Exploring the Optimal Replication Strategy in P2P-VoD Systems: Characterization and Evaluation

Weijie Wu, *Student Member, IEEE,* and John C.S. Lui, *Fellow, IEEE*

Abstract—P2P-Video-on-Demand (P2P-VoD) is a popular Internet service which aims to provide a scalable and high quality service to users. At the same time, content providers of P2P-VoD services also need to make sure that the service is operated with a manageable operating cost. Given the volume-based charging model by ISPs, P2P-VoD content providers would like to reduce peers' access to the content server so as to reduce the operating cost. In this paper, we address an important open problem: what is the "*optimal replication ratio*" in a P2P-VoD system such that peers will receive service from each other and at the same time, reduce the access to the content server? We address two fundamental issues: (a) what is the optimal replication ratio of a movie if we know its popularity, and (b) how to achieve these optimal ratios in a distributed and dynamic fashion. We first formally show how movie popularities can impact server's workload, and formulate the video replication as an optimization problem. We show that the conventional wisdom of using the proportional replication strategy is "*sub-optimal*", and expand the design space to both "*passive replacement policy*" and "*active push policy*" to achieve the optimal replication ratios. We consider practical implementation issues, evaluate the performance of P2P-VoD systems and show how to greatly reduce server's workload and improve streaming quality via our distributed algorithms.

Index Terms—Distributed applications, protocols, modeling, performance

1 Introduction

In the past few years, we have seen a growing interest of using the peer-to-peer (P2P) technology to deliver video-on-demand (VoD) services. There are already a number of successful P2P-VoD services offer by companies like PPLive, PPStream and UUSee. The advantage of using this technology is that the system can utilize peers' resources to satisfy other peers' viewing need, and thus to achieve a more scalable system as compared to the traditional client-server architecture. The main difference between P2P-VoD systems and P2P live streaming systems is that peers in P2P-VoD service need to contribute a *much larger* local cache space, e.g., physical memory and local storage so as to cache any multimedia data, as well as their upload bandwidth to transfer data among peers. For example, in PPLive system [6], each peer needs to dedicate 1 GB of local storage for the P2P-VoD service. This local storage is for caching some *previously watched movies* so that in case a new peer joins the system, this peer does not need to download from the VoD content server but rather, obtains the data directly from other peers. This unique feature not only frees up the VoD content server to serve other peers, but more importantly, reduces the upload bandwidth traffic of P2P-VoD content providers. Since content providers usually pay ISPs based on the *volume-based charging* scheme, it is to the best interest of a P2P-VoD content provider to reduce the server's upload traffic, and a proper caching policy can surely achieve such goal. Given the distributed cache spaces of all peers in a P2P-VoD system, the aim of our paper is to address how to utilize these resource efficiently so as to reduce the operating cost of a P2P-VoD content provider. Specifically, given the caching and upload capacities of peers, can one characterize the "*optimal caching policy*" so as to minimize the traffic to the P2P-VoD content server?

Different from the P2P file sharing applications, to support peers in a P2P-VoD service, one has to guarantee (1) a low start-up latency in accessing the movies; (2) to support the required playback-rate for all peers throughout their viewing sessions. To achieve these goals, a good replication policy should replicate enough replicas so as to support the required workload. To illustrate, consider a P2P-VoD system with 100 peers and each peer offers one cache unit (i.e., let's say each cache unit can store one movie only). There are two movies M_1, M_2 with the same viewing popularity, i.e., each with 50 viewers. Assume a replication policy keeps 99 replicas for M_1 but only one replica for M_2. Then, requests for M_2 will fail to find sufficient peers' support and have to access the P2P-VoD server. The system is caching excessive number of M_1 replicas, furthermore, the system cannot take advantage of the upload bandwidth of those peers which cache M_1.

It is challenging to characterize the optimal caching policy in a P2P-VoD system, especially when peers have different upload capacities and movies have different viewing popularities. These factors make designing the optimal replication policy an open problem so far. In this paper, we address the following technical issues:

• *The authors are with Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.*
E-mail: {wjwu,cslui}@cse.cuhk.edu.hk

- Given movies' viewing popularities, what is the *optimal* ratio of total storage space that should be dedicated to each movie such that when a new peer arrives, it is most likely to be served by other peers?
- Given the system dynamics (e.g., peers' churn), how to design distributed and adaptive algorithms to achieve the optimal replication ratios?

It is interesting to note that previous work [6] and conventional wisdom suggested using the *proportional replication strategy*, i.e., replicating movies proportionally to their popularity. However, authors in [17] reported a contradicting result that the proportional replication may have poor performance, especially for unpopular movies. In this work, we give the formal justifications on this contradiction and provide insights to the characterization of the optimal replication ratios. These insights provide us the principles in designing practical and effective replication algorithms. Our contributions are:

- We show that proportional replication is not the optimal strategy. Instead, one should replicate movies proportionally to the *deficit bandwidth* (we will formally define this concept in Sec. 2.4). The optimal replication strategy should be more greedy for unpopular movies.
- We expand the design space by considering both *passive* and *active* replication. We propose a passive replacement algorithm to decide which movie to purge when a local storage is full; we also propose an active replication algorithm to aggressively push data to peers so as to achieve the desired replication ratios. We evaluate our proposed algorithms and show significant improvement of video quality and large reduction on server's workload.

This paper is extended from our earlier work [20]. The outline is as follows. In Section 2 we present a mathematical model for P2P-VoD system and give an optimization framework for the replication problem. In Section 3 and 4, we discuss the optimal replication ratios under the software and set-top box deployment scenarios, respectively. In Section 5, we present our passive replacement and active push algorithm to achieve the optimal replication ratios. In Section 6, we consider several important practical issues in implementation and present the performance results of our algorithms. Section 7 states the related work and Section 8 concludes.

2 System Architecture and Model

2.1 System Architecture

We first briefly describe a general architecture of a P2P-VoD system, which mainly consists of (a) content servers, (b) trackers and (c) peers. The content servers are the source of content and can upload data to requesting peers. Each peer seeks to download data from the system, and in the meanwhile, it reports to the tracker the data it caches in the local storage so as to provide upload service to other peers. Each video file is divided into small pieces called chunks. When a peer joins the system, a local tracker recommends it a set of other peers for downloading its required chunks. This peer exchanges the buffer map with the recommended peers so that it can determine the sources to download the required chunks from. Interested readers may refer to [6] for a detailed description.

In general, there are two types of deployments: software based and set-top box based. They share the similar architecture as described above, but differ in the following. In a software deployment like PPLive or PPstream, the network is unstructured and regarded as full mesh. A peer usually contributes to others only when it is online watching a movie. The system operator cannot control any peer to be online or offline. On the other hand, an emerging trend is to install set-top boxes (with hard-disk inside) at user's homes and they form a P2P network. This network is usually fully connected within a local area so that a single tracker schedules the upload/download among peers. The operator can control the whole network where peers are always "*online*" even though they are not watching any movie.

Formally, we define *active peers* to be those currently watching a movie and at the same time, providing upload service to other peers, while *inactive peers* are those currently not watching any movie but staying online and contributing their upload bandwidth to other peers. Hence, a software based system only consists of active peers, while a set-top box deployment may involve both active and inactive peers.

In general, VoD includes various quality of services. Supporting Standard Definition (e.g., 2Mbps) or High Definition (e.g., 4Mbps) can be quite challenging due to the high bandwidth requirement. Some VoD companies restrict these services to a limited amount of premium users. Some movies are of low resolution and can be easily supported by other peers. Replication mainly helps in supporting the movies with medium playback rates (e.g., 500kbps). P2P-VoD companies may also use other techniques to partially support movies with extremely high or low popularities (e.g. broadcast for very hot movies and unicast for very cold movies). For most movies with popularity in the middle range, replication strategy needs to be carefully designed so as to alleviate the server's workload.

2.2 System Model

We consider a P2P-VoD system $\mathcal{G} = \{\mathcal{S}, \mathcal{P}, \mathcal{M}\}$, where $\mathcal{S}$ is the logical P2P-VoD content server of the system[1], $\mathcal{P} = \{P_1, ..., P_N\}$ is the set of peers, and $\mathcal{M} = \{M_1, \ldots, M_K\}$ is the set of movies stored in $\mathcal{S}$. Let $\mathcal{P}_\mathcal{A}$ represent the set of active peers, and let $\mathcal{P}_k$ represent the set of active peers which are watching movie M_k. We have $\mathcal{P}_k \subseteq \mathcal{P}_\mathcal{A}$. Similarly, denote $\mathcal{P}_\mathcal{I}$ as the set of inactive peers. Let $\widetilde{u}_j$ be the maximal upload rate (i.e., upload capacity) of P_j.

1. A logical P2P-VoD content server may consist of many physical servers which process any request from peers in a P2P-VoD system.

Peers which watch movie M_k can access the data from (a) $\mathcal{S}$, or (b) any other peer which is watching M_k, or (c) peers which have watched and cached M_k.

The upload bandwidth consumption of $\mathcal{S}$ is affected by (a) the peer scheduling policies, and (b) movie replication strategies. For the ease of presentation, we decompose these two design problems and use a realistic and practical peer scheduling strategy: a peer first requests data from other peers, and only when other peers cannot supply sufficient bandwidth, then this peer requests data from $\mathcal{S}$. In here, we focus on the movie replication strategy and show how it can significantly impact the upload bandwidth consumption of $\mathcal{S}$. Let us first state the general P2P-VoD setting, then we give a general model to show how replication strategy can affect the upload bandwidth consumption of $\mathcal{S}$.

2.3 Peer Scheduling Policy

A peer scheduling strategy includes two design issues: (1) from a downloader's viewpoint, it needs to decide which peers to request the data from; (2) from an uploader's viewpoint, it needs to decide which peers to upload the data to when it receives multiple requests.

Let us first describe how an active peer downloads a movie. Suppose a peer wants to watch movie M_k. In general, its request can be supported by three different sources: (a) the server $\mathcal{S}$; (b) active peers which are currently watching the same movie M_k, and we denote these as the *concurrent peers*; (c) peers which are not watching movie M_k but have stored M_k at their local cache, and we denote these as the *replication peers*. Note that a replication peer may be an active peer watching another movie; or an inactive peer which has movie M_k in its cache. Lastly, in order to have a short start-up latency and good streaming quality, each peer needs to download the movie at its playback rate r.

In real systems, peers may perform various operations like fast forward, rewind and replay during the viewing process. In this paper, we first consider a simplified *sequential viewing* model, where peers watch the movies from the beginning to the end sequentially. We use this simplification to get a neat model and show the benchmark performance of our replication algorithms. In fact, we only need some simple modifications so as to adapt to the peer operations, which we will discuss later. We consider the following practical peer scheduling policy: a peer first seeks help from concurrent peers; if the playback rate cannot be satisfied, the peer seeks help from replication peers; if the playback rate still cannot be satisfied, the server $\mathcal{S}$ will upload the data to this peer so as to satisfy the playback rate.

We would like to comment on the rationale of this scheduling policy. We let $\mathcal{S}$ be the last requesting target of a peer in order to reduce the upload consumption of $\mathcal{S}$ so as to reduce the operating cost of the P2P-VoD system. Note that, in real systems, a peer may cache only *part* but not the whole movie (details are discussed in Section 6). Therefore, requesting from a replication peer may result in frequent rescheduling: a peer P_i that requests data from a replication peer P_j needs to look up other peers for support when P_i's viewing part is not cached in P_j. Concurrent peers may also perform various operations like fast forward, rewind, replay or even depart, however, a typical peer is more likely to stick to a movie it is interested in, so it is more likely that P_i can download from one particular concurrent peer P_k throughout the watching period. This is why we prefer concurrent peers over replication peers to support the viewing requirement. We note that in practical systems, there might be complicated considerations on scheduling. Our model is not restricted in this preference; in fact, it can be easily extended to other scheduling policies.

There can be various scheduling policies regarding how peer P_i should seek help from its concurrent peers. We would like to point out that our model is applicable to any policy that satisfies the following two properties: (a) *sequentiality*: only the concurrent peers that arrived *earlier* than P_i can assist in the data upload for P_i; and (b) *greediness*: P_i downloads from earlier-arrived concurrent peers as much as possible but not exceeding its required playback rate; peers arriving later than P_i will not be reserved for uploading unless P_i's viewing requirement is totally satisfied by the concurrent peers. Note that the sequentiality property should be satisfied in any P2P-VoD system due to the asynchronous nature of VoD system. The greediness property is needed because it ensures the maximum efficiency of cooperation among concurrent peers within a swarm, which we will show in later section. We call a scheduling policy that satisfies the greediness property as *greedy* policy. Note that in a P2P-VoD system, a greedy policy allows a peer to download from earlier arrived concurrent peers in any manner, so the scheduling policy can be easily implemented.

From an uploader's viewpoint, it may receive multiple download requests simultaneously. In our work, an active peer will serve its concurrent peers with a higher priority. Only when its upload bandwidth is not fully consumed by concurrent peers, the active peer will provide the remaining upload bandwidth to other peers. The rationale of the priority follows the same reasoning stated above. For an inactive peer, it will upload to those peers watching any movie that is stored in its local cache. The assumption is not only for mathematical tractability, but this simple scheduling rule can also maximize the number of peers to receive data at the playback rate r.

2.4 Model of Movie Popularity on Server's Workload

Let us illustrate how the movie popularity can affect the server's upload bandwidth. First, peer P_i's download rate can be supported partly by concurrent peers, partly by replication peers, and partly by the server $\mathcal{S}$. Let $\widetilde{d}_i^c$ and $\widetilde{d}_i^r$ be the total download rates supported by concurrent peers and replication peers, respectively. If $\widetilde{d}_i^c+\widetilde{d}_i^r$ is below the playback rate r, the server needs to fill

up the gap $r-\tilde{d}_i^c-\tilde{d}_i^r$ so as to satisfy the QoS guarantee. For the ease of presentation, we denote $\widetilde{d}_i^s = r-\widetilde{d}_i^c-\widetilde{d}_i^r$ ($\forall i \in \mathcal{P}_A$) as the filling upload rate from the server to P_i. The expected upload bandwidth consumption of the server $\mathcal{S}$ is denoted as U, which can be expressed as:

$$U = E\left(\sum_{i\in\mathcal{P}_A}\widetilde{d}_i^s\right) = E\left(\sum_{k\in\mathcal{M}}\sum_{i\in\mathcal{P}_k}\widetilde{d}_i^s\right). \quad (1)$$

Based on the peer scheduling policy above, we have:

Lemma *1:* In a large scale P2P-VoD system, the expected upload consumption at the server is approximately

$$U \approx \sum_{k\in\mathcal{M}}\left[E\left(\sum_{i\in\mathcal{P}_k}(r-\widetilde{d}_i^c)\right)-\mathcal{R}_k\right]^+, \quad (2)$$

where $\mathcal{R}_k$ is the maximal upload bandwidth from all replication peers that can contribute to M_k, or $\sum_{i\in\mathcal{P}_k} d_i^r \le \mathcal{R}_k$.

Due to page limit, we omit all the proofs. Interested readers may refer to the supplementary file for details. Let us define the following important notation.

Definition *1:* For movie M_k, we define *deficit bandwidth* as

$$\widetilde{\mathcal{D}}_k(n_k) = \sum_{i\in\mathcal{P}_k}(r-\widetilde{d}_i^c) = \sum_{i\in\mathcal{P}_k}(\widetilde{d}_i^r+\widetilde{d}_i^s), \quad (3)$$

and let

$$\mathcal{D}_k(n_k) = E\left[\widetilde{\mathcal{D}}_k(n_k)\right] \quad (4)$$

be the *expected deficit bandwidth* of having n_k concurrent peers in $\mathcal{P}_k$, i.e., the bandwidth gap between the total required playback rates and the total download rates that can be supported by n_k concurrent peers. We can now rewrite Eq. (2) and directly obtain the following lemma:

Lemma *2:* Given the number of concurrent peers, the expected upload consumption at the server is

$$U = \sum_{k\in\mathcal{M}}(\mathcal{D}_k(n_k)-\mathcal{R}_k)^+. \quad (5)$$

Our objective is to find a replication strategy, which determines $\mathcal{R}_k$, such that U, the expected upload bandwidth consumption of the server $\mathcal{S}$, can be minimized. Thus, we want to explore how the popularity of a movie may affect $\mathcal{D}_k(n_k)$, which in turn will affect the choice of replication. First we state the following lemma:

Lemma *3:* Under sequentiality requirement, any *greedy* scheduling policy minimizes the deficit bandwidth.

Lemma 3 highlights the condition that concurrent peers can cooperate most effectively. We state an important lemma to calculate this minimum deficit bandwidth.

Lemma *4:* Using a scheduling policy that satisfies the sequentiality and greediness properties, deficit bandwidth can be iteratively calculated by

$$\widetilde{\mathcal{D}}_k(n_k) = n_k r - \sum_{i\in\mathcal{P}_k}\widetilde{d}_i^c, \quad (6)$$

$$\widetilde{d}_i^c = \begin{cases} 0, & i=1; \\ \min\left(\sum_{j=1}^{i-1}(\widetilde{u}_j-\widetilde{d}_j^c), r\right), & i\ge 2. \end{cases} \quad (7)$$

	P_1	P_2	P_3	P_4	P_5	P_6
u_i (kbps)	500	800	200	800	300	1000
$\widetilde{d}_i^c$ (kbps)	0	500	600	400	600	500
$\widetilde{\mathcal{D}}_k(6)$ (kbps)	1000					

TABLE 1: Example for calculating $\widetilde{d}_i^k$ and $\widetilde{\mathcal{D}}_k(n_k)$

Let us use an example to illustrate the above framework in calculating $\widetilde{d}_i^c$ and $\widetilde{\mathcal{D}}_k(n_k)$. Assume six peers are watching movie M_k. These peers are $P_1,\ldots,P_6$, with P_1 arriving the earliest. We fix the movie playback rate at $r = 600$ kbps. If we know their upload capacities $\widetilde{u}_i$ as illustrated in Tab. 1, we can calculate $\widetilde{d}_1^c,\ldots,\widetilde{d}_6^c$ respectively, and by summing up all $(r-\widetilde{d}_i^c)$ we can get the deficit bandwidth, or $\widetilde{\mathcal{D}}_k(6) = 1000$ kbps. This $\widetilde{\mathcal{D}}_k(6)$ represents the bandwidth required to support all these six peers. Referring to Eq. (3), if there is no replication (or $\widetilde{d}_i^r = 0$), this is exactly the total upload bandwidth that the server $\mathcal{S}$ needs to provide.

In summary, given each peer's upload capacity $\widetilde{u}_i$, one can determine $\widetilde{d}_i^c$ and $\widetilde{\mathcal{D}}_k(n_k)$. Furthermore, given the upload capacity distribution of $\mathcal{P}_k$, we may determine $\mathcal{D}_k(n_k)$ by taking the expectation. Based on this framework, we can show how movie's popularity impacts the upload bandwidth consumption of the server $\mathcal{S}$. In later sections, we will show some interesting implications.

2.5 Discussion on Peer Operations

We have considered the "sequential viewing" model in the above analysis. In practice, the peers may perform various operations (e.g., fast forward, rewind or replay) when they watch the movies. In order to adapt to these practical issues, we propose a revision on the system design with a limited overhead on the trackers. Each peer reports to the tracker the chunk it needs downloading as well as the largest chunk index it has already downloaded. Denote by c_d and c_l the indices of these two chunks. Note that c_d might be smaller than c_l due to the fast forward operation, and that the set of chunks a peer caches is a subset of $\{1,2,\ldots,c_l\}$ since peers may skip some parts in the middle.

We use a simple model to capture the feature. When a peer downloads the chunk c_d of a movie, the tracker recommends a set of other peers with $c_l \ge c_d$. Assume that a recommended peer has a probability β to have cached this chunk. In general, $\beta < 1$ since it is possible to have skipped this chunk before. However, β may not be too small since a typical peer caches a large part, if not all, of the movie it watches.

When considering peer operations, a greedy scheduling policy may not minimize the deficit workload; however, if the fast forward operations are rare (comparing to rewind and replay operations), a greedy scheduling policy is near to optimal. In later sections, we will compute the deficit workload using this revised model.

2.6 Model of Replication on Server's Workload

Let us show how replication can help to reduce the server's workload. Intuitively, without using replication

(or when $\widetilde{d}_i^r = 0$), $\mathcal{D}_k(n_k)$ will be the upload bandwidth consumption of server $\mathcal{S}$ for movie M_k. Replication helps to reduce server's upload bandwidth consumption since peers could use replicas to partially support the deficit bandwidth $\mathcal{D}_k(n_k)$. For the ease of presentation, we have the following assumptions:

- We consider the system $\mathcal{G} = \{\mathcal{S}, \mathcal{P}, \mathcal{M}\}$, where $\mathcal{P} = \{P_1, ..., P_N\}$, $\mathcal{M} = \{M_1, ..., M_K\}$, with αN out of the N peers are active. An active peer wants to watch movie M_k with probability ρ_k, with $\sum_{k=1}^{K} \rho_k = 1$.
- Each peer, either active or inactive, caches one movie that it has watched before[2].
- Peers' upload and download scheduling policies follow the properties in Section 2.3.

To model the distribution of number of active peers in each movie, one can use a closed queueing network to represent the P2P-VoD system. Let random variable $\widetilde{n}_k$ denote the number of peers watching movie M_k. Similar to the work in [18], we can express the probability for movie M_k having n_k viewers, for $k = 1, 2, \ldots, K$, as:

$$P(\widetilde{n}_1 = n_1, \ldots, \widetilde{n}_K = n_K) = (\alpha N)! \frac{\rho_1^{n_1}}{n_1!} \ldots \frac{\rho_K^{n_K}}{n_K!}, \quad (8)$$

with $\sum_{k=1}^{K} n_k = \alpha N$. Here, ρ_i implies the relative popularity of movie M_i. Based on this queueing model, we have the following proposition:

Proposition *1:* The average upload consumption of $\mathcal{S}$ is

$$\overline{U}(\mathbf{q}) = \sum_{\sum n_k = \alpha N} \Bigg\{ (\alpha N)! \frac{\rho_1^{n_1}}{n_1!} \ldots \frac{\rho_K^{n_K}}{n_K!} \times \sum_{k \in \mathcal{M}} \Bigg[\mathcal{D}_k(n_k) - q_k \sum_{l \in \mathcal{M}} (C_l^u(n_l) + \mathcal{D}_l(n_l) - n_l r) - q_k C_{\mathcal{I}}^u((1-\alpha)N) \Bigg]^+ \Bigg\}, \quad (9)$$

where q_k is the fraction of cache space dedicated to M_k, $C_l^u(n_l) = E\left(\sum_{i \in \mathcal{P}_l} \widetilde{u}_i\right)$, $C_{\mathcal{I}}^u(n_I) = E\left(\sum_{i \in \mathcal{P}_{\mathcal{I}}} \widetilde{u}_i\right)$.

Now the model of the *optimal replication strategy* can be formulated as follows. We seek to find a set of replication ratios $\mathbf{q}$ which satisfies:

$$\begin{aligned} \min_{\mathbf{q}} \quad & \overline{U}(\mathbf{q}) \\ \text{subject to} \quad & \mathbf{q}^T \cdot \mathbf{1} = 1. \end{aligned} \quad (10)$$

In other words, we want to determine the number of replicas for each movie $M_k \in \mathcal{M}$, so that the average upload bandwidth consumption of $\mathcal{S}$ can be minimized.

In general it is difficult to get a close form solution for this optimization problem. However, as we will show in the following section, when we consider some common deployment cases, one can get some interesting implications and insights in designing replication strategies.

2. In general, a peer may also cache multiple movies. In such systems, the optimal caching policy is coupled with the optimal scheduling strategy and is hence hard to decide due to the NP-hardness of scheduling strategy. Nevertheless, note that the typical cache size of a peer is 1 GB (e.g., in PPLive), and typical movie size is about 500 MB. Thus, we assume that besides the currently watching movie, an active peer can cache one extra movie so as to cope with typical settings of real systems.

3 Optimal Replication in Software Deployment Scenario

In this and the next section, we discuss two typical deployment scenarios for P2P-VoD services. These scenarios have different fraction of active peers and they will affect the replication policies.

3.1 Deployment Scenario and Operation Modes

Recall the previous section, the two typical deployment scenarios we consider are software and set-top box deployment. In general, under the software deployment scenario, the fraction of active peers, or α, is close to 1. On the other hand, under the set-top box deployment scenario, α can take on value in $[0, 1]$ and may be significantly less than 1. The fraction of active peers can greatly influence the optimal replication strategy. In order to further analyze any P2P-VoD system, we define two operating modes as follows:

- *Deficit mode:* the average upload capacity of all peers is less than the playback rate of a movie, i.e., $\overline{u} < r$.
- *Surplus mode:* the average upload capacity of all peers is larger than the playback rate of a movie. In addition, we further divide it into two modes (a) *slightly-surplus mode*, i.e., $\overline{u} \gtrsim r$, and (b) *highly-surplus mode*, i.e., $\overline{u} >> r$.

3.2 Server's workload in Software Deployed Systems

In this subsection, we derive the server's workload for software deployment scenario where all peers in the system are active, i.e., $\alpha = 1$. Note that the deficit mode is *not* a rational operating point for a software P2P-VoD system where replication can rarely help. To see this, we know that under the deficit mode, we have $\overline{u} < r$. The server's total upload bandwidth consumption to support all peers is $N(r - \overline{u})$, where N is the number of peers in the system. When N increases, the server's bandwidth consumption will become very large as it increases linearly with N. Meanwhile, the upload bandwidths of peers are all used up to support concurrent peers, and thus no replication strategy can help to reduce the workload in $\mathcal{S}$. The only way to reduce the workload in $\mathcal{S}$ is to lower the playback rate r so that the system can be in the surplus mode.

On the other hand, when the system is in the surplus mode, replication is not always necessary. For a system operating in the highly-surplus mode, concurrent peers can support each other with ease. In other words, the abundance of peers' upload bandwidth makes replication unnecessary. This abundance in resource will motivate the P2P-VoD system designer to increase the playback rate r so as to improve video quality by better utilizing peers' resources, which eventually brings the system to the slightly-surplus mode.

To summarize, the only *interesting case* for the software deployment is when the system is operating at the

bandwidth (kbps)	768	384	256	128
share	50%	30%	5%	15%

TABLE 2: Peers' upload capacity distribution

slightly-surplus mode. As we will show, designing an optimal replication strategy is a non-trivial task.

Under the software deployment scenario, by setting $\alpha = 1$ in Eq. (9), we have the following proposition:

Proposition *2:* In software deployment, the average upload bandwidth consumption at the server $\mathcal{S}$ is:

$$\overline{U}(\mathbf{q}) = \sum_{\sum n_k = N} \left\{ N! \frac{\rho_1^{n_1}}{n_1!} \cdots \frac{\rho_K^{n_K}}{n_K!} \cdot \sum_{k=1}^{K} \left[\mathcal{D}_k(n_k) - q_k \sum_{l=1}^{K} (C_l^u(n_l) + \mathcal{D}_l(n_l) - n_l r) \right]^+ \right\}. \quad (11)$$

3.3 Characteristics of Deficit Bandwidth

To minimize $\overline{U}$, the decisions on q_k are highly dependent on the shape and characteristics of $\mathcal{D}_k(n_k)$. Let us consider some typical distributions (e.g., uniform, normal, ...) to represent peers' upload capacity so as to gain a deeper understanding. In addition, we also use numerical and fitting methods to show the characteristics of $\mathcal{D}_k(n_k)$. In here, we present some typical examples.

Example 1: (Exploring the characteristics of $\mathcal{D}_k(n_k)$): Assume n_k peers are watching movie M_k, and the movie playback rate is $r = 500$ kbps. Fig. 1 illustrates the impact of movie popularities on the deficit bandwidth $\mathcal{D}_k(n_k)$ when we vary the $n_k \in [1, 100]$. In Fig. 1(a), the peers' upload capacity is normally distributed with average of 540 kbps and variance 200. It shows that $\mathcal{D}_k(n_k)$ is *concave* in n_k and can be fitted into the form $\mathcal{D}_k(n_k) = \mathcal{D}_0 - Ae^{-\lambda(n_k - n_0)}$. In Fig. 1(b), we set a more realistic bandwidth distribution according to [4]. The upload capacity distribution is illustrated in Tab. 2. In Fig. 1(c), we still use this bandwidth distribution, and we consider the impact of user operations. We assume that due to this effect, when a peer seeks help from a concurrent peer, it has a probability 0.8 to successfully find the chunk needed. The deficit bandwidth $\mathcal{D}_k(n_k)$ shows very similar trend as the previous figures, but only differs in that the deficit workload is a bit larger compared to Fig. 1(b). We omit the similar cases that we studied, but they all imply the following observation:

Observation 1:

- *$\mathcal{D}_k(n_k)$ is a concave function on movie popularity n_k, and it converges to a fixed value when n_k approaches infinity.*
- *The convergence rate of $\mathcal{D}_k(n_k)$ increases with the average of upload capacity and decreases with the variance.*

Physical meaning: The concavity and convergence features imply that for unpopular movies, they have a much higher marginal increase on the deficit bandwidth $\mathcal{D}_k(n_k)$ while for popular movies, the marginal increase on $\mathcal{D}_k(n_k)$ is minimal. An intuitive explanation of this feature is that peers within a larger set can cooperate more effectively and thus achieve higher utilization of peers' upload bandwidth. Using law of large number on Eq. (7), for large enough i, we have $\sum_{j=1}^{i-1} \widetilde{u}_j \approx (i-1)\overline{u}$ with *high* probability. Furthermore, if we assume $\overline{u} > r$ and $i > r/(\overline{u} - r) + 1$, then we have

$$\widetilde{d}_i^c \simeq \min\left((i-1)\overline{u} - \sum_{j=1}^{i-1} \widetilde{d}_j^c, r \right) \geq \min\left((i-1)(\overline{u}-r), r\right) = r,$$

which means that any late arriving peer can receive sufficient download from concurrent peers and will not incur any bandwidth consumption on $\mathcal{S}$.

The above analysis shows the reason why deficit bandwidth $\mathcal{D}_k(n_k)$ is sub-linear. This also implies that the conventional wisdom of using the proportional replication strategy, or caching the more popular movies [6], [21], [22], is *sub-optimal*. Looking at Fig. 1, we see that a popular movie with 25 viewers needs nearly the same deficit bandwidth as a movie with 100 viewers, while an unpopular movie of five viewers needs more than half of the deficit bandwidth as compared with a popular movie with 100 viewers.

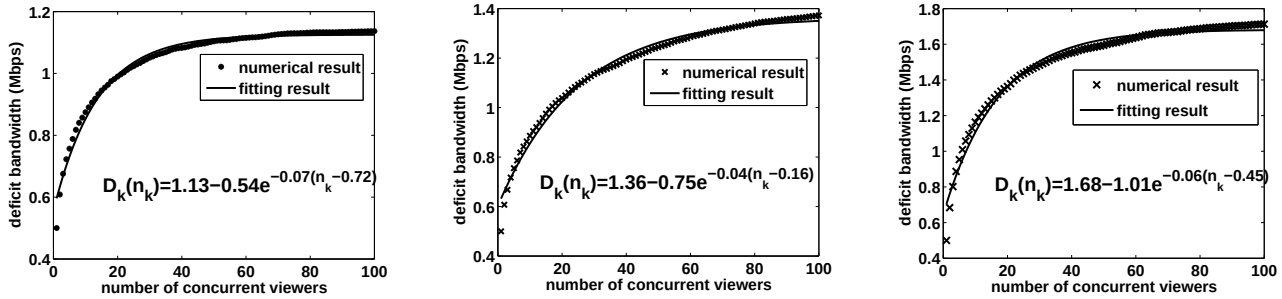
3.4 Characterizing the Optimal Replication Strategy

Now the impact of movie popularities on server's workload is clear: without replication, the server's workload for movie M_k is simply $\mathcal{D}_k(n_k)$. The goal of replication is to reduce the workload on $\mathcal{S}$. The non-linearity of $\mathcal{D}_k(n_k)$ implies one should be *greedy* in replicating unpopular movies. In the following, we use examples to justify our argument, and at the same time, determine the optimal replication ratios $\mathbf{q}$.

Example 2 (Sub-optimality of the proportional replication strategy): Assume that the system consists of $N = 600$ active peers and $K = 25$ movies. Movie M_1 - M_5 are popular movies while movie M_6 - M_{25} are unpopular. Peers choose each of the popular movies with a probability of $\rho_{pop} = 1/6$, or each of the unpopular movies with a probability of $\rho_{unpop} = 1/120$. The playback rate is set at $r = 500$ kbps, Peers' upload capacity follows Tab. 2.

Denote q_{unpop} as the fraction of cache space (which can store 600 movies) of caching any unpopular movie (M_6 - M_{25}), and $1 - q_{unpop}$ as the fraction of caching popular movies (M_1 - M_5). We develop a simulator to explore the replication strategy. In particular, we vary the fraction of cache space that is used to store unpopular movies from 0% (i.e., all cache space is used to store popular movies) to 100% (i.e., all cache space is used to store unpopular movies), and evaluate the system performance. Fig. 2 illustrates the upload bandwidth consumption on the server when we vary q_{unpop}.

If the system only replicates popular movies, then all deficit bandwidth $\mathcal{D}_k$ of unpopular movies will contribute to the server's workload. This corresponds to the first point from the left (about 15 Mbps) in Fig. 2. Similarly, if all replications are for unpopular ones, the server's workload is around 6 Mbps and this corresponds to the last point at the right of the figure. Since



(a) upload capacity: normal distribution with average 540 kbps and variance 200.

(b) upload capacity: discrete distribution in Tab. 2.

(c) upload capacity: discrete distribution with peer operations.

Fig. 1: Deficit bandwidth $\mathcal{D}_k$ vs. n_k

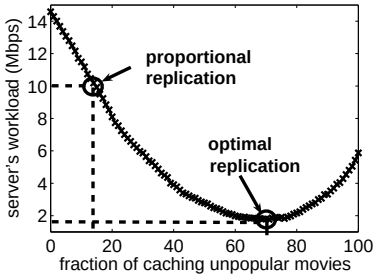


Fig. 2: Replication impact on server's workload in a software-deployed system, where $r = 500$ kbps.

	popular	moderate	unpopular
proportional fraction	71%	14%	14%
optimal replication fraction	18%	32%	50%
deficit bandwidth fraction ($\hat{q}_k$)	21%	30%	49%

TABLE 3: Optimal replication vs. popularity fractions

$5/6$ fraction of the requests are for popular movies, using the *proportional replication strategy*, only $1/6$ or 17% of the cache space should be dedicated to store the unpopular movies. From the figure, we see that the workload on the server will be around 10 Mbps. However, the *optimal replication fraction* for unpopular movie should be around 70% where the server's upload bandwidth consumption is less than 2 Mbps. This simple example validates the following observation:

Observation 2: *In a software-based system, proportional replication is far from optimal and incurs a higher server's workload as compared with the optimal replication strategy.*

Example 3 (Exploring optimal replication ratio): We consider a P2P-VoD system with 700 peers and 35 movies. M_1 - M_5 are popular movies with $\rho_{pop} = 1/7$; M_6 - M_{15} are moderately popular movies with $\rho_{mid} = 1/70$; while M_{16} - M_{35} are unpopular movies with $\rho_{unpop} = 1/140$. We simulate this system and explore the server's workload at each possible integral fraction combination $\mathbf{q} \in \{(q_{pop}, q_{mid}, q_{unpop}) : q_{pop} + q_{mid} + q_{unpop} = 100\%\}$. Using exhaustive search, the popularity fractions and optimal replication fractions of different movies are shown in Tab. 3. Obviously, the optimal replication strategy should be "aggressive" in caching unpopular movies.

It is mathematically difficult to derive the optimal ratios of replication from the optimization problem, and one cannot afford to use exhaustive search for large scale P2P-VoD systems. However, we obtain an interesting finding: the optimal ratio for replication for movie M_k should be close to the fraction of deficit bandwidth $\mathcal{D}_k(n_k)$ for movie M_k among the total deficit bandwidth, i.e., let

$$\hat{q_k} = \frac{\mathcal{D}_k(n_k)}{\sum_{i=1}^{K} \mathcal{D}_i(n_i)}, \tag{12}$$

then we have

$$q_k^{opt} \simeq \hat{q_k}. \tag{13}$$

For instance, in the above example $\hat{q}_{pop} = 21\%, \hat{q}_{mid} = 30\%$ and $\hat{q}_{unpop} = 49\%$, which are near to the optimal ratios for replication. This example reveals the following observation:

Observation 3: *Proportional deficit bandwidth replication is a good estimate to the optimal replication ratios.*

Our model can be easily extended to heterogeneous playback rates of movies. Further, Observation 3 still holds for this case. Due to page limit, we omit the details; interested readers may refer to the supplementary file.

Examples using other typical settings all strongly support our observations. We like to point out that there is an intuitive explanation to the above resource allocation strategy: without replication, $\mathcal{D}_k(n_k)$ is exactly the server's bandwidth consumption for movie M_k, and thus, allocating remaining bandwidth in this ratio is the most efficient method to balance each movie's request load to the server $\mathcal{S}$. This idea leads to the replication algorithm which we will discuss in the next section.

4 Optimal Replication in Set-top Box Deployment Scenario

In the set-top box deployment scenario, peers are always online even when they are not watching any movie. In other words, α might be significantly less than 1. The existence of inactive peers greatly changes the system design. Since the number of downloading peers may be significantly less than uploading peers, therefore, the playback rate of movies could be set higher compared with the software based deployment. For example, if only half of the total N peers in the system are watching movies, then total available upload bandwidth from peers is still $\overline{u}N$, while the total download requirement

is $r \cdot N/2$. Thus, the server can work with low workload if r is set no larger than $2\overline{u}$. For better quality of service, VoD providers have incentive to increase the playback rate as high as possible, or $r \simeq 2\overline{u}$. In this case, the movie playback rate is usually larger than the average upload capacity.

Since the playback rate is larger than the average upload bandwidth, an active peer is unlikely to have remaining bandwidth after serving concurrent peers. In other words, nearly all the replication peers are inactive peers, so

$$\overline{U}(\mathbf{q}) \simeq \sum_{\sum n_k = \alpha N} \left\{ (\alpha N)! \frac{\rho_1^{n_1}}{n_1!} \cdots \frac{\rho_K^{n_K}}{n_K!} \sum_{k=1}^{K} \left[n_k(r-\overline{u}) - q_k C_{\mathcal{I}}^u \left((1-\alpha)N \right) \right]^+ \right\}. \quad (14)$$

Fig. 3(a) illustrates this scenario with $r = 1.1$ Mbps and the peer's upload capacity follows Tab. 2. From the figure, we can see that the deficit workload $\mathcal{D}_k(n_k)$ is *linear* in terms of n_k, or

$$\mathcal{D}_k(n_k) \simeq n_k(r - \overline{u}). \quad (15)$$

Intuitively, since the deficit bandwidth of concurrent peers is linear in terms of the movie popularity, we should replicate a movie *proportionally* to its popularity. However, since the statistic of estimating the popularity of an unpopular movie has a higher variance, one should be a bit more "greedy" in replicating unpopular movies. We use an example to support this claim.

Example 5 (Optimal replication in a set-top box system): The setting of this example is similar to Example 2, but the differences are that we add 600 idle peers, i.e., $\alpha = 0.5$, and increase the playback rate to $r = 1.1$ Mbps. Fig. 3(b) shows total workload of the server with respect to q_{unpop}. The optimal value for q_{unpop} is around 22%, a bit larger than the fraction of unpopular movie viewers (17%). This result validates the above argument.

Note that the replication ratios in Observation 3 is still a good estimate to the optimal replication ratio. In this example, $\widetilde{q}_{pop} = 81\%$ and $\widetilde{q}_{unpop} = 19\%$, where the server's workload is near the minimum.

Last but not least, comparing this result with that in previous examples, we can conclude that the optimal replication strategy is very different for the software deployed and the set-top box deployed P2P-VoD systems.

5 Algorithms to Control Replication Ratios

Let us present several algorithms to achieve the optimal replication ratio $\mathbf{q}$ in a P2P-VoD system. Note that a P2P-VoD is with high churn, therefore, the replication algorithms need to be robust and efficient in minimizing the server's workload. In general, there are two ways to control or adjust the replication ratios:

Algorithm 1 Replacement Algorithm

```
for any peer that starts to watch a new movie do
  if this movie is already stored in its local cache then
    do not replace any movie;
  else
    for each movie M_i in its local cache do
      calculate the expected # of caches for M_i:
          N_i^exp = [D_i(n_i)N] / sum_k D_k(n_k);
      calculate the current # of caches for M_i: N_i^cur;
      calculate the satisfaction index for M_i:
          SI_i = N_i^cur / N_i^exp;
    end for
    replace the movie with the highest SI_i.
  end if
end for
```

- **Passive adjustment:** peers adjust the replication ratio in the P2P-VoD system using replacement algorithms, i.e., when a peer's local cache is full, it decides which movie(s) to purge so as to accommodate a new movie that can reduce the server's workload;
- **Active adjustment:** the server $\mathcal{S}$ and all peers actively push out (or upload) data for which the replicas are not at the desired replication ratios.

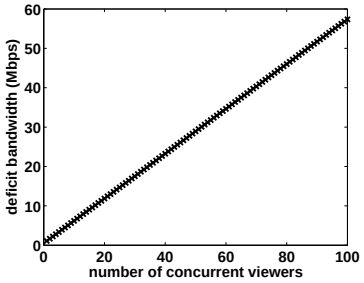
5.1 Passive Adjustment via Replacement Algorithm

When the cache space of a peer is full, a peer has to decide which movie(s) to purge from the cache space such that new movies could be cached. Since we want to explore the impact of movie popularities to the server's workload, we consider replication algorithms that replace the whole movie, instead of partial replacement. Nevertheless, generalization can be easily made.

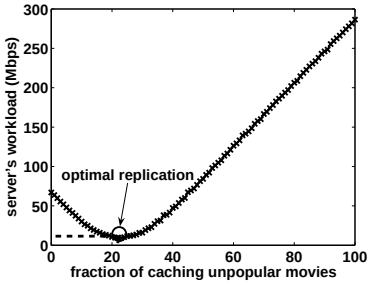
Previous work [6] suggested a weight-based evaluation scheme, or replicating proportionally to a movie's popularity. Our previous examples already show the sub-optimality of such replication algorithm and one should replicate movie proportionally to its *deficit bandwidth*. This leads to the following replacement algorithm, which is illustrated in Algorithm 1.

The main idea behind this algorithm is to keep the number of replicas proportional to the deficit bandwidth. We define *satisfaction index* (SI) to express the extent that a movie has enough replicas. When $SI_k < 1$, it means that the number of replicas for movie M_k is below the desired replication ratio. On the other hand, if $SI_k > 1$, the number of replicas of M_k exceeds the desired ratio. Therefore, in each round, we purge the movie that has the highest SI from the local cache.

The idea of the algorithm is derived from the optimization framework. The objective of replication is to minimize $\overline{U}$. Noting Eq. 8, the distribution of peers in each movie is centered around $\tilde{n_k} = N\rho_k, \forall k$. Recalling Proposition 1, if we set $q_k = \rho_k$ (which is the objective of



(a) Deficit bandwidth



(b) Replication impact on server's workload

Fig. 3: Replication in set-top-box system, where $r = 1.1$ Mbps, upload capacity distribution follows Tab. 2

Algorithm 2 Push Algorithm

1: **while** system is in an idle time slot **do**
2: find out a list of replication peer candidates which will most likely be inactive at the publishing instance;
3: estimate the total volume available for push
4: calculate the total volume desired for coming-to-hot movies:
$$\sum_k \mathcal{D}_k(n_k) \times \text{(movie length)}$$
5: **if** total available volume is larger than desired **then**
6: assign each movie M_i with the volume it desires;
7: **else**
8: **for** each movie M_i **do**
9: assign volume
$$\frac{\mathcal{D}_i(n_i)}{\sum_k \mathcal{D}_k(n_k)} \times \text{(total available volume)}$$
to the list of replication peer candidates;
10: **end for**
11: **end if**
12: **end while**

Algorithm 1), then we can minimize $\sum_{k=1}^{K} \left[\mathcal{D}_k(N\rho_k) - \mathcal{R}_k(\mathbf{n}, q_k)\right]^+$, and can hence approximately minimize the average workload of the server in Eq. 9.

Although requiring global information, this algorithm is efficient in practice. Each peer reports to the tracker the current cached movies, and the tracker can calculate the satisfaction indices and broadcast to all peers. These can be done in a proper frequency (e.g., once in five minutes), and thus the overhead can be minimal. It is also worth noting that this frequency cannot be too low, otherwise, some movies will oscillate between high and low satisfaction indices and the system cannot converge. In Sec. 6, we evaluate the impact of delay in broadcasting SI and show the convergence of the system.

5.2 Active Adjustment via Push Strategy

Our proposed replacement algorithm can reduce the server's workload, other factors still constrain the system performance. For instance, the server's workload may dramatically increase when a popular movie is newly released, since there will be a sudden surge of demand but only few peers in the system have cached this movie. By simply relying on the replacement algorithm, the system will take a long time to converge to the optimal replication ratios. To overcome this limitation, one can take a proactive approach to push out a movie if we know that it will be a popular object. This leads to the push algorithm as depicted in Algorithm 2. In essence, the push algorithm utilizes the excessive bandwidth in idle time slots to push some data for future use so server's workload could be "*flattened*" over time. This algorithm causes an overhead at the server. During a time period, the server incurs an extra upload consumption of $N_o r$ if it pushes the data to N_o peers. In the next section, we will show that this overhead can be minimal if we limit the number of peers to push the data to.

Predicting the dynamics of movie popularity is important in our push algorithm, but is difficult in general. One important feature is the "*daily periodicity*" [6], i.e., most peers access the network every evening, and this is in particular true for TV series. We can take this advantage and proactively replicate the content in the daytime. The authors in [11] discussed popularity prediction in a P2P-VoD setting using time-series analysis techniques. Besides periodicity, the movie content also plays a major role. An accurate prediction on the popularity dynamic is still a challenging research topic and is beyond the scope of this paper. In the following section, we assume the popularity dynamics is known a priori and present the performance results on using our proposed algorithms.

Our push algorithm shares the similar idea proposed in [14], which considers homogeneous peers with the same upload bandwidth and storage space. In this work, the authors evaluate the system performance when the request arrival rate is low, i.e., the system is in the highly surplus mode. Hence, the system performs well even though each video has only one replica. In contrast, by carefully calculating the number of replicas for each video, our design can support a higher playback rate where the total upload bandwidth from all peers is slightly larger than the total download requirement. We do not compare the performance of the two approaches due to the different system settings.

6 Performance Evaluation

6.1 Performance of Replacement Algorithm

We carry out extensive simulation to validate our models and evaluate the performance of our proposed replacement algorithm. We consider a software-based P2P-VoD system where $\alpha = 1$. Unless otherwise specified, we use the following as inputs to our simulator:

- The system contains $N = 10,000$ active peers. Peer's upload capacity distribution follows Tab. 2, with average of 531 kbps.
- The system provides $K = 250$ kinds of movies. Each movie is with the playback rate of $r = 500$ kbps.
- Movie popularity follows a Zipf distribution [13] with parameter $\gamma = 1$.
- At each round, an active peer requests a movie. Active peers switch movie channel every round.

The movie playback rate and average upload capacity we set are based on realistic settings like PPLive. The tracker in the system records the current cache distribution in the system and that all peers can use this information to do replacement. We compare server's workload using the following replacement algorithms: (a) *Our proposed algorithm*, which keeps replicas proportional to deficit bandwidth; (b) *Proportional algorithm*, which keeps the replicas proportional to movie popularity; (c) *First-in-First-out (FIFO) algorithm*, which keeps the newly cached movie and purges the oldest one.

Fig. 4 illustrates the comparison of the proportional replica distribution vs. the optimal replica distribution (both normalized to 1). We use $\gamma = 1$ as the default value unless otherwise specified, and arrange the movies in a decreasing order of popularity. Our proposed algorithm keeps fewer popular movies than the proportional distribution and is more greedy in replicating unpopular movies. Thus, the optimal replica distribution is "*flatter*" than the proportional distribution.

Fig. 5 compares the server's workload using different replacement algorithms. We record the server's load after round 20 since we are interested in the steady state performance. In Fig. 5(a), the server's average workload is around 9 Mbps using our proposed replacement algorithm, but is 75 Mbps using the proportional algorithm or 74 Mbps using the FIFO algorithm. Hence, our proposed push algorithm reduces the server's workload by a *factor of eight*. In Fig. 5(b), we add into user interactions, i.e., peers may perform various operations, and has a probability $\beta = 0.8$ of having the chunk requested by a concurrent peer. We recompute the deficit workload and decide the replication ratios, which shows a similar reduction on server's workload.

We also evaluate the performance of algorithms under different system parameters. Due to the page limit, we present our simulation results in the supplementary file.

Another important performance measure is the number of peers which can watch movie at the normal playback rate r (which we call *satisfied peers*) when given a finite server upload capacity. We set the simulation environment as the previous case and the server upload capacity is restricted at 10 Mbps. Fig. 5(c) shows the comparison on fraction of satisfied peers using different replacement algorithms. From the figure, our proposed replacement algorithm can ensure the maximum ratio of peers that can watch the movie at the playback rate, achieving approximately 100% of satisfied peers, while the ratio is around 96% when using the proportional or FIFO algorithm. These results validate that our proposed replacement algorithm can reduce the server's workload and at the same time, improve streaming quality.

We also considered practical issues on implementations, e.g., the delay of broadcasting information from the tracker, replacement based on incomplete information, the impact of multiple cache space and the peer departure. Due to the page limit, we present the evaluation the system performance in the supplementary file.

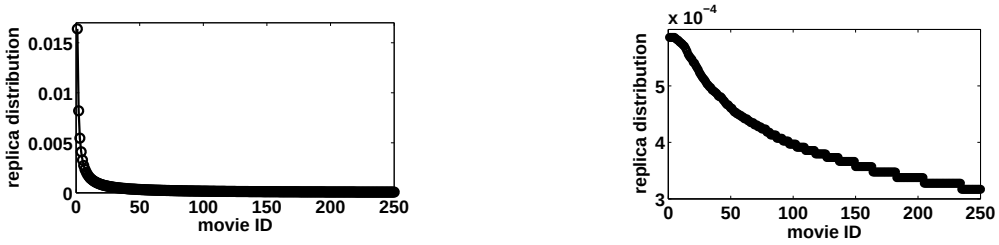
6.2 Performance of Push Algorithm

In here, we show that the proposed push algorithm can significantly improve the system performance, particularly for a set-top box deployed system. The key idea is to push some data to some inactive peers in advance such that the server's workload could be flattened down when publishing a new popular movie. The simulation setting is similar to the previous one but the differences are:

- There are 5,000 active peers and 5,000 inactive peers.
- Playback rate of a movie is doubled to $r = 1.0$ Mbps.
- Initially there are $K = 249$ kinds of movies $\{M_1,...,M_{249}\}$ with a Zipf popularity distribution ($\gamma = 1$). At the beginning of round t_p, a new popular movie M_{250} is published. The popularities of these 250 movies follow a new Zipf distribution and M_{250} is the most popular one.

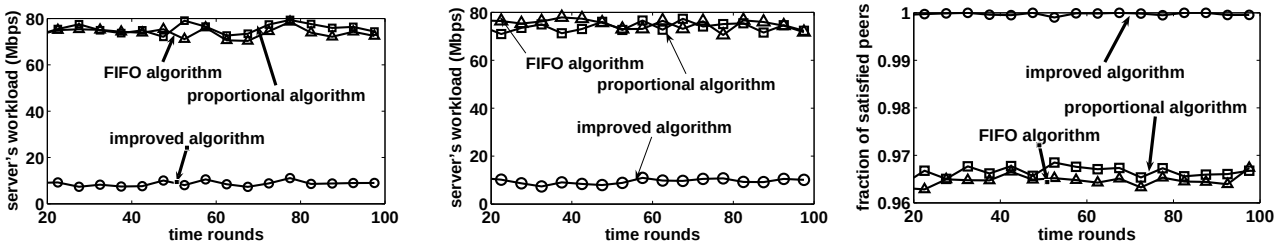
Movie M_1 - M_{249} can be stored in local cache of active and inactive peers. In a traditional system, before round t_p, M_{250} will not be in any local cache since no peer requests this unpublished movie. Hence, the system will have a very heavy workload at the publishing instance. With the push strategy, the new movie M_{250} can be pushed into inactive peers' cache before being published. We divide a day into 24 rounds (1 hour/round), of which we allow 18 rounds in advance to push a new movie before the publication. To avoid introducing much overhead, in each round we only push the data of this new movie into forty inactive peers. At $t_p = 19$, the new movie is published. After the publishing instance of the new movie, we do not push the data any more.

Fig. 6 illustrates the improvement using the push strategy. With a limited overhead, this strategy can significantly decrease the server's workload at the publishing instance of a new movie. In other words, this allows the system to "prepare" the data ahead of the flash crowd event. In comparison, the system without using the push strategy will experience heavy workload not only at the time of the new movie publication, but also



(a) proportional distribution (b) optimal distribution

Fig. 4: Proportional vs. optimal replica distribution



(a) impact on server's workload (b) replication with peer operations (c) impact on satisfied peers

Fig. 5: Comparison of replacement algorithms

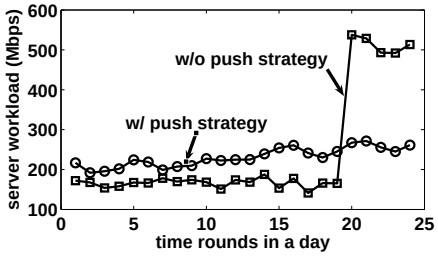


Fig. 6: Performance improvement using push strategy

in the following rounds after publishing the movie, as it is slow to adjust to the optimal replication ratio.

7 Related Work

Replication has long been an effective strategy to improve performance in P2P systems. Existing literatures have different objectives and hence lead to different conclusions. Kangasharju *et al.* [7] studied how to replicate P2P content distribution systems so as to optimize file availability. The main result is the logarithmic replication rule which states the number of copies for a file should be proportional to the logarithm of file popularity. Lin *et al.* [9] formulated the replication problem as an optimization problem and compared the performance of several decentralized algorithms in terms of improving file availability. They showed each algorithm has an advantage depending on system parameters. Tewari *et al.* [16] claimed that in P2P networks, proportional replication is optimal in terms of minimizing the average number of links traversed to a download. They showed that the least-recently-used (LRU) algorithm automatically achieves near-proportional replication. Cohen *et al.* [3] discussed the optimal replication to improve the effectiveness of blind search in a decentralized unstructured P2P network. They showed that uniform and proportional replication yield the same performance while the optimal lies in between them. Leukopoulos *et al.* [10] proposed a hybrid Genetic Algorithm to study the data replication for given current replica distribution. Khan *et al.* [8] compared ten static heuristic Internet data replication algorithms so as to give insight the selection of a particular algorithm to be used in a given scenario.

Recent popularity of P2P-VoD systems leads to the rethinking of replication. Previous replication algorithms are not suitable due to a very different objective: to minimize the server's workload. However, early works in P2P-VoD did not address this issue. Authors in [4], [5] showed the benefit of a peer-assisted VoD system by utilizing peers' upload bandwidth. The deployment was restricted to single video approach, i.e., peers only redistribute content to concurrent peers and hence there is no real replication. Huang *et al.* [6] did original research in the design and implementation of a real P2P-VoD system. The multiple movie cache scheme revealed the possibility of replication, and showed reduction on server's workload. However, the weight-based replacement algorithm, which replicates movies proportionally to popularity, was reported to have poor performance for unpopular movies in [17]. Similar proportional replication strategies were also implemented in [21], [22].

Recently some work focused on replication for P2P VoD systems. Poon *et al.* [12] reformulated the replication problem in terms of minimizing server's workload. Cheng *et al.* [2] proposed and evaluated a heuristic algorithm of lazy replication. These two works were based on simulations but did not provide theoretical analysis. Tan *et al.* [15] discussed content placement for a P2P-

VoD system. Wu *et al.* [19] used dynamic programming to derive the optimal replacement strategy for P2P-VoD system. These two works assumed homogeneous upload capacity, which is different from our model which considers peer heterogeneity. Besides, our model differentiates concurrent peers and replication peers which is different from [19]. Chen *et al.* [1] considered prefetching strategy for a peer-assisted IPTV system based on a *structured IPTV platform*, which is different from unstructured P2P systems.

8 Conclusion

Movie replication is an effective technique in the design of P2P-VoD systems. However, it remains an open problem to answer what the optimal replication policy should be so as to minimize the server's workload. In this paper, we present mathematical models and formulate an optimization framework to understand the impact of movies' popularities on server's workload, and reveal important principles in designing optimal replication algorithms. We show that conventional proportional replication strategy is far from optimal; rather, one should be more "*aggressive*" to replicate unpopular movies. Furthermore, to operate the system at the optimal point, we propose both passive replacement and active push strategies. Our algorithms are effective even under a dynamic environment (e.g., with peers arriving or leaving the P2P-VoD system) and movies with different playback rates. We discuss practical implementation issues and validate via extensive simulations to show that we achieve high QoS guarantee in streaming and at the same time, reduce workload at the server.

References

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

[2] B. Cheng, L. Stein, H. Jin, and Z. Zhang. A framework for lazy replication in p2p vod. In *Proc. of NOSSDAV*, 2008.

[3] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proc. of ACM SIGCOMM*, 2002.

[4] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? In *Proc. of ACM SIGCOMM*, 2007.

[5] C. Huang, J. Li, and K. W. Ross. Peer-assisted vod: Making internet video distribution cheap. In *Proc. of IPTPS*, 2007.

[6] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang. Challenges, design and analysis of a large-scale P2P-VoD system. In *Proc. of ACM SIGCOMM*, 2008.

[7] J. Kangasharju, K. Ross, and D. Turner. Optimizing file availability in peer-to-peer content distribution. In *Proc. of IEEE INFOCOM*, 2007.

[8] S. Khan and I. Ahmad. Comparison and analysis of ten static heuristics-based internet data replication techniques. *Journal of Parallel and Distributed Computing*, 68(2):113–136, 2008.

[9] W. Lin, C. Ye, and D.-M. Chiu. Decentralized replication algorithms for improving file availability in P2P networks. In *Proc. of IWQoS*, 2007.

[10] T. Loukopoulos and I. Ahmad. Static and adaptive distributed data replication using genetic algorithms. *Journal of Parallel and Distributed Computing*, 64(11):1270–1285, 2004.

[11] D. Niu, Z. Liu, B. Li, and S. Zhao. Demand forecast and performance prediction in peer-assisted on-demand streaming systems. In *Proc. of IEEE INFOCOM*, pages 421–425.

[12] W. Poon, J. Lee, and D.-M. Chiu. Comparison of data replication strategies for peer-to-peer video streaming. In *Proc. of ICICS*, 2005.

[13] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, and J. Xu. Modeling channel popularity dynamics in a large iptv system. In *Proc. of ACM SIGMETRICS*, 2009.

[14] K. Suhy, C. Dioty, J. Kurosey, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello. Push-to-peer video-on-demand system: Design and evaluation. *IEEE JSAC*, 25(9):1706–1716, 2007.

[15] B. Tan and L. Massoulie. Brief announcement: adaptive content placement for peer-to-peer video-on-demand systems. In *Proc. of ACM PODC*, 2010.

[16] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer networks. In *Proc. of IEEE INFOCOM*, 2006.

[17] K. Wang and C. Lin. Insight into the P2P-VoD system: Performance modeling and analysis. In *Proc. of ICCCN*, 2009.

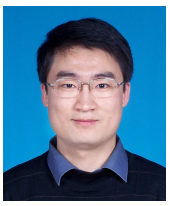
[18] D. Wu, Y. Liu, and K. Ross. Queuing network models for multi-channel P2P live streaming systems. In *Proc. of IEEE INFOCOM*, 2009.

[19] J. Wu and B. Li. Keep cache replacement simple in peer-assisted vod systems. In *Proc. of IEEE INFOCOM mini conference*, 2009.

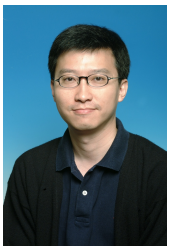
[20] W. Wu and J. C. Lui. Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation. In *Proc. of IEEE INFOCOM*, 2011.

[21] L. Ying and A. Basu. pcVOD: Internet peer-to-peer video-on-demand with storage caching on peers. In *Proc. of DMS*, Banff, Canada, 2005.

[22] W.-P. Yiu, X. Jin, and S.-H. Chan. Vmesh: Distributed segment storage for peer-to-peer interactive video streaming. *IEEE JSAC*, 25(9):1717–1731, 2007.



Weijie Wu is working towards his Ph.D. degree in the Department of Computer Science & Engineering at The Chinese University of Hong Kong. He is doing research in the Advanced Networking and System Research Group, under the supervision of Prof. John C.S. Lui. His current research interests include P2P networks and network economics. Before coming to CUHK, he obtained his Bachelor's degree from School of Electronics Engineering and Computer Science, Peking University in July 2008.



John C.S. Lui was born in Hong Kong and is currently a professor in the Department of Computer Science & Engineering at The Chinese University of Hong Kong. He received his Ph.D. in Computer Science from UCLA. His current research interests are in communication networks, network/ system security (e.g., cloud security, mobile security, etc), network economics, network sciences (e.g., online social networks, information spreading, etc), cloud computing, large scale distributed systems and performance evaluation theory. John serves in the editorial board of IEEE/ACM Transactions on Networking, IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Journal of Performance Evaluation and International Journal of Network Security. John was the chairman of the CSE Department from 2005-2011. He received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award. He is also a corecipient of the IFIP WG 7.3 Performance 2005 and IEEE/IFIP NOMS 2006 Best Student Paper Awards. He is an elected member of the IFIP WG 7.3, Fellow of ACM, Fellow of IEEE and Croucher Senior Research Fellow. His personal interests include films and general reading.