Peer-Assisted On-Demand Streaming: Characterizing Demands and Optimizing Supplies

Fangming Liu, *Member*, *IEEE*, Bo Li, *Fellow*, *IEEE*,
Baochun Li, *Senior Member*, *IEEE*, and Hai Jin, *Senior Member*, *IEEE*

Abstract—Nowadays, there has been significant deployment of peer-assisted on-demand streaming services over the Internet. Two of the most unique and salient features in a peer-assisted on-demand streaming system are the *differentiation* in the demand (or request) and the *prefetching capability* with caching. In this paper, we develop a theoretical framework based on queuing models, in order to 1) justify the superiority of service prioritization based on a taxonomy of requests, and 2) understand the fundamental principles behind optimal prefetching and caching designs in peer-assisted on-demand streaming systems. The focus is to instruct how limited uploading bandwidth resources and peer caching capacities can be utilized most efficiently to achieve better system performance. To achieve these objectives, we first use priority queuing analysis to prove how service quality and user experience can be statistically guaranteed, by prioritizing requests in the order of significance, including urgent playback (e.g., random seeks or initial startup), normal playback, and prefetching. We then proceed to construct a fine-grained stochastic supply-demand model to investigate peer caching and prefetching as a global optimization problem. This not only provides insights in understanding the fundamental characterization of demand, but also offers guidelines toward optimal prefetching and caching strategies in peer-assisted on-demand streaming systems.

Index Terms—On-demand video streaming, peer-to-peer, queuing model, performance evaluation

1 Introduction

In recent years, peer-assisted on-demand streaming systems have not only been the target of a substantial amount of research, but also been core industry products in both startup and established corporations alike, such as PPLive [1] and Joost [2]. Such systems offer great potential to bring a rich repository of video content to users' fingertips. Typical peer-assisted on-demand streaming systems provide users the convenience and flexibility of *watching whatever video clips whenever they wish*: they are able to play back the video, pause the video, perform random seeks to an arbitrary point of playback, or switch to and startup new videos. To offload dedicated streaming servers that maintain a baseline on content availability, peers contribute their uploading bandwidth resources and limited local cache capacities to cooperatively serve one another. Despite a large variety of different design choices in the literature (e.g., [1]), they have primarily been driven by engineering intuitions, rather than a rigorous set of design principles based on a solid theoretical foundation. In particular, we believe that two critical aspects of the design space need to be revisited from a theoretical perspective: *requests* on the "demand" side, and *caching* on the "supply" side.

First, in response to a large number of playback and random seek requests from peers, it would be desirable to have guaranteed continuous playback and short latencies after a random seek or an initial startup. How can a certain level of *guarantees* be provided with respect to service quality and user experience? *Second*, as peer caching has been used in both memory and nonvolatile storage to improve the "supply" of video content [1], [3], it is possible for peers to *actively* prefetch and cache certain content in local peer caches. The hope is that such prefetched content may become useful to other peers, and as such mitigate the load on dedicated servers. While a plethora of heuristics are available (e.g., [4]), potential benefits of such prefetching do not come without costs of bandwidth. There have been no founding principles on when and how prefetching should be performed, again based on solid theoretical analysis. If we jointly consider *requests* that make *demands* for content and *prefetching* that augments *supplies* of content with additional a priori consumption of upload bandwidth, we may be able to develop a theoretical framework that cultivates the root of a high-quality on-demand streaming system design.

In this paper, we seek to address these fundamental questions with the following contributions. Rather than heuristics based on intuition, we construct a new analytical framework based on queuing models, that 1) advocates and justifies the superiority of service prioritization based on a

- *F. Liu and H. Jin are with the Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, No. 5 Eastern Building, No. 1037 Luoyu Road, Hongshan District, Wuhan 430074, China. E-mail: {fmliu, hjin}@mail.hust.edu.cn.*
- *B. Li is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon 999077, Hong Kong. E-mail: bli@cse.ust.hk.*
- *B. Li is with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, ON M5S 3G4, Canada. E-mail: bli@eecg.toronto.edu.*

Manuscript received 24 Mar. 2011; revised 6 Sept. 2011; accepted 16 Oct. 2011; published online 10 Nov. 2011.
Recommended for acceptance by S. Nikoletseas.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-03-0193.
Digital Object Identifier no. 10.1109/TC.2011.222.

taxonomy of streaming requests, and 2) helps understand the fundamental principle behind optimal prefetching and caching designs. Uploading bandwidth and peer caching are finite resources in any design of on-demand streaming systems, and it is our focus to provide guidelines how a finite pool of uploading bandwidth resources and peer caching capabilities are to be utilized most efficiently toward better performance.

In particular, we first use priority queuing analysis to prove how service quality and user experience can be statistically guaranteed by prioritizing requests in the order of urgent playback (e.g., random seeks or initial startup), normal playback, and prefetching. We then proceed to a fine-grained stochastic supply-demand model to investigate peer caching and prefetching as a global optimization problem. This not only provides insights in understanding the fundamental characterization of the demand, but also offers guidelines toward optimal prefetching and caching strategies in peer-assisted on-demand streaming systems.

The remainder of this paper is organized as follows: in Section 2, we discuss our contributions in the context of related work. In Section 3, we present a conceptual queuing model for peer-assisted on-demand streaming systems, and justify the superiority of service prioritization based on a taxonomy of requests. Section 4 presents our investigation on caching and prefetching based on a fine-grained stochastic supply-demand model. Finally, we conclude the paper in Section 5.

2 Related Work

Though a number of commercial peer-assisted on-demand media streaming systems have been deployed (e.g., [1]), there is a lack of theoretical foundations and analyses toward the understanding of fundamental design principles and challenges in peer-assisted on-demand streaming systems. From a theoretical perspective, to date, only a few analytical studies on peer-assisted on-demand streaming exist in the literature. Suh et al. [3] have analytically investigated the design of a push-to-peer video-on-demand system in cable networks, and bounds of catalog sizes in such a system have been derived in a subsequent work [5]. More recently, Parvez et al. [6] have developed a theoretical model to analyze the performance of BitTorrent-like protocols for on-demand stored media streaming. However, these models have not been able to characterize the differentiation in demand, and thus are not in a position to provide insights on how to guarantee the service quality for urgent requests, such as random seeks. In contrast, our model characterizes demand of different levels of urgency through a priority queuing analysis, which both qualitatively and quantitatively justifies how service quality and user experience can be statistically guaranteed through service prioritization.

With respect to studies on caching and prefetching in peer-assisted on-demand streaming systems, there exist a number of proposals (e.g., [7]), mostly driven by engineering intuitions or heuristics. Guo et al. have proposed a hybrid media proxy system assisted by P2P networks, called PROP [8], in which DHT-based overlay management and cache replacement heuristics are designed to achieve system scalability and reliability. Instead of dictating specific system infrastructure and implementation, we focus on the underlying *resource allocation problem*—with respect to finite peer bandwidth and cache capacities in any design of on-demand streaming systems—from a theoretical perspective.

Rather than designing specific prefetching or caching strategies (e.g., [9]), we analytically characterize and understand the fundamental problem behind caching and prefetching designs based on sound stochastic queuing analysis. Rather than optimizing prefetching from an individual peer's viewpoint (e.g., [10]), we investigate optimal caching and prefetching as a global optimization problem to balance the system-wide demand and supply. Similarly, a cache redundancy model is developed in [8] to achieve the optimal distribution of media segment replicas across the system, under the assumption of Zipf-like distribution of segment popularity and the storage constraint of peer caches. In comparison, our model not only has no a priori assumption on segment popularity distributions, but also cohesively incorporates both storage and upload bandwidth constraints at peers.

While data replication and caching have been extensively studied in P2P file sharing and distributed storage systems to satisfy search queries and download demand (e.g., [11]), how to adapt existing replication schemes to peer-assisted on-demand streaming systems is still an open problem. Our model and performance analysis, which intrinsically consider the timing and bandwidth requirements in peer-assisted on-demand streaming, can reveal the fundamental principle of a *family* of prefetching and caching strategies. We show that simple heuristics used in recent real-world systems essentially lie in this family, and can potentially achieve comparable performance to optimal strategies.

In addition, there have been extensive prior studies on conventional web caching for reducing user access latencies to text-based web content. Different from web traffic caching under the widely accepted Zipf-like access pattern (e.g., [12]), the reference locality in many media workloads is shown to be less skewed than that of web objects [13]. Accordingly, we have evaluated peer-assisted caching designs under both the classical Zipf and the recently observed stretched exponential (SE) distribution [14] to obtain complementary understanding. This demonstrates that the performance of caching in peer-assisted on-demand streaming systems is distinctively affected by two representative access patterns, which is consistent with the implications from [14]. Also, server-based content delivery networks (CDNs) have been deployed to replicate media content across the Internet to move the content close to the viewers. For instance, it is shown that the popularity of live media programs hosted by Akamai CDN exhibits a two-mode Zipf distribution [15]. Such server-based solutions can improve the media streaming performance, yet they suffer from prohibitive cost [8]. In contrast, we focus on the cost-effective peer-assisted on-demand streaming, with mitigated server bandwidth costs demonstrated by our analysis.

3 Demand Characterization and Service Prioritization

3.1 Characterizing Requests for Segments

We begin by stating our model of a peer-assisted on-demand streaming system providing a rich repository of video files, each encoded with a constant bit rate. While different videos can have distinct lengths, each of them is divided into a number of fixed-length segments, and the total number of

segments is V. We assume that there exists a pool of servers, with an aggregate upload capacity U_s, deployed by the service provider, and there are N peers participating in the system in steady state. Let u_i denote the upload capacity of peer $i, \forall i \in N$. Then, the total uploading bandwidth resource in the system is $C = U_s + \sum u_i$. Without loss of generality, we assume that time t is slotted and each time unit corresponds to the amount of time for a peer to play back one segment. Given a fixed segment size as mentioned earlier, the total system upload capacity C can be normalized as c, in terms of the number of segments that can be served per unit time, henceforth referred to as *c service units*.

As peers play back video segments over time, they will also request segments from the system, which are represented by the total arrival rate of *segment requests* λ. Different from live video streaming, there is a unique and critical differentiation with respect to *requests* in on-demand streaming, which can be represented in a taxonomy of three categories:

- *Urgent playback* requests that are used to meet immediate buffering and playback needs after user interactions, such as initial startup of a video stream, and random seek activities.
- *Normal playback* requests that are used to maintain a buffer of segments in the near future, so that a smooth playback quality can be sustained within the vicinity of the current playback point.
- *Prefetching* requests that are used to prefetch certain desirable segments beyond the vicinity of the current playback point (and may even be in a different video) for local caching. Such prefetching requests are made with the hope that the prefetched segments can be used in the future to serve other peers, so that server bandwidth costs will be mitigated (as we shall elaborate in Section 4).

Each segment request requires to be satisfied by one of the available service units (i.e., uploading bandwidth resource from either peers or servers), in order to satisfy the *demand* from peers.

3.2 Prioritizing Services: Priority Queuing Analysis

To guarantee service quality and user experience, we advocate that when requests for segments—on the "demand" side—are served, they should be *prioritized* in the order of their urgencies, with three priority types $\mathcal{J} = \{3, 2, 1\}$ for urgent playback requests (with arrival rate λ_3), normal playback requests (with arrival rate λ_2), and prefetching requests (with arrival rate λ_1), respectively, where $\sum_{j=1}^{3} \lambda_j = \lambda$. In other words, requests belonging to a type $j = 3, 2, 1$ are associated with *high priority*, *medium priority*, and *low priority*, respectively, while requests within the same priority type will be served with the First-Come-First-Served (FCFS) policy.

As illustrated in Fig. 1, the entire system is abstracted as a service station consisting of c identical parallel service units with the mean service rate $\mu = 1$ each (i.e., one segment per unit time). Note that though peers can have heterogeneous upload capacities, our model can mitigate such a complexity by transforming the overall uploading bandwidth resource of the system to a number of homogeneous service units, under the assumption of heavy

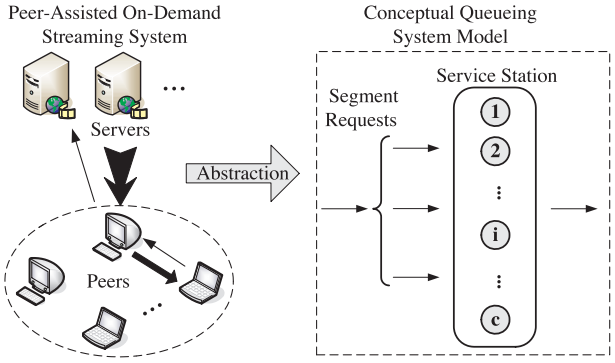


Fig. 1. A conceptual queuing system consisting of multiple parallel service units in response to the segment requests sent from peers, with the same aggregate upload capacity as the original peer-assisted on-demand streaming system.

workloads in such on-demand streaming services that tend to fully exploit the limited pool of bandwidth resources. Essentially, we abstract peer-assisted on-demand streaming with service prioritization as a *multiserver multipriority classes model*, represented by Kendall's notation [16] $M/M/c - PR$, where segment requests arrive with a Poisson process and service times follow an exponential distribution, c is the number of parallel service units, and PR specifies the queuing discipline as *Priority*, and the *system utilization factor* $\rho = \lambda/(c\mu) \in [0, 1)$ for steady state analysis. Our considerations include the following aspects:

- While still allowing peers to have heterogeneous capacities in original practical systems, our model can analyze the system performance through classical queuing models with homogeneous service units to yield tractable analysis. Otherwise, an asymmetric queuing system with heterogeneous service units is hard to obtain tractable results, or its analysis is only restricted to special cases such as $M/M/2$ systems [16].
- Since a peer (or server) can correspond to multiple service units depending on its upload capacities, our model can also cover the typical scenario where a peer (or server) is simultaneously serving multiple segments and its uploading bandwidth is fairly shared between these concurrent transmissions. For practical peer-assisted systems wherein peers usually cooperate with multiple partners, such a Processor Sharing (PS) policy is an adequate model of fair sharing between concurrent TCP connections [3].
- Note that each service unit essentially represents the *average* upload capacity of serving one segment per unit time. Due to practical factors such as network congestion and bandwidth fluctuation, there could be variations in serving segments. This implies that the service time of each service unit should be a random variable, rather than a deterministic constant that would be too idealized. To this end, we assume that the service time of each service unit follows an exponential distribution with mean service rate $\mu = 1$. This is considered to be a reasonable assumption and typically used in existing analytical studies of peer-assisted systems [17]. On

the other hand, even if one prefers to assume a deterministic service time, our model can be easily amended to a $M/D/c$ system.

- We mainly focus on the nonpreemptive mode for serving requests, though our model can also be amended to accommodate a preemptive mode.

To analyze and justify the superiority of service prioritization, we next use representative performance metrics of our queuing model to characterize the service quality and user experience in on-demand streaming systems, including the *playback fluency* and *latency after user interaction* (e.g., initial startup and random seeks).

3.2.1 Playback Fluency

Formally, given a certain number of segments, d, that are buffered ahead of the real-time playback point, it is drained at the playback rate (i.e., one segment per unit time), while fed by subsequent segments as a peer continuously views a video. Hence, the playback fluency can be reflected by the probability of $Pr((L-d)W \leq d)$, where W is the waiting time for a segment request to obtain service in steady state, and L is the time length for which the peer has been viewing the video. For example, L can be up to the video length. Let a factor $\alpha = \frac{d}{L}$ represent the relative playback buffer length, we have

$$Pr((L-d)W \leq d) = Pr\Big(W \leq \frac{\alpha}{1-\alpha}\Big), \tag{1}$$

where $Pr(W \leq \frac{\alpha}{1-\alpha})$ is essentially the *waiting time distribution* function for a segment request, $Pr(W \leq \tau)$, with a desired delay bound $\tau = \frac{\alpha}{1-\alpha}$. Intuitively, a higher level of $Pr(W \leq \tau)$ would imply a timely and sustained playback buffer, and thus a fluent viewing experience. Specifically, an extreme exercise with $\tau = 0$ and $Pr(W = 0)$ can represent the most stringent timing requirement, i.e., if the first segment in the playback buffer closest to the playback point is missing.

Unfortunately, in our model of peer-assisted on-demand streaming with service prioritization, the exact waiting time distributions in the corresponding multiserver multipriority classes model $M/M/c-PR$ are far less tractable, and most existing analytical studies are restricted to only two priority classes (e.g., [18]). Instead, we resort to the following simple approximation of the waiting time distribution for multipriority classes systems, with both theoretical support [19] and simulation-based observations [20]. Such a reasonable approximation is useful to estimate the trend of $Pr(W_j \leq \tau)$, $\forall j \in \mathcal{J}$, and facilitate our later comparison with traditional nonprioritization systems, especially under heavy workloads (i.e., $\rho \to 1$):

$$Pr(W_j \leq \tau) \approx 1 - \rho e^{-\rho\tau/\overline{W_j}}, \quad \tau > 0, \tag{2}$$

where $\overline{W_j}$ is the mean waiting time for jth priority segment requests in steady state, as given below:

$$\overline{W_j} = \frac{\left[\frac{(c\rho)^c}{c!(1-\rho)}p_0\right]}{c\mu(1-\sigma_j)(1-\sigma_{j+1})}, \tag{3}$$

where $\sigma_j = \sum_{i=j}^{|\mathcal{J}|}\rho_i$, $\rho_j = \lambda_j/(c\mu)$, $\rho = \sum_{j\in\mathcal{J}}\rho_j$, and

$$p_0 = \left[\sum_{k=0}^{c-1}\frac{(c\rho)^k}{k!} + \frac{(c\rho)^c}{c!}\frac{1}{1-\rho}\right]^{-1}$$

is the state probability of 0 jobs in the system.

Note that though we focus on three priority types (i.e., $\mathcal{J} = \{1,2,3\}$) according to on-demand streaming features, (3) is also general for $M/M/c-PR$ models with any number of priority types [16].

3.2.2 Latency after User Interaction

Another performance concern in peer-assisted on-demand streaming is the *latency* to retrieve segments after initial startups or random seeks. When a peer joins a video or seeks to a new playback position, at least a minimum buffer of segments, up to the playback buffer size d, need to be requested and downloaded before the video starts playback. Such a *latency* can be modeled as a function $D(d,R)$, depending on the playback buffer size d and the retrieval time R for a segment. Specifically, we use the following monotonically increasing function as the expected latency:

$$\overline{D(d,R)} = \beta d\overline{R}, \quad \beta \in \left[\frac{1}{d}, 1\right], \tag{4}$$

where $\overline{R} = \overline{W} + 1/\mu$ is the mean retrieval time for a segment in steady state, and β is a factor depending on how many segment requests within the buffer are served in parallel by the system. Alternatively, β can also be interpreted as an acceptable buffer level for startups. For example, a buffer level of 75 percent is empirically regarded as satisfactory in UUSee [21]. Note that $\overline{D(d,R)}_{\min} = \overline{R}$ essentially places a lower bound on the achievable startup delay [6] or seek latency, since at least the segment closest to the playback (or seek position) needs to be downloaded. Intuitively, a smaller value of $\overline{R}$ would imply a shorter latency to retrieve segments.

For our model of peer-assisted on-demand streaming with service prioritization, from (3), the lower bound of expected latency $\overline{D_j(d,R_j)}_{\min} = \overline{R_j}$ for different priority types $j \in \mathcal{J}$ under the $M/M/c-PR$ model is given as

$$\overline{R_j} = \frac{\left[\frac{(c\rho)^c}{c!(1-\rho)}p_0\right]}{c\mu(1-\sigma_j)(1-\sigma_{j+1})} + \frac{1}{\mu}, \tag{5}$$

where $\sigma_j = \sum_{i=j}^{|\mathcal{J}|}\rho_i$, and p_0 is given earlier.

According to (4) and (5), we have $\overline{D_j(d,R_j)} \geq \overline{R_j} \geq 1/\mu, \forall j \in \mathcal{J}$. This is also true in real-world systems, since it at least takes a certain amount of time to download the target segments after a random seek or an initial startup, even if such requests can be served immediately.

3.3 Statistical Guarantees with Service Prioritization

We are now ready to justify and demonstrate how service quality and user experience can be statistically guaranteed with our service prioritization principle (henceforth referred to as *prioritization systems*). Specifically, to establish a baseline performance reference point, we also include a comparison with traditional on-demand streaming systems without service prioritization (henceforth referred to as *nonprioritization systems*). In nonprioritization systems, all segment

requests from downstream peers to the serving peers and servers will be treated identically. Essentially, this corresponds to $M/M/c-FCFS$ queuing models [16], and the corresponding performance measures $Pr(W_{FCFS} \leq \tau)$ and $\overline{R_{FCFS}}$ can be represented by the waiting time distribution and mean sojourn time of a job in steady state, respectively:

$$Pr(W_{FCFS} \leq \tau) = \begin{cases} 1 - \dfrac{(c\rho)^c}{c(1-\rho)} p_0, & \tau = 0, \\ 1 - \dfrac{(c\rho)^c}{c!(1-\rho)} p_0 e^{-c\mu(1-\rho)\tau}, & \tau > 0, \end{cases}$$

$$\overline{R_{FCFS}} = \overline{W_{FCFS}} + \frac{1}{\mu}, \tag{6}$$

where $\overline{W_{FCFS}} = [\frac{\rho}{(1-\rho)\lambda}][\frac{(c\rho)^c}{c!(1-\rho)} p_0]$ is the mean waiting time of a job in the corresponding $M/M/c-FCFS$ system in steady state, and $p_0 = [\sum_{k=0}^{c-1} \frac{(c\rho)^k}{k!} + \frac{(c\rho)^c}{c!}\frac{1}{1-\rho}]^{-1}$ is the state probability of 0 jobs in the corresponding $M/M/c-FCFS$ system.

First, we have the following relationship with respect to the latency performance.

Theorem 1. *Given a limited system upload capacity, prioritization systems with three priority types $\mathcal{J} = \{3, 2, 1\}$ for urgent playback, normal playback, and prefetching can 1)* guarantee *to improve the service quality and user experience for urgent requests (*e.g., *random seeks and initial startup); 2)* conditionally *improve the normal playback fluency; and 3) at a cost of deferring prefetching requests with a* conditionally *bounded proportion to the performance of nonprioritization systems, as characterized by the following relationship:*

$$\overline{R_3} < \overline{R_{FCFS}} < \overline{R_1}, \quad \text{for } \rho \in [0, 1), \tag{7}$$

$$\overline{R_2} \leq \overline{R_{FCFS}}, \quad \text{for } \frac{1-\rho}{(1-\rho_2-\rho_3)(1-\rho_3)} \leq 1, \tag{8}$$

$$\overline{R_2} > \overline{R_{FCFS}}, \quad \text{for } \frac{1-\rho}{(1-\rho_2-\rho_3)(1-\rho_3)} > 1, \tag{9}$$

$\frac{\overline{W_1}}{\overline{W_{FCFS}}}$ remains constant as ρ increases, when ρ_1 increases while ρ_2 and ρ_3 remain constant.

Proof. First, based on conservation law, we have $\overline{W_3} < \overline{W_2} < \overline{W_1}, \forall \rho \in [0, 1)$. By simply adding the mean service time $1/\mu$, we have $\overline{R_3} < \overline{R_2} < \overline{R_1}, \forall \rho \in [0, 1)$.

Furthermore, to compare $\overline{R_{FCFS}}$ and $\overline{R_j}, \forall j \in \mathcal{J}$, we need to compare the corresponding $\overline{W_{FCFS}}$ and $\overline{W_j}$, $\forall j \in \mathcal{J}$, as follows:

$$\frac{\overline{W_j}}{\overline{W_{FCFS}}} = \frac{1-\rho}{(1-\sigma_j)(1-\sigma_{j+1})},$$

(by (3) and (6)), where $\sigma_j = \sum_{i=j}^{|\mathcal{J}|} \rho_i$.

Hence, for high priority $j = 3$, we have

$$\frac{\overline{W_3}}{\overline{W_{FCFS}}} = \frac{1-\rho}{1-\sigma_3} = \frac{1-\rho}{1-\rho_3} < 1, \quad (\text{typically } \rho_2 + \rho_1 > 0). \tag{10}$$

Then, for low priority $j = 1$, we have

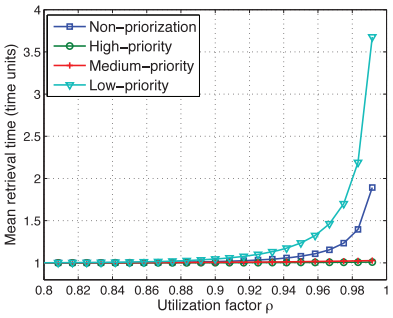


Fig. 2. Mean retrieval times for high-priority, medium-priority, and low-priority requests in prioritization on-demand streaming systems compared to the mean retrieval times in traditional nonprioritization systems, along with the increase of the system utilization factor.

$$\frac{\overline{W_1}}{\overline{W_{FCFS}}} = \frac{1-\rho}{(1-\sigma_1)(1-\sigma_2)} = \frac{1}{1-\rho_2-\rho_3} > 1. \tag{11}$$

Combining (10) and (11) gives $\overline{R_3} < \overline{R_{FCFS}} < \overline{R_1}$, $\forall \rho \in [0, 1)$.

In addition, (11) also indicates that $\frac{\overline{W_1}}{\overline{W_{FCFS}}}$ can keep constant as ρ increases, when ρ_2 and ρ_3 keep constant while ρ_1 increases.

Finally, for medium priority $j = 2$, we have

$$\frac{\overline{W_2}}{\overline{W_{FCFS}}} = \frac{1-\rho}{(1-\sigma_2)(1-\sigma_3)} = \frac{1-\rho}{(1-\rho_2-\rho_3)(1-\rho_3)}. \tag{12}$$

From this, we can obtain the conditions (8) and (9) for $\overline{R_2} \leq \overline{R_{FCFS}}$ and $\overline{R_2} > \overline{R_{FCFS}}$, respectively. □

Remark. Our model with Theorem 1 not only characterizes the unique and critical differentiation in three distinct request classes in peer-assisted on-demand streaming systems, but also qualitatively instructs how limited uploading bandwidth resources can be effectively utilized through service prioritization, so as to provide statistical guarantees on service quality and user experience. This aims to provide a theoretical underpinning of peer-assisted on-demand streaming systems, which is complementary to existing heuristic-based studies in this area.

3.4 Service Prioritization: Performance Evaluation

To more clearly demonstrate the superiority of service prioritization, we perform a series of performance analysis. Fig. 2 plots the mean retrieval times for high-priority, medium-priority, and low-priority requests in prioritization systems compared to the mean retrieval time in nonprioritization systems. As the system utilization factor ρ increases, the performance of traditional nonprioritization systems will degrade as a whole, which implies an unsatisfactory service quality and user experience, especially for urgent requests (e.g., random seeks and initial startup) and normal playback. This is due to the heavy competition among all segment requests for a limited pool of uploading bandwidth resources in the system, and the passive resource allocation mechanism without being aware of service quality and user experience.

In contrast, under the same system capacity and utilization factor, prioritization systems can effectively allocate a limited pool of uploading bandwidth resources, and successfully maintain a nearly constant level of

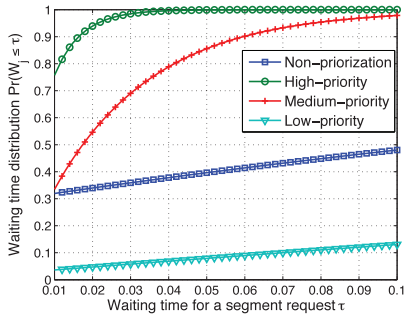


Fig. 3. Waiting time distribution for high-priority, medium-priority, and low-priority requests in prioritization on-demand streaming systems compared to traditional nonprioritization systems, under a heavy system utilization factor $\rho = 0.975$.

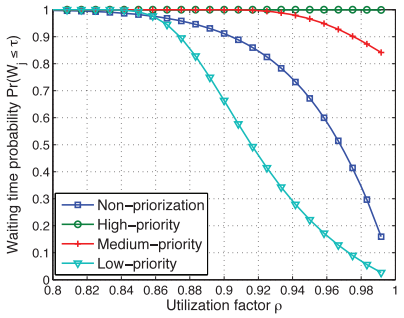


Fig. 4. Waiting time probability (with a desired $\tau = 0.06$) for high-priority, medium-priority, and low-priority requests in prioritization on-demand streaming systems compared to traditional nonprioritization systems, along with the increase of system utilization factor.

performance for high-priority and medium-priority requests, which implies timely responses for random seeks and initial startup, as well as fluent playback, even though at a cost of deferring prefetching requests that are expected not to directly affect user viewing experiences.

Next, we quantitatively examine the performance with respect to playback fluency. Fig. 3 plots the waiting time distribution for high-priority, medium-priority, and low-priority segment requests in prioritization systems, compared to traditional nonprioritization systems, under a heavy system utilization factor $\rho = 0.975$. Specifically, we examine a representative range of $\tau = \frac{\alpha}{1-\alpha} \in [0.01, 0.1]$, and assume L is set as an expected video length of 1 hour.

We obtain the following insights: 1) As the desired delay bound for a segment request τ increases ($\alpha = \frac{d}{L} \uparrow$), the waiting time probabilities $Pr(W_{FCFS} \leq \tau)$ and $Pr(W_j \leq \tau)$, $\forall j \in \mathcal{J}$, all increase. This indicates that a longer playback buffer length can help improve the playback fluency, which is consistent with empirical experiences. 2) However, $Pr(W_{FCFS} \leq \tau)$ in nonprioritization systems is relatively low, which implies a higher probability of interrupted viewing experience. In contrast, the playback fluency in prioritization systems (i.e., $Pr(W_2 \leq \tau)$) grows rapidly, and can even approach 1 with more adequate buffer lengths. This clearly demonstrates that prioritization systems can offer a better service quality and user experience than traditional nonprioritization systems. 3) The performance for high-priority requests in prioritization systems is even better with $Pr(W_3 \leq \tau)$ quickly approaching 1, which implies timely responses for urgent requests such as initial startup and random seeks. On the other hand, the waiting time for low-priority prefetching requests could become correspondingly longer due to the conservation law. Since prefetching requests beyond the playback buffer mainly aim to cache and serve future demand, a relatively longer delay for such requests is acceptable in practice, given that *the service qualities for normal playback and urgent demands are statistically guaranteed*.

Further, under the same setting of $\tau = 0.06$, Fig. 4 compares the waiting time distributions of segment requests. We observe that as we increase ρ by allowing more prefetching requests, the playback fluency in nonprioritization systems could degrade significantly. In contrast, the playback fluency in prioritization systems can be maintained at a high level.

4 Optimizing Prefetching and Caching under a Stochastic Supply-Demand Model

To jointly consider requests that make *demands* for content and prefetching that augments *supplies* of content, we further proceed to analyze the prefetching and caching designs in peer-assisted on-demand streaming systems as a global optimization problem, under a stochastic supply-demand model as follows:

- ▷ *Demand*. Assume the system-wide normal playback requests for segments arrive as a Poisson process of rate λ_2, and the access probabilities of segments are denoted as $(a_1, a_2, \ldots, a_V)$, which satisfies $\sum_{i=1}^{V} a_i = 1$. Without loss of generality, we assume $a_i \geq a_j$, $\forall i < j$. Hence, the expected amount of requests for segment i per unit time is $\lambda_2 a_i$.
- ▷ *Supply*. Assume that each peer in the system maintains a limited size of local cache, up to B segments. The system-wide caching state of segments at time t is denoted as $X(t) = (x_1(t), x_2(t), \ldots, x_V(t))$, where $x_i(t) \leq N$ represents the number of peers holding segment i in their local caches at time t. Without loss of generality, we assume peer local caches will be completely filled as time progresses, and hence $\sum_{i=1}^{V} x_i(t) = NB, \forall t$. To fully utilize their local caches to serve one another, peers can proactively prefetch certain segments to their local caches, and replace certain segments due to the limited cache size. We are interested in the optimal prefetching and cache replacement strategies to produce the optimal system-wide cache state, so that system-wide normal playback demand can be satisfied by peers holding those segments, rather than by servers.

As peers cache and prefetch segments, each segment is associated with a certain service capacity, denoted as $(o_1(t), o_2(t), \ldots, o_V(t))$, in terms of the number of segments that can be served by peers per unit time. Generally, we assume $\sum_{i=1}^{V} o_i(t) = \varepsilon C_p$, where $C_p = \sum_{j=1}^{N} u_j$ is the total uploading capacity of peers, and $\varepsilon \in (0, 1]$ represents the efficiency of utilizing peer uploading bandwidth. Intuitively, the larger the number of peers holding segment i, the larger the capacity there exists to serve this segment. To facilitate our analysis, we use a proportion function for the expected service capacity of a segment, $\overline{o_i(t)} = (\frac{x_i(t)}{NB})\, \varepsilon C_p$, which is considered to be reasonable and typically assumed in existing modeling studies of peer-assisted systems [22].

Instead of directly optimizing the prefetching and cache replacement of segments—which would lead to an overwhelming number of variables—we first derive the desired optimal *cache profile* $X^*(t) = (x_1^*(t), x_2^*(t), \ldots, x_V^*(t))$ in response to a given system-wide normal playback demand, represented by λ_2 and $(a_1, a_2, \ldots, a_V)$. Then, by examining the discrepancy between $X^*(t)$ and the current peer cache state $X(t)$, we can determine which segments are relatively undersupplied (to be prefetched) or oversupplied (to be replaced), so as to provide guidelines and insights toward optimal prefetching and cache replacement strategies.

4.1 Stochastic Queuing Analysis

We mainly consider two stochastic evaluation modes [3] based on queuing theory: the *waiting mode* wherein system performance is measured by the waiting time probability, and the *blocking mode* wherein system performance is measured by the request blocking probability. In the waiting mode, our supply-demand model of each segment corresponds to an $M/M/1$ system, by assuming Poisson arrival of normal playback requests for segment i with mean rate $\lambda_2 a_i$, and exponentially distributed service times with mean rate $\overline{o_i(t)}$. Then, the waiting time distribution for segment i in our supply-demand model is given as

$$Pr(W_i \le \tau) = \begin{cases} 1 - \frac{\lambda_2 a_i}{\overline{o_i(t)}}, & \tau = 0, \\ 1 - \frac{\lambda_2 a_i}{\overline{o_i(t)}} e^{(\lambda_2 a_i - \overline{o_i(t)})\tau}, & \tau > 0. \end{cases} \tag{13}$$

Using (13), we can derive the optimal cache profile $X^*(t)$ by maximizing the expected amount of normal playback requests that can be served within a certain desired delay bound τ, i.e., $\sum_{i=1}^{V} \lambda_2 a_i Pr(W_i \le \tau)$. However, such a waiting mode intrinsically assumes $\rho_i = \frac{\lambda_2 a_i}{\overline{o_i(t)}} < 1$, $i = 1, 2, \ldots, V$, for the system to be stable. Such an assumption may not always hold across all segments in the system, as the demand for certain highly popular segments could possibly exceed their corresponding service capacities in the system. Hence, we consider the blocking mode without such a restriction.

In the blocking mode, our supply-demand model of segment i corresponds to a Processor Sharing system $M/M/1/K_i - PS$ with a finite queuing capacity K_i representing the allowed number of simultaneous downloads for segment i, which depends on the service capacity $\overline{o_i(t)}$. For simplicity, we can let $K_i = \lceil \overline{o_i(t)} \rceil$ in order for each concurrent transmission of segment to be roughly served with the playback rate (i.e., one segment per unit time). If a request for segment i arrives when K_i is saturated, it is dropped and redirected to the servers. It is revealed by insensitivity results [23] that the blocking probability Pb_i for an $M/M/1/K_i - PS$ system is identical to that for the corresponding $M/M/1/K_i$ system, as given below:

$$Pb_i = \frac{\rho_i^{\lceil \overline{o_i(t)} \rceil}(1 - \rho_i)}{1 - \rho_i^{\lceil \overline{o_i(t)} \rceil + 1}}, \tag{14}$$

$$\rho_i = \frac{\lambda_2 a_i}{\overline{o_i(t)}} = \frac{\lambda_2 a_i NB}{\varepsilon C_p x_i(t)}, \quad i = 1, 2, \ldots, V. \tag{15}$$

As such, intrinsically, a blocking mode queuing model does not need to assume $\rho_i < 1$ for the system to be stable [16], it can be used to evaluate the performance of peer-assisted on-demand streaming systems under any supply-demand conditions across all segments, e.g., either deficit ($\rho_i > 1$) or surplus ($\rho_i < 1$).

To minimize server cost, we essentially need to maximize the *system-wide effective throughput* over all segments (i.e., the total number of unblocked segment requests served by peers):

$$\text{Maximize} \quad \sum_{i=1}^{V} \lambda_2 a_i (1 - Pb_i), \tag{16}$$

$$\text{Subject to :} \quad \sum_{i=1}^{V} x_i(t) = NB, \forall t, \tag{17}$$

$$0 \le x_i(t) \le N, i = 1, 2, \ldots, V, \tag{18}$$

$$x_i(t) \in \mathbb{N}, i = 1, 2, \ldots, V, \forall t, \tag{19}$$

where $\sum_{i=1}^{V} a_i = 1$. With a specific amount of demand λ_2, the problem above can be transformed to minimize the *system-wide blocking probability* as follows: differing from the cache hit ratio derived in a relevant cache redundancy model [8] with the constraint of the total cache size, our model and performance metric cohesively incorporate the constraints on both the upload bandwidth and storage of peers. The rationale is that even if a segment is cached by peer(s), there is no guarantee that the currently available bandwidth from peer(s) is sufficient to satisfy the video playback rate.

$$\text{Minimize} \quad \sum_{i=1}^{V} a_i Pb_i = \sum_{i=1}^{V} \frac{a_i \rho_i^{\lceil \overline{o_i(t)} \rceil}(1 - \rho_i)}{1 - \rho_i^{\lceil \overline{o_i(t)} \rceil + 1}} \tag{20}$$
$$\text{Subject to :} \quad \text{constraints}(17), (18), (19).$$

This is a nonlinear integer programming problem, which is NP-hard in general, and even harder to solve than integer linear programming problems [24]. More specifically, by plugging (15) into (20), the structure of the objective function can be viewed as a *sum-of-ratios problem in fractional programming*, which still remains a hard problem in the literature of global optimization [25]. Though it is mathematically challenging to reach global optimality, our construction of (20) analytically characterizes critical factors that are general in any caching and prefetching designs in peer-assisted on-demand streaming systems.

Due to the intractability of solving the general optimization problem, we seek to derive near-optimal solutions that reveal helpful insights. To simplify the problem, we utilize a reasonable upper bound of the system-wide blocking probability.

Lemma 1. *Under any given cache profile $X(t)$, the system-wide blocking probability has the following corresponding upper bound:*

$$\sum_{i=1}^{V} a_i Pb_i \le \sum_{i=1}^{V} \frac{a_i \rho_i}{1 + \rho_i}. \tag{21}$$

Proof. By limiting the queuing capacity of each segment to $K_i = 1, i = 1, 2, \ldots, V$, our supply-demand model of each segment $M/M/1/K_i - PS$ becomes the corresponding $M/M/1/1 - PS$ system. This essentially imposes more

stringent timing requirements for segment requests, and as such a limited queuing capacity results in lower chances for segment requests to wait, which alternatively means that segment requests become less tolerant to delay.

Specifically, the blocking probability of segment i under $M/M/1/1-PS$ becomes $\rho_i/(1+\rho_i)$. Then, we have the following relationship for both $\rho_i < 1$ and $\rho_i \geq 1$:

$$\frac{a_i\rho_i}{1+\rho_i} = \frac{a_i\rho_i^{\lceil\overline{o_i(t)}\rceil}(1-\rho_i)}{\rho_i^{\lceil\overline{o_i(t)}\rceil-1}-\rho_i^{\lceil\overline{o_i(t)}\rceil+1}} \geq \frac{a_i\rho_i^{\lceil\overline{o_i(t)}\rceil}(1-\rho_i)}{1-\rho_i^{\lceil\overline{o_i(t)}\rceil+1}} = a_iPb_i.$$

Note that given any cache profile $X(t)$, it is possible that certain segments may not be cached by any peers in the system, i.e., $\exists i$, $x_i(t)=0$, and $\overline{o_i(t)}=0$. In such cases, we have both $\rho_i/(1+\rho_i)=1$ and $Pb_i=1$, and hence the above still holds.

Then, since the above relationship holds for any segment $i=1,2,\ldots,V$, the summation over all segments gives (21). Specifically, when $x_i(t)=0,\forall i$, both sides of (21) degenerate to 1, which also captures the traditional server-based on-demand streaming systems without peer assistance, where all segment requests need to be satisfied by the servers. □

Since such an upper bound is able to preserve relevant factors in problem (20) while being more reflective of stringent timing requirements, it is reasonable to simplify problem (20) by minimizing such an upper bound.

Theorem 2. *In response to a given system-wide normal playback demand represented by λ_2 and $(a_1,a_2,\ldots,a_V)$, to minimize the upper bound of system-wide blocking probability given by Lemma 1, the optimal cache profile $X^*(t)$ should match the demand as $x_i^*(t)=NBa_i$, $i=1,2,\ldots,V$.*

Proof. To minimize the upper bound of system-wide blocking probability, we need to solve the following optimization problem:

$$\begin{aligned}\text{Minimize} \quad & \sum_{i=1}^{V}\frac{a_i\rho_i}{1+\rho_i}\\ \text{Subject to:} \quad & \text{constraints}(17),(18),(19).\end{aligned}$$

By relaxing the integer constraints (19) and the inequality constraints (18) (this does not undermine the optimality of the solution as discussed later), we can use Lagrangian multiplier and Karush-Kuhn-Tucker conditions [26] to solve it:

$$\Lambda = \sum_{i=1}^{V}\frac{a_i\rho_i}{1+\rho_i}+\omega\left(\sum_{i=1}^{V}x_i(t)-NB\right),$$

$$\frac{\partial\Lambda}{\partial x_i(t)} = \omega - \frac{\lambda_2a_i^2NB\varepsilon C_p}{(\varepsilon C_px_i(t)+\lambda_2a_iNB)^2}=0,$$

which gives $x_i^*(t)=(\frac{a_i}{\varepsilon C_p})\sqrt{\frac{\lambda_2NB\varepsilon C_p}{\omega}}-\frac{\lambda_2a_iNB}{\varepsilon C_p}$. Then, through the equality constraint (17), we can obtain the optimal cache profile $x_i^*(t)=NBa_i$, $i=1,2,\ldots,V$. □

Remark. Though the optimal cache profile above does not guarantee the inequality constraint $x_i(t)\leq N, i=1,2,\ldots,V$, it is still a "good" approximation. According to the heavy tail effect of content popularity observed in real-world on-demand streaming systems [27], a considerable portion of $x_i^*(t)$ in the optimal cache profile could satisfy $x_i^*(t)=NBa_i\leq N$. For those highly popular segments resulting in $x_i^*(t)=NBa_i>N$, we can simply adjust their corresponding $x_i^*(t)=N$, and shift the surplus portion to replicate other less popular ones. Such a near-optimal cache profile does not undermine the optimality of the solution. It conveys the fundamental principle for caching and prefetching designs in peer-assisted on-demand streaming systems: optimal prefetching and caching strategies should help the peer cache state evolve toward the optimal cache profile, which tends to balance the system-wide demand and supply.

Our aforementioned insights can lead to a general framework of optimal caching principle as described in Algorithm 1. Rather than a specific heuristic, this abstracts a *family* of prefetching and caching strategies toward the optimal cache profile. For instance, a popularity-based caching heuristic in [9] estimates the segment popularity and peer cache state through a distributed averaging scheme [28]; and the discrepancy between demand and supply is captured by the difference between normalized segment popularity and the average number of replicas. Another weight-based replication heuristic in [1] relies on the tracking server to provide the availability to demand ratio (ATD) for each video, which captures the discrepancy between supply and demand in a different form. These heuristics, albeit with different schemes to capture the system-wide supply-demand relationship and their discrepancy, essentially follow the key principle of our framework to better match demand and supply. While neither of them base their designs upon sound theoretical foundations, our general framework based on stochastic queuing analysis and global optimization can be used to characterize and understand the essence of such prefetching and caching designs.

Algorithm 1. A General Framework of Optimal Caching (with Prefetching) Principle

1: Obtaining information on system-wide demand and supply (either in a centralized or distributed manner as illustrated later):
   i) Estimating segment popularity a_i, $i=1,2,\ldots,V$;
   ii) Estimating the peer cache state $X(t)$, $i=1,2,\ldots,V$.
2: Examining the discrepancy between the peer cache state $X(t)$ and the optimal cache profile $X^*(t)$ in response to a_i, $i=1,2,\ldots,V$.
3: Balancing the system-wide demand and supply according to the discrepancy:
   i) Preferentially prefetch those under-supplied segments (e.g., popular yet less cached in the system) to meet the demand;
   ii) Preferentially replace those over-supplied segments (e.g., unpopular yet excessively cached in the system).

4.2 Performance Evaluation

We now quantitatively examine the optimal caching and prefetching principle represented by Theorem 2 and Algorithm 1 (henceforth referred to as *optimal*), its corresponding *upper bound* in Lemma 1, as well as one of the representative weight-based replication *heuristic* used in a real-world system [1]. The original heuristic in [1] is primarily limited to cache replacement at the video level, without allowing

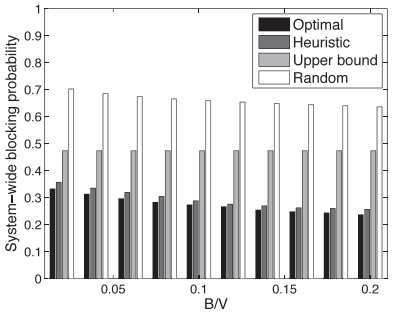


Fig. 5. System-wide blocking probability versus the increase of relative peer cache size, under a heavy relative system-wide workload of $\lambda_2/(\varepsilon C_p) = 0.9$ and a Zipf popularity distribution of segments.

prefetching. To perform a fair comparison, we modify it to a segment level and extend it to allow active prefetching, also using the hints based on the ATD information. Specifically, the lower the ATD of a segment, the relatively higher probability will be prefetched. In addition, we also include a *randomized* segment replication as a baseline performance benchmark, where peers randomly choose segments to replicate with an equal probability.

First, in Fig. 5, we examine the performance of the aforementioned candidates in terms of the system-wide blocking probability versus the increase of relative peer cache size B/V, under a heavy relative system-wide workload $\lambda_2/(\varepsilon C_p) = 0.9$ and a Zipf popularity distribution of segments, as typically observed in real-world Internet media workloads. We observe that: 1) The performance of all candidates improves as the relative peer cache size increases, which demonstrates the potential benefits of peer caching and prefetching to meet user demand by peers alone and thus offload the servers. 2) The optimal consistently outperforms the others. Meanwhile, by following the key principle to balance the system-wide demand and supply, the heuristic shows a comparable performance to the optimal. 3) The upper bound of system-wide blocking probability under the optimal cache profile is insensitive to the relative peer cache size as long as the relative system-wide workload is fixed, which represents a conservative estimation of system performance.

Second, in Fig. 6, we further investigate the performance of the aforementioned candidates by varying the relative system-wide workload $\lambda_2/(\varepsilon C_p)$ from a moderate level of 0.8 to an intense level of 1.25, under a setting of $B/V = 10\%$.

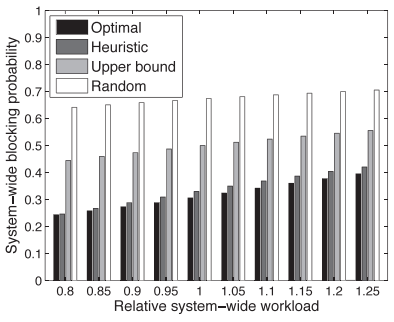


Fig. 6. System-wide blocking probability versus the increase of relative system-wide workload $\lambda_2/(\varepsilon C_p)$, under a relative peer cache size of $B/V = 10\%$ and a Zipf popularity distribution of segments.

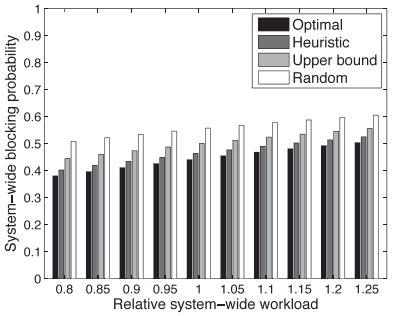


Fig. 7. System-wide blocking probability versus the increase of relative system-wide workload $\lambda_2/(\varepsilon C_p)$, under a relative peer cache size of $B/V = 10\%$ and stretched exponential distribution of segment popularity.

We observe that: 1) The system-wide blocking probability notably increases along with relatively more demand imposed on the entire system. However, the optimal and the heuristic are still able to mitigate a substantial portion of server load. 2) While the gap between the optimal and the heuristic is insignificant under moderate workloads, it becomes relatively more profound under heavy workloads. Meanwhile, the upper bound of system-wide blocking probability under the optimal cache profile becomes relatively tighter under heavy workloads.

Third, since a recent measurement study has argued that access patterns in many media workloads could follow the *stretched exponential* distribution [29] that deviates from Zipf, we further exercise the aforementioned candidates under such access patterns in Fig. 7. Under the same settings as in Fig. 6, we observe that the performance of the optimal and the heuristic notably decreases compared to that in Fig. 6; and their performance margin over the randomized one becomes less substantial. This is essentially because that the SE popularity distribution usually considers a much longer measurement duration (e.g., weeks or even months), which results in less skewed popularity distribution compared to Zipf, and thus potentially renders the caching benefit relatively restricted [29]. Nevertheless, we believe that the performance gain would be more profound over a shorter period of time and under larger cache sizes.

Finally, it is also essential to examine the system-wide performance with respect to the segment retrieval latency in our model. To this end, we define the system-wide segment retrieval latency as $F = \sum_{i=1}^{V} a_i(F_i^{nb} + F_i^b)$, where $F_i^{nb} = (1 - Pb_i)w_i$ is the expected latency for unblocked requests for segment i, and w_i is the mean sojourn time [16] of a job in the corresponding $M/M/1/K_i$ system based on Little's Law. In addition, $F_i^b = Pb_i\overline{\delta}$ is the expected latency for blocked requests for segment i, where $\overline{\delta}$ represents an expected tolerance threshold beyond which such blocked ones shall be satisfied by the servers as the last resort. Then, the system-wide relative retrieval latency can be obtained through normalizing F by $\overline{\delta}$, since relative values yield more relevant insights. Figs. 8, 9, and 10 plot the system-wide relative segment retrieval latency under the same setting as in Figs. 5, 6, and 7, respectively.

As the relative peer cache size increases in Fig. 8, the system-wide segment retrieval latencies of all the candidates have notably decreased, which implies an improved

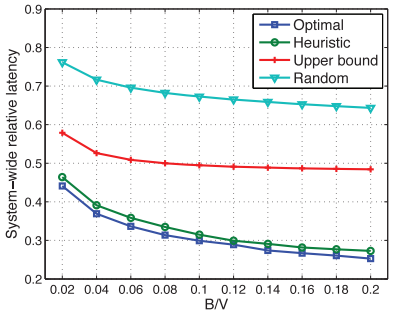


Fig. 8. System-wide relative segment retrieval latency versus the increase of relative peer cache size, under a heavy relative system-wide workload of $\lambda_2/(\varepsilon C_p) = 0.9$ and a Zipf popularity distribution of segments.

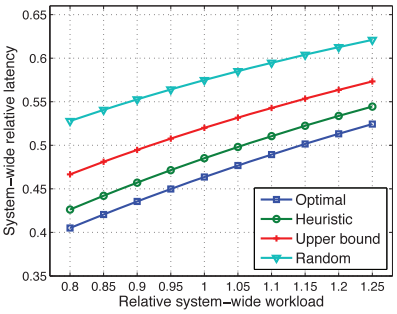


Fig. 10. System-wide relative segment retrieval latency versus the increase of relative system-wide workload $\lambda_2/(\varepsilon C_p)$, under a relative peer cache size of $B/V = 10\%$ and stretched exponential distribution of segment popularity.

streaming performance under more adequate caching capacities. On the other hand, as shown in Fig. 9, the performance could degrade under excessive system-wide workloads, and the performance gap between the optimal and the heuristic expands. A similar trend is also observed under the SE popularity distribution in Fig. 10, yet the performance margin enjoyed by the optimal and heuristic over the randomized alternative becomes less substantial, which is consistent with our previous discussions.

5 Concluding Remarks

Despite a large variety of heuristics proposed in existing peer-assisted on-demand streaming designs, there is a lack of sound theoretical principles and analytical insights to guide the design of two critical aspects: *serving requests* and *prefetching*. In this paper, we construct a new theoretical framework for peer-assisted on-demand streaming systems based on queuing models. In particular, we characterize demands with different levels of urgency through a priority queuing analysis, which both qualitatively and quantitatively justifies how service quality and user experience can be statistically guaranteed through service prioritization. Based on a fine-grained stochastic supply-demand model that we developed, we further investigate peer caching and prefetching as a global optimization problem. Our study not only provides insights in understanding the fundamental characterization of the demand, but also offers guidelines toward optimal prefetching and caching strategies in peer-assisted on-demand streaming systems.

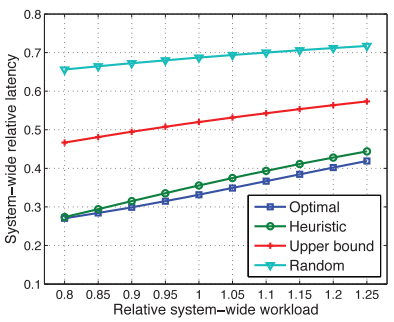


Fig. 9. System-wide relative segment retrieval latency versus the increase of relative system-wide workload $\lambda_2/(\varepsilon C_p)$, under a relative peer cache size of $B/V = 10\%$ and a Zipf popularity distribution of segments.

Acknowledgments

The research was support in part by a grant from The National Natural Science Foundation of China (NSFC) under grant No. 61103176, by a grant from the NSFC under grant No. 61133006, by a grant from the Independent Innovation Research Fund of Huazhong University of Science and Technology under grant No. 2011QN051. The authors would like to thank the editor and anonymous reviewers of the *IEEE Transactions on Computers* for their helpful comments and suggestions in improving the quality of the paper.

References

[1] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-Scale P2P-VoD System," *Proc. ACM SIGCOMM*, Aug. 2008.

[2] Y. Hall, P. Piemonte, and M. Weyant, "Joost: A Measurement Study," technical report, School of Computer Science, Carnegie-Mellon Univ., May 2007.

[3] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-Peer Video-on-Demand System: Design and Evaluation," *IEEE J. Selected Areas in Comm.*, vol. 25, no. 9, pp. 1706-1716, Dec. 2007.

[4] C. Huang, J. Li, and K. Ross, "Can Internet Video-on-Demand Be Profitable?" *Proc. ACM SIGCOMM*, Aug. 2007.

[5] Y. Boufkhad, F. Mathieu, F. de Montgolfier, D. Perino, and L. Viennot, "Achievable Catalog Size in Peer-to-Peer Video-on-Demand Systems," *Proc. Int'l Workshop Peer-to-Peer Systems*, Feb. 2008.

[6] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson, "Analysis of Bittorrent-Like Protocols for On-Demand Stored Media Streaming," *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems*, June 2008.

[7] Y. Choe, D. Schuff, J. Dyaberi, and V. Pai, "Improving VoD Server Efficiency with Bittorrent," *Proc. Int'l Conf. Multimedia*, Sept. 2007.

[8] L. Guo, S. Chen, and X. Zhang, "Design and Evaluation of a Scalable and Reliable P2P Assisted Proxy for On-Demand Streaming Media Delivery," *IEEE Trans. Knowledge and Data Eng.*, vol. 18, no. 5, pp. 669-682, May 2006.

[9] W. Yiu, X. Jin, and S. Chan, "VMesh: Distributed Segment Storage for Peer-to-Peer Interactive Video Streaming," *IEEE J. Selected Areas in Comm.*, vol. 25, no. 9, pp. 1717-1731, Dec. 2007.

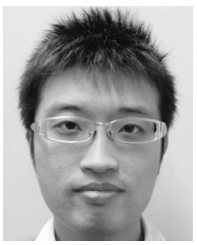
[10] Y. He, G. Shen, Y. Xiong, and L. Guan, "Optimal Prefetching Scheme in P2P VoD Applications with Guided Seeks," *IEEE Trans. Multimedia*, vol. 11, no. 1, pp. 138-151, Jan. 2009.

[11] S. Tewari and L. Kleinrock, "Proportional Replication in Peer-to-Peer Networks," *Proc. IEEE INFOCOM*, Apr. 2006.

[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. IEEE INFOCOM*, Mar. 1999.

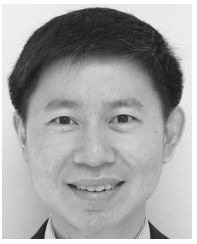
[13] M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and Analysis of a Streaming Media Workload," *Proc. Third USENIX Symp. Internet Technologies and Systems*, Mar. 2001.

[14] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang, "The Stretched Exponential Distribution of Internet Media Access Patterns," *Proc. ACM Symp. Principles of Distributed Computing (PODC '08),* Aug. 2008.
[15] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An Analysis of Live Streaming Workloads on the Internet," *Proc. ACM SIGCOMM Conf. Internet Measurement (IMC),* Oct. 2004.
[16] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications.* Wiley-Interscience, 2006.
[17] X. Yang and G. de Veciana, "Service Capacity of Peer to Peer Networks," *Proc. IEEE INFOCOM,* Mar. 2004.
[18] W. Feng, M. Kowada, and K. Adachi, "Analysis of a Multi-Server Queue with Two Priority Classes and (M, N)-Threshold Service Schedule I: Non-Preemptive Priority," *Int'l Trans. Operational Research,* vol. 7, no. 6, pp. 653-671, 2000.
[19] Y. Jiang, C. Tham, and C. Ko, "An Approximation for Waiting Time Tail Probabilities in Multi-Class Systems," *IEEE Comm. Letters,* vol. 5, no. 4, pp. 175-177, Apr. 2001.
[20] B. Walke, "Improved Bounds and an Approximation for a Dynamic Priority Queue," *Proc. Third Int'l Symp. Measuring, Modelling and Evaluating Computer Systems,* Oct. 1977.
[21] C. Wu, B. Li, and S. Zhao, "Exploring Large-Scale Peer-to-Peer Live Streaming Topologies," *ACM Trans. Multimedia Computing, Comm. and Applications,* vol. 4, no. 3, pp. 175-177, 2008.
[22] Z. Ge, D. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling Peer-Peer File Sharing Systems," *Proc. IEEE INFOCOM,* Apr. 2003.
[23] D. Burman, "Insensitivity in Queueing Systems," *Advances in Applied Probability,* vol. 13, pp. 846-859, 1981.
[24] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, "Nonlinear Integer Programming," *50 Years of Integer Programming 1958-2008: The Early Years and State-of-the-Art Surveys,* Springer-Verlag, 2009.
[25] S. Schaible and J. Shi, "Fractional Programming: The Sum-of-Ratios Case," *Optimization Methods and Software,* vol. 18, no. 2, pp. 219-229, 2003.
[26] D. Bersekas, *Nonlinear Programming.* Athena Scientific, 1999.
[27] B. Cheng, L. Stein, H. Jin, and Z. Zhang, "Towards Cinematic Internet Video-on-Demand," *Proc. Third ACM SIGOPS/EuroSys European Conf. Computer Systems,* Apr. 2008.
[28] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray, "Asynchronous Distributed Averaging on Communication Networks," *IEEE/ACM Trans. Networking,* vol. 15, no. 3, pp. 512-520, June 2007.
[29] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang, "The Stretched Exponential Distribution of Internet Media Access Patterns," *Proc. ACM Symp. Principles of Distributed Computing (PODC),* Aug. 2008.



Fangming Liu (S'08-M'11) received the BEngr degree in 2005 from the Department of Computer Science and Technology, Tsinghua University, Beijing, China; and the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology in 2011. He is currently an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. In 2007, he worked as a research assistant at the Department of Computer Science and Engineering, Chinese University of Hong Kong. From August 2009 to February 2010, he was a visiting student at the Department of Electrical and Computer Engineering, University of Toronto, Canada. Since 2010, he has also been collaborating with the ChinaCache Content Delivery Network Research Institute in Tsinghua University. His research interests are in the areas of peer-to-peer networks, rich-media distribution, cloud computing, and large-scale datacenter networking. He is a member of the IEEE and the IEEE Communications Society, and a member of the ACM.

Bo Li (S'89-M'92-SM'99-F'11) received the BEngr degree in computer science from Tsinghua University, Beijing, and the PhD degree in electrical and computer engineering from the University of Massachusetts at Amherst. He is a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He was with IBM Networking System, Research Triangle Park, between 1993 and 1996. He was an adjunct researcher at Microsoft Research Asia (MSRA) from 1999 to 2006, where he spent his sabbatical leave from 2003 to 2004. He has made original contributions on Internet proxy placement, capacity provisioning in wireless networks, routing in WDM optical networks, and Internet video streaming. He is best known for a series of works on a system called Coolstreaming (Google entries over 1,000,000 in 2008 and Google scholar citations over 800), which attracted millions of downloads and was credited as the first large-scale Peer-to-Peer live video streaming system in the world. His recent work on the peer-assisted online hosting system, FS2You (2007-2009) (Google entries 800,000 in 2009) has also attracted millions of downloads worldwide. He has been an editor or guest editor for 17 IEEE/ACM journals and magazines. He was the co-TPC chair for IEEE Infocom '04. He is a fellow of the IEEE.



Baochun Li (S'98-M'00-SM'05) received the BEngr degree in 1995 from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, and the MS and PhD degrees in 1997 and 2000 from the Department of Computer Science, University of Illinois at Urbana-Champaign. Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently a professor. He has been holding the Bell University Laboratories Endowed Chair in computer engineering since August 2005. In 2000, he was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems. In 2009, he was the recipient of the Multimedia Communications Best Paper Award from the IEEE Communications Society. His research interests include large-scale multimedia systems, peer-to-peer networks, applications of network coding, and wireless networks. He is a senior member of the IEEE, and a member of the ACM.

Hai Jin received the PhD degree in computer engineering from Huazhong University of Science and Technology (HUST), China, in 1994. He is a Cheung Kung Scholars chair professor of computer science and engineering at the Huazhong University of Science and Technology. He is now the dean of the School of Computer Science and Technology at HUST. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. He is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientist of National 973 Basic Research Program Project of Virtualization Technology of Computing System. He is the member of Grid Forum Steering Group (GFSG). He has coauthored 15 books and published more than 400 research papers. His research interests include computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security. He is the steering committee chair of International Conference on Grid and Pervasive Computing (GPC), Asia-Pacific Services Computing Conference (APSCC), International Conference on Frontier of Computer Science and Technology (FCST), and Annual ChinaGrid Conference. He is a member of the steering committee of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), the IFIP International Conference on Network and Parallel Computing (NPC), and the International Conference on Grid and Cooperative Computing (GCC), International Conference on Autonomic and Trusted Computing (ATC), International Conference on Ubiquitous Intelligence and Computing (UIC). He is a senior member of the IEEE and a member of the ACM.