy, all peers are served using a single large swarm, with the replica with the highest request rate acting as the single helper. However, the replica only serves remote peers if there are no local peers in the swarm. For this policy, $\sum_{j \in \mathcal{R}} B_{i,j} = B(\lambda_i)$. We approximate $f_{i,j}$ using the probability that there is no local peer (at replica 1) whenever the swarm would require pieces to be uploaded using an upload policy that uploads whenever there are peers in the swarm. For such policy the fraction served remotely is $\frac{e^{-\lambda_i T(1-p)} - e^{-\lambda_i T}}{1 - e^{-\lambda_i T}}$, where $p = \frac{\lambda_{i,1}}{\lambda_i}$.

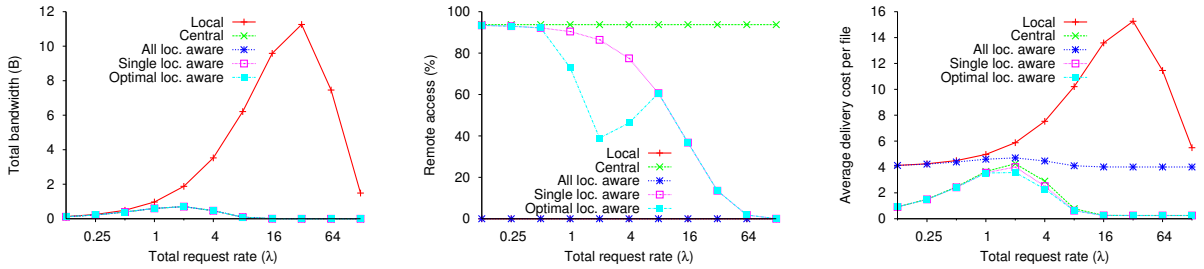
Optimal, locality aware: As with the above locality-aware policies, all peers are served using a single large swarm, and the replica servers with a copy give priority to local peers, when service is needed. To achieve optimality, the content is replicated to the P_i^* $(1 \leq P_i^* \leq R)$ replica sites with the highest local load, where P_i^* is selected such that the overall cost function is the smallest among the R candidate allocations. For each of the R candidates, we calculate the fraction that is served remotely as $\frac{e^{-\lambda_i T(1-p)} - e^{-\lambda_i T}}{1 - e^{-\lambda_i T}}$, where $p = \frac{\sum_{j=1}^{P_i} \lambda_{i,j}}{\lambda_i}$.

We note that the first two policies do not require peers or servers to take into account locality. In contrast, the three locality-aware policies require (at least) the servers to keep track of which peers are local and which are remote, as well as (in the case of the two last policies) determine which server out of the replica servers with local peers should be responsible for sustaining the torrent.

B. Policy evaluation

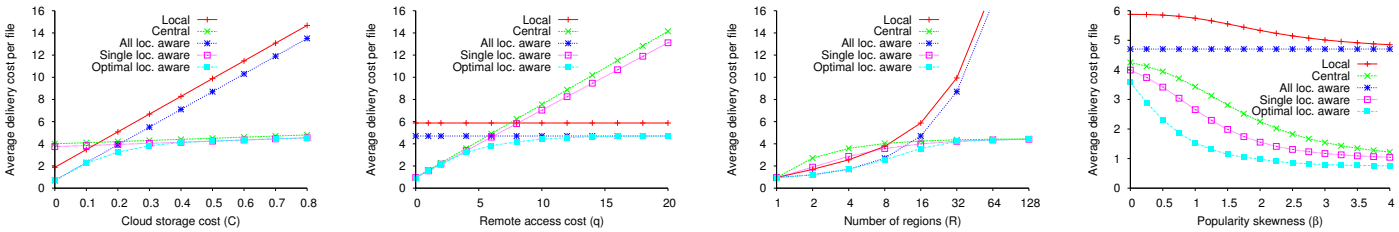
We first consider the cost components for the different policies separately. Figure 9 shows that all policies except the *local* policy always minimize the replica server bandwidth usage (Figure 9(a)), but that the fraction of remote service may differ significantly between the policies (Figure 9(b)). These differences, can also be observed in the total delivery costs (Figure 9(c)). Naturally, the *optimal locality-aware* policy does the best. However, it is interesting to note that the *central* policy and *single, locality aware* policy does very well too.

As the *central* policy is significantly easier to implement than the other policies, we wanted to see how the policies perform under a wider range of scenarios. We conclude this section with a comparison of the delivery costs of our baseline policies, for a range of scenarios. Figure 10 shows the delivery



(a) Server bandwidth (b) Remote access (c) Total delivery cost

Fig. 9. Replication analysis under varying request rates. Default scenario: $R = 16$, $\beta = 0$ (i.e., $\lambda_{i,j} = \frac{\lambda_i}{R}$), $C = 0.5$, $q = 5$, L, $U = 1$.



(a) Storage cost (C) (b) Remote access cost (q) (c) Number of regions (R) (d) Load skewness (β)

Fig. 10. Policy comparison under varying workload conditions. Default scenario: $R = 16$, $\beta = 0$, $\lambda_i = 2$, $C = 0.5$, $q = 5$, L, $U = 1$.

costs, as we vary the cloud storage cost C, the remote access cost q, the number of replica regions R, and the load skew β.

While our default scenario has uniform load across the replica sites (i.e., $\beta = 0$), we note that the relative differences between the costs of the policies can be fairly high when the load is skewed (higher β). As this is likely the case in many real systems, we note that carefully designed locality-aware policies are important for content providers wanting to minimize their delivery costs in such systems.

We note that both simpler policies (i.e., *central* and *local*) do well for particular regions of the parameter space, but poorly in other regions. In fact, only the *optimal locality-aware* policy does well for all regions. This suggests that careful system design may be important, especially in environments in which the pricing of cloud-based resources is such that it may provide cost savings for local service.

VI. Related works

The dynamics of the number of downloaders and uploaders in BitTorrent-like systems were modeled using stochastic models [15] and fluid models [16]. In [17] and [18] these models were extended to analyze the impact of server and cache upload bandwidth on the system dynamics, respectively. In contrast to these works, our focus is on the server upload bandwidth needed in peer-to-peer file sharing systems with service guarantees.

The impact of bundling different contents on the peers' download times and on the lifetime of the swarms was considered in [7], [8], [19]. These works showed that various static and dynamic bundling strategies can extend the swarm lifetime significantly. Cross-swarm seeding was considered in [13], [20] in order to leverage the unused upload bandwidth of popular swarms to help the download in less popular swarms. In contrast to these works, we consider how bundling and cross-swarm seeding can be used most effectively to reduce the server upload bandwidth requirements.

A stochastic fluid model of the server bandwidth needed for video-on-demand (VoD) systems was provided in [21]. Upper bounds on the minimum required server upload bandwidth have also been proposed [22]. The server bandwidth needed for P2P live streaming was analyzed in [23] using a stochastic fluid model and simulations. Frameworks for cloud-assisted P2P streaming to accommodate time varying demands were considered for live streaming in [24] and for VoD in [25]. In VoD and live streaming systems chunk delivery needs to be approximately in-order, unlike in the case of the content distribution systems we address in this paper, where the order of chunk delivery is unrestricted. Apart from this difference, the consideration of cloud storage and bandwidth costs together with bundling and cross-swarm seeding distinguish our work from the above papers.

VII. Conclusions

In this work we considered the problem of peer-assisted content delivery with service time guarantees. We developed a simple model of the upload bandwidth requirements as a function of the content popularity, and showed that it is the moderately popular contents that require most upload bandwidth. We used the model to give insight into the optimal combination of server and cloud bandwidth resources with the aim of decreasing the total bandwidth cost. We then considered torrent inflation and cross-swarm seeding, and showed that a hybrid policy can significantly decrease the bandwidth requirements. Finally, we addressed the case of several cloud providers and showed that optimized cloud replica placement and a good locality policy can lead to significant cost savings. We believe that the insights provided using our simple model will be valuable for developers of future systems.

VIII. Acknowledgements

We thank the anonymous reviewers. This work was supported by National ICT Australia (NICTA), the Natural Sciences and Engineering Research Council (NSERC) of Canada, the ACCESS Linnaeus Centre at KTH, and CENIIT at Linköping University.

References

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, Mar. 1999.

[2] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proc. ACM SOSP*, Oct. 2003.

[3] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti, "Characterizing Web-based video sharing workloads," *ACM Transactions on the Web*, vol. 5, pp. 8:1–8:27, May 2011.

[4] G. Dán and N. Carlsson, "Power-law revisited: A large scale measurement study of P2P content popularity," in *Proc. IPTPS*, Apr. 2010.

[5] ——, "Dynamic swarm management for improved bittorrent performance," in *Proc. IPTPS*, Apr. 2009.

[6] D.Menasche, A. Rocha, E. Silva, R. Leao, D. Towsley, and A. Venkataramani, "Estimating self-sustainability in peer-to-peer swarming systems," in *Proc. IFIP Performance*, Nov. 2010.

[7] D. Menasche, A. Rocha, B. Li, D. Towsley, and A. Venkataramani, "Content availability and bundling in swarming systems," in *Proc. ACM CoNEXT*, Dec. 2009.

[8] N. Carlsson, D. L. Eager, and A. Mahanti, "Using torrent inflation to efficiently serve the long tail in peer-assisted content delivery systems," in *Proc. IFIP/TC6 Networking*, May 2010.

[9] S. Zhang, N. Carlsson, D. Eager, Z. Li, and A. Mahanti, "Dynamic file bundling for large-scale content distribution," in *Proc. IEEE LCN*, Oct. 2012.

[10] J. Han, S. Kim, T. Chung, T. T. Kwon, H. chul Kim, and Y. Choi, "Bundling practice in bittorrent: What, how, and why," in *Proc. ACM SIGMETRICS/Performance*, June 2012.

[11] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee, "Bittorrent is an auction: Analyzing and improving bittorrent's incentives," in *Proc. ACM SIGCOMM*, Aug. 2008.

[12] N. Carlsson and D. Eager, "Modeling priority-based incentive policies for peer-assisted content delivery systems," in *Proc. IFIP/TC6 Networking*, May 2008.

[13] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurement, analysis, and modeling of bittorrent-like systems," in *Proc. ACM IMC*, Oct. 2005.

[14] N. Carlsson and D. L. Eager, "Peer-assisted on-demand streaming of stored media using bittorrent-like protocols," in *Proc. IFIP/TC6 Networking*, May 2007.

[15] X. Yang and G. de Veciana, "Service capacity of peer-to-peer networks," in *Proc. IEEE INFOCOM*, Mar. 2004.

[16] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proc. ACM SIGCOMM*, Aug/Sept. 2004.

[17] Y. Sun, F. Liu, B. Li, and B. Li, "Peer-assisted online storage and distribution: modeling and server strategies," in *Proc. ACM NOSSDAV*, June 2009.

[18] F. Lehrieder, G. Dán, T. Hossfeld, S. Oechsner, and V. Singeorzan, "The impact of caching on bittorrent-like peer-to-peer systems," in *Proc. IEEE P2P*, Aug. 2010.

[19] N. Lev-tov, N. Carlsson, Z. Li, C. Williamson, and S. Zhang, "Dynamic file-selection policies for bundling in bittorrent-like systems," in *Proc. IEEE IWQoS*, June 2010.

[20] R. S. Peterson and E. G. Sirer, "Antfarm: Efficient content distribution with managed swarms," in *Proc. NSDI*, May 2009.

[21] D. Ciullo, V. Martina, M. Garetto, E. Leonardi, and G. Torrisi, "Stochastic analysis of self-sustainability in peer-assisted vod systems," in *Proc. IEEE INFOCOM*, Mar. 2012.

[22] D. Ciullo, V. Martina, M. Garetto, E. Leonardi, and G. L. Torrisi, "Performance analysis of non-stationary peer-assisted vod systems," in *Proc. IEEE INFOCOM*, Mar. 2012.

[23] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for p2p streaming systems," in *Proc. IEEE INFOCOM*, May 2007.

[24] F. Wang, J. Liu, and M. Chen, "CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proc. IEEE INFOCOM*, Mar. 2012.

[25] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. IEEE INFOCOM*, Mar. 2012.

Appendix

We derive two simple lower bounds. First, a *lower bound* of all server upload policies can be obtained by assuming infinite upload bandwidth and a schedule where each peer uploads the pieces in the last possible moment before leaving the system (if needed). In this case, the server must only upload the file data L one time per busy (or idle) period of an $M/G/\infty$ queue with service time L/U. This results in the following lower bound expression

$$\lambda L e^{-\lambda L/U}. \tag{7}$$

This expression also corresponds to that of a server policy that uploads whenever there is exactly one peer in the system.

A second *lower bound* can be obtained as follows. Assume that peers can upload at rate U (at most) and the download rate D is sufficiently large to not be a constraint. Furthermore, assume that each download takes exactly $T = L/U$, and order all downloads in a request sequence based on their deadlines t_i, from earliest to last download ($1 \le i \le K$).

Now, at the end of each deadline t_i we can look at the largest possible amount of the file that can be left in the system after peer $(i-1)$ left the system (at time t_{i-1}). Let's denote this amount by x_i. Clearly, if this amount is less than the file size L, the difference must be scheduled by deadline t_i. Hence the total server bandwidth can be bounded by: $\sum_{i=1}^{\infty} s_i$, where $s_i = L - \min[L, x_i]$ is the minimum amount that the server must deliver that it had not previously scheduled by deadline t_{i-1}. Furthermore, the amount x_i can be calculated as:

$$x_i = \sum_{j=1}^{i-1} \max[0, U(t_j - (t_i - T))] + \sum_{j=1}^{i-1} \min\left[s_j, \sum_{k=i}^{K} \max[0, \min[U(t_j - (t_k - T)), U(t_i - t_j)]] \right] \tag{8}$$

The first sum of terms corresponds to the amount the peer could have received directly from older peers. The second sum of terms corresponds to the amount that younger peer k could have relayed (from the server) to an older peer j, and later shared with peer i (after the completion of j). Note that such a peer k at most could relay $\max[0, U(t_j - (t_k - T))]$ to peer j, and this peer at most could upload $\max[0, U(t_i - t_j)]$ out of this data to peer i before peer i's deadline. Furthermore, note that $x_i = 0$ and $s_i = L$ for any peer (including the first peer $i = 1$) that arrive to the system when it is empty. For such a peer $t_j < t_i - T$ and $t_j < t_k - T$, for all $k > i$.

Finally, we note that a tighter (combined) bound could be obtained by taking the maximum of the above two lower bounds. Again, for the purpose of our analysis the approximation is sufficient.