This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Proactive Serving Decreases User Delay Exponentially: The Light-Tailed Service Time Case

Shaoquan Zhang, Longbo Huang, Minghua Chen, *Senior Member, IEEE*, and Xin Liu

***Abstract*— In online service systems, the delay experienced by users from service request to service completion is one of the most critical performance metrics. To improve user delay experience, recent industrial practices suggest a modern system design mechanism: *proactive serving*, where the service system predicts future user requests and allocates its capacity to serve these upcoming requests proactively. This approach complements the conventional mechanism of capability boosting. In this paper, we propose queuing models for online service systems with proactive serving capability and characterize the user delay reduction by proactive serving. In particular, we show that proactive serving decreases average delay *exponentially* (as a function of the prediction window size) in the cases where service time follows light-tailed distributions. Furthermore, the exponential decrease in user delay is robust against prediction errors (in terms of miss detection and false alarm) and user demand fluctuation. Compared with the conventional mechanism of capability boosting, proactive serving is more effective in decreasing delay when the system is in the light-load regime. Our trace-driven evaluations demonstrate the practical power of proactive serving: for example, for the data trace of light-tailed YouTube videos, the average user delay decreases by 50% when the system predicts 60 s ahead. Our results provide, from a queuing-theoretical perspective, justifications for the practical application of proactive serving in online service systems.**

***Index Terms*— Proactive serving, queuing model, user delay.**

I. Introduction

THE fast growing number of personal devices with Internet access, *e.g.*, smart mobile devices, has led to the blossoming of diverse online service systems, such as cloud computing, cloud storage, online social networks, mobile Internet access, and a variety of online communication applications. In online service systems, delay experienced by users from service request to service completion is one of the most critical performance metrics. For example, experiments at Amazon showed that every 100-millisecond increase in the loading time of Amazon.com decreased revenues by 1% [3]. Google also found that an extra 0.5 seconds in search page generation time decreased traffic by 20% [3].

Traditionally, to reduce user delay and to improve quality of experience, service providers often resort to *capacity boosting*, *i.e.*, increasing the service capacity by deploying more servers. However, such a mechanism may be expensive as it needs to provision for peak demand, which results in low average utilization due to the bursty nature of service requests, especially when user arrivals are time-varying.

Recent industry practices suggest *proactive serving*, *i.e.*, serving upcoming requests before they arrive, as a modern approach for reducing user delay. Proactive serving is based on a key observation: many service requests are predictable. This technique has been widely used in computer systems, for example in cache pre-loading and command prefetching.[1] Similarly, in cloud service systems, it is common to have predictable service requests. For example, in cloud computing platforms, service jobs, such as indexing, page ranking, backing up, crawling, and performing maintenance, are often predictable. In fact, in an industrial-grade cloud computing system, researchers observe a significant portion, *e.g.*, up to 76%, of the workload to be periodic and thus predictable [4]. Furthermore, individual user behaviors often follow predictable patterns [5]. For instance, if a user watches sports news regularly in the morning, then such content can be pre-loaded to the user's device.

Recently, Amazon launched *Amazon Silk*, a mobile web browser for its Kindle Fire [5]. All web traffic from the browser goes through the Amazon cloud and is managed by the servers in the cloud. Based on cached user traffic, the cloud uses machine learning techniques to predict what users will browse next. When service in the cloud is available, web pages that are likely to be requested are pre-loaded to users' tablets. Thus, when a user clicks on the corresponding content, it loads instantly, reducing user delay to zero. This technique speeds up request responses and improves user browsing experience.

Manuscript received March 26, 2015; revised December 28, 2015 and July 4, 2016; accepted August 8, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Eryilmaz. This work was supported in part by the National Basic Research Program of China under Project 2013CB336700 and the University Grants Committee of the Hong Kong Special Administrative Region, China, under Area of Excellence Grant under Project AoE/E-02/08 and the General Research Fund under Grant 14209115. The work of L. Huang was supported by the National Basic Research Program of China under Grants 2011CBA00300 and 2011CBA00301, the National Natural Science Foundation of China under Grants 61033001, 61361136003, and 61303195, and the China Youth 1000-Talent Grant. The work of X. Liu was supported by NSF under Grants CNS-1547461, CNS-1457060, and CCF-1423542. This paper was presented in part at ACM SIGMETRICS 2014 as a poster paper [1] and at ACM MAMA 2015 [2]. *(Corresponding authors: Shaoquan Zhang; Minghua Chen.)*

S. Zhang and M. Chen are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: imarch1.zhang@gmail.com; minghua@ie.cuhk.edu.hk).

L. Huang is with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China.

X. Liu was with Microsoft Research Asia, Beijing 100080, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2016.2607840

[1]Cache pre-loading means that the system can send contents to users' caches before users request them. Command pre-fetching means that the system can run commands before they are actually called.

1063-6692 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

The above mentioned application scenarios suggest proactive serving as a new design mechanism for reducing user delay: based on user request arrival prediction, the system can allocate its capacity proactively and pre-serve upcoming requests, reducing the delay experienced by users. This observation naturally leads to two fundamental questions:

- How much can we reduce user delay by proactive serving?
- How does proactive serving compare to capacity boosting in reducing user delay?

Motivated by these open questions, in this paper, we investigate the fundamentals of proactive serving from a queuing theory perspective. In particular, we study proactive serving with a prediction window of size ω, in which one has the ability to predict upcoming requests and serve them when capacity allows. We investigate how proactive serving reduces user delay as a function of ω. We also consider the case of imperfect prediction. As discussed in Section II, this work differs from all existing ones in its model and problem setting. It provides theoretical foundation for using proactive serving to reduce user delay.

A. Challenges

We address two technical challenges in our study. First, a generic approach to obtain average user delay is to model the queuing system with proactive serving capability (based on perfect or imperfect prediction) using a multi-dimensional Markov chain. Then we compute its steady-state distribution and subsequently average user delay. However, it is highly non-trivial to derive closed-form expressions for steady-state distributions of multi-dimensional Markov chains, and it is hard to generalize this approach to scenarios with imperfect prediction. We address this challenge by developing a new approach that relates the user delay distribution with proactive serving to that without proactive serving. This allows us to obtain the delay distribution without proactive serving, which is usually a less complicated task, and then to derive the desired one with proactive serving. This simple approach reveals insights into the delay reduction enabled by proactive serving. More importantly, the approach can be generalized to various queuing models.

Second, even with our new approach, it is still non-trivial to characterize user delay with proactive serving in the presence of prediction errors. We carefully model the system behavior under two types of prediction errors, namely miss detection and false alarm, as a priority queue. Through an involved derivation, we obtain closed-form expressions for the average user delay. The expressions allow us to reveal the exact relationship of the delay performance and system design parameters.

B. Contributions

We make the following contributions.

$\triangleright$ In Section III, we present the first set of queuing models for service systems with proactive serving capability. These models allow us to leverage queuing theory tools to characterize the delay reduction by proactive serving.

$\triangleright$ In Section IV, for stable M/M/1 queuing system with proactive serving capability, we show that the average user delay decreases *exponentially* in the prediction window size ω. Furthermore, based on the insights from the M/M/1 systems, we prove that exponential delay decrement holds for the more general G/G/1 queuing system.

$\triangleright$ In Section V, we study the impact of imperfect prediction on delay reduction. We consider two types of prediction errors: miss detection and false alarm. We construct queuing models for the study and characterize the average user delay under proactive serving with imperfect prediction. We show that in the presence of miss detection, average user delay still decreases exponentially in the prediction window size, but it converges to a positive constant determined by the fraction of miss detections (instead of converging to zero as would happen with perfect prediction). Meanwhile, in the presence of false alarm, we show that there exists an "effective service rate" determined by both the system service rate and the fraction of false alarms. User delay decreases exponentially to zero as prediction window size increases if the actual user request arrival rate is smaller than the effective service rate; and otherwise it decreases to a positive constant determined by the fraction of false alarms.

$\triangleright$ In Section VI, by comparing proactive serving to capacity boosting, we show that proactive serving is more effective in reducing user delay in a light-load regime.

$\triangleright$ In Section VII, we evaluate the performance of proactive serving using simulations based on real-world traces. Specifically, for the data trace of light-tailed Youtube videos, user delay decreases by 50% when the system can predict 60 seconds ahead. For the data trace of heavy-tailed Youtube videos, user delay decreases by 50% when the system can predict 84 seconds ahead. We note that there are efficient mechanisms for predicting user requests in the near future for industrial-grade VoD systems [6].

II. Related Work

In [7], the authors study how prediction can be used to facilitate queue admission control, *i.e.*, which requests should be redirected away (up to certain rate) in order to minimize the average queue length. They find that, in the heavy-load regime (arrival rate $\lambda \to 1$), when the size of look-ahead window is $O(\log \frac{1}{1-\lambda})$, the achieved average queue length is the same as that when we have complete future knowledge. Similar idea is also adopted in [8] to reduce waiting time for patients in Emergency Department by proactively managing the admission control. These two works use future information to control admission of requests into the system, instead of pooling idle service capacity to proactively serve future requests as we explore in this paper.

When the system can pre-serve upcoming requests based on prediction, the authors in [9] consider predictive scheduling in controlled queuing systems. They propose the *Predictive Backpressure* algorithm to achieve the optimal utility performance. With proactive serving, the authors in [10] show that the probability of server outage in wireless networks decreases exponentially with the size of look-ahead window. They also show that appropriate use of prediction by primary users

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

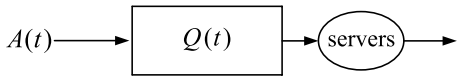


Fig. 1. A single queue service system.

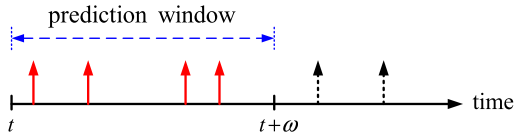


Fig. 2. Prediction model: Each upright arrow represents a request arrival. At time t, the system knows in $(t, t+\omega)$ the request arrival epochs (red solid arrows) and the corresponding users who generate these requests by its prediction mechanism.

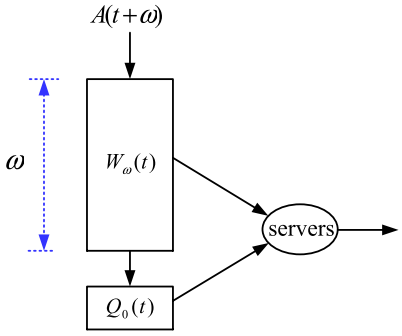


Fig. 3. Service system with perfect prediction: $Q_0(t)$ represents the queue of the requests that have arrived at the system and are waiting for service at time t. $W_\omega(t)$ is the prediction window of size ω. Each arrived user request first goes through the prediction window $W_\omega(t)$ and then enters the queue $Q_0(t)$. The servers can observe and serve the requests in both $Q_0(t)$ and $W_\omega(t)$.

can improve the gain of the secondary network at no cost of primary users in cognitive networks. The authors in [11] explore the idea of proactive serving in data networks to shape user demand so that the time average expected cost incurred by the service provider is minimized. In [12], the authors combine smart pricing and proactive content caching in mobile service system and show that the combination can increase the profit of the service provider while at the same time reducing end-users' costs. In [13] and [14], the authors provide a prediction model and develop techniques for proactively serving web contents. What differentiates our work from theirs is that we study the effect of proactive serving on decreasing user delay.

In addition, there have been many works on proactively serving mobile content based on predicting mobile user traffic and mobility patterns [15]–[19]. In [20], the authors present a system architecture for mobile pre-fetching where informed pre-fetching is structured as a library to which any mobile application may link. For general computer systems, the authors in [21]–[23] explore prediction and pre-fetching for file systems, databases, and DRAM respectively. These works mainly focus on practical system design and call for theoretical investigation.

III. Model

A. Service System Without Proactive Serving

Consider a service system shown in Fig. 1. Incoming user requests arrive at the system according to a continuous process $\{A(t)\}_t$. For all t, $A(t) \in \{0,1\}$ where $A(t) = 1$ if a user request arrives at t and $A(t) = 0$ otherwise. Servers in the system provide service upon user requests. When a user request arrives, the request will be served if there is idle service capacity. Otherwise, the request waits in the queue $Q(t)$ until service is available. The request will leave the system upon service completion. We define user delay as the time from user request arrival till its departure.

Traditionally, queuing theory has been applied to model such a system and study its performance. In particular, queuing theoretic analysis suggests boosting system capacity as a principled mechanism to reduce average user delay. For example, one can use standard M/M/1 queuing model to represent the service system in Fig. 1, and it is well known that average user delay decreases inverse-proportionally in service capacity.

B. Service System With Proactive Serving

We now consider a service system that can proactively serve upcoming user requests based on arrival prediction. For the ease of presentation, we consider perfect prediction of arrivals in this section and in Section IV, and we study imperfect prediction in Section V.

As shown in Fig. 2, we assume that the system can predict user request arrivals ω amount of time ahead. That is, at time t, the system knows exactly in $(t, t+\omega)$ the request arrival epochs and the corresponding users who generate the requests.[2] We assume that the system does not know the service time, *i.e.*, how long it takes to serve a request. The rationale behind the assumption is that even if the system knows the workload of the request (*e.g.*, size of a video), it is still difficult for the system to know exactly how long it takes to serve the request because of the dynamics in server capabilities and the time-varying available bandwidth between servers and users.

Based on arrival prediction, the system can serve upcoming requests proactively. The user requests that are pre-served will not enter the system. For the service system without proactive serving capability, servers remain idle when there is no request in the system. In contrast, for the service system with proactive serving capability, server capacity can be allocated to serve upcoming requests when there is no request in the system.

We depict the service system with proactive serving capability under perfect prediction in Fig. 3. In the figure, $Q_0(t)$ denotes the queue that stores the requests that have arrived at the system and are waiting for service at time t. $W_\omega(t)$ denotes the prediction window of size ω. Each user request first goes through the prediction window $W_\omega(t)$ and then enters the queue $Q_0(t)$. The servers can serve the requests in both $Q_0(t)$ and $W_\omega(t)$. We remark that each request entering $W_\omega(t)$ will transit to $Q_0(t)$ after exactly ω amount of time, if it has not been pre-served completely before that. Requests

[2]For service systems that are content-centric, we consider the timescale at which the content refresh/update is much larger as compared to the prediction horizon. Thus, contents delivered proactively to a customer will not be outdated by the time the customer actually requests them.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

will *not* queue up in $W_\omega(t)$. Thus $W_\omega(t)$ should be viewed as a pipe rather than a queue. User delay corresponds to the time that the request spends in $Q_0(t)$ and with the server, and it does not include the time spent in $W_\omega(t)$. Slightly abusing the notations, we sometimes use $Q_0(t)$ and $W_\omega(t)$ to denote their corresponding sizes respectively. For example, $W_\omega(t)$ can represent the number of arrivals within the prediction window that have not been pre-served completely at time t.

In this paper, we assume no constraints on the user side when proactive serving is conducted. For example, when cache pre-loading scenario is considered, we assume that there is no limit on user's cache size and the cache can be always accessed.

C. Queuing Models for Service System With Proactive Serving Capability

In this paper, we are interested in understanding the fundamental benefit of proactive serving on user delay reduction. For this purpose, we extend the classical queuing model to capture the proactive serving behaviors.

- In the classical queuing model, requests arrive randomly at the system. If all servers are busy, requests will wait in the queue for service. Servers serve requests according to a service policy. The amount of time to serve a request is random.
- In our extended model, there is also a prediction window (modeled as a pipe). Each request first goes through the pipe before entering the queue. Servers can serve requests in either the queue or the pipe according to a service policy.

In the classical queuing theory, Kendall's notation is widely used to describe a queuing system. We extend it to describe the service system with proactive serving capability as

$$\mathrm{A/S}/k^{[\omega]}/\mathrm{POLICY}.$$

Here A represents the distribution of request inter-arrival time, S represents the distribution of service time, k is the number of servers, and POLICY denotes the service discipline, such as First-Come-First-Served (FCFS). $[\omega]$ denotes that the system can predict upcoming arrivals ω amount of time ahead and serve them proactively.

For example, one can use M/M/$1^{[\omega]}$/FCFS to model the system in Fig. 3. Both the inter-arrival time and the service time follow exponential distributions. There is a single server in the system. The system can predict upcoming arrivals ω amount of time ahead and serve them proactively. When the server becomes idle, it first examines $Q_0(t)$. If $Q_0(t) > 0$, the server serves the request at the head of $Q_0(t)$. If $Q_0(t) = 0$, the server then checks the prediction window $W_\omega(t)$ and pre-serves the earliest request in it.

A queuing system is *stable* if arrivals happen slower than service completions, and is unstable otherwise. Unstable systems do not have steady-state distributions and consequently do not have well-defined average user delay. Thus we only study user delay for stable queuing systems. To focus on characterizing the benefit of proactive serving and avoid the complication of service policy design, we assume FCFS as the service policy in the rest of the paper, unless mentioned otherwise. Our analysis shows that even under the simple scheduling policy FCFS, proactive serving is very effective in reducing user delay.

IV. Proactive Serving With Perfect Prediction

In this section, we start from the simple M/M/$1^{[\omega]}$ model and show that proactive serving can decrease the average user delay exponentially. Based on the insights from the M/M/$1^{[\omega]}$ model, we extend the analysis to the general G/G/$1^{[\omega]}$ model and show that exponential delay decrement holds regardless of inter-arrival time and service time distributions.

We remark that we have also obtained similar exponential user-delay decrement results for the Markovian/Geo/$1^{[\omega]}$ model, in which the user request arrival rate is time-varying, governed by an underlying Markov process. We skip the details here due to the space limitation and refer interested readers to our technical report [24].

A. Average User Delay of M/M/$1^{[\omega]}$

In an M/M/$1^{[\omega]}$ model, there is a single server. User requests arrive according to a Poisson process $\{A(t)\}_t$ with rate λ, and service times of requests are independent and identically distributed according to an exponential distribution with parameter μ. The system can predict upcoming arrivals ω amount of time ahead and serve them proactively. Define D^ω as the user delay and $\rho = \frac{\lambda}{\mu}$.

When $\omega = 0$, *i.e.*, no proactive serving, the M/M/$1^{[\omega]}$ model reduces to the classical M/M/1 queuing model. It's well known that the average user delay is given by

$$E\left[D^0\right] = \frac{1}{\mu - \lambda}. \tag{1}$$

The probability density function of D^0 is also known as [25]

$$f_{D^0}(t) = (\mu - \lambda)e^{-(\mu-\lambda)t}, \quad t \geq 0. \tag{2}$$

To characterize the average user delay with proactive serving, *i.e.*, $E[D^\omega]$, a generic approach is to discretize the system and then use a multi-dimensional Markov chain to model the number of requests in queue and the prediction window, where user requests passing through the window can be captured by moving from one state to its subsequent state in the Markov chain. If we can compute the steady-state distribution of the multi-dimensional Markov chain, then we can compute the average number of user requests in the queue, and consequently the average user delay by using Little's Law. However this approach suffers from two limitations. One is that it is difficult to derive the stationary distribution of multi-dimensional Markov chain. The other is that it is hard to generalize the method to study more complex models. Detailed discussions can be found in Appendix B.

Instead of applying the generic method, we leverage problem structure to analyze the average user delay. To analyze the user delay for M/M/$1^{[\omega]}$, we first prove that $Q^{sum}(t) \triangleq Q_0(t) + W_\omega(t)$ evolves the same as an M/M/1 queue with a properly initialized queue. Based on this observation, the distribution of user delay under proactive serving,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

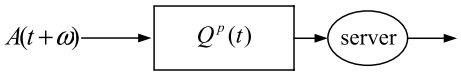


Fig. 4. M/M/1: The arrival process is $\{A(t+\omega)\}_t$. Service times of requests are independent and identically exponentially distributed with mean $1/\mu$. The initial value $Q^p(0)$ is $|A(0:\omega)| + Q_0(0)$. The service policy is FCFS.

i.e., D^ω, turns out to be a "shifted" version of that of user delay without proactive serving as shown in (2). Once we know the distribution of D^ω, we can compute the average user delay $E[D^\omega]$. More interestingly, this approach can be applied to various queuing models.

To proceed, consider $Q_0(t)$ and $W_\omega(t)$ as a group. The request arrival process of the group is $\{A(t+\omega)\}_t$. The time to serve a request is exponentially distributed. The server serves the requests in the group according to the FCFS policy. This essentially mimics an M/M/1 queue. Intuitively, the size of the group evolves statistically the same as the queue size of the M/M/1 queue. The following lemma confirms this observation.

Lemma 1: Define $Q^p(0) = |A(0:\omega)| + Q_0(0)$, where $A(0:\omega) = \{A(\tau), 0 < \tau \le \omega\}$ is the set of arrivals from time 0 to time ω and $|A(0:\omega)|$ is its size. $Q^p(t)$ is an M/M/1 queue with initial value $Q^p(0)$ as shown in Fig. 4. We have

$$Q^{sum}(t) = Q^p(t), \quad \text{for all } t.$$

Proof: Under the condition of $Q^p(0) = |A(0:\omega)| + Q_0(0)$ and $Q^{sum}(0) = Q^p(0)$. By time t, $Q^{sum}(t)$ and $Q^p(t)$ accept the same set of arrivals. Because both queues contain the same set of arrivals and adopt the same queuing discipline, $Q^{sum}(t)$ and $Q^p(t)$ have the same sequence of departures up to time t. As a result, $Q^{sum}(t) = Q^p(t)$. ■

Lemma 1 reveals a useful observation: the total time that a user request spends in M/M/1$^{[\omega]}$, *i.e.*, the sum of those in the prediction window $W_\omega(t)$, the queue $Q_0(t)$, and with the server, is statistically the same as that in an M/M/1 queue. As discussed at the end of Section III-B, the time spent in $W_\omega(t)$ is excluded from the user delay calculation. Then the user delay in M/M/1$^{[\omega]}$, *i.e.*, D^ω, has the same distribution as $\max(0, D^0 - \omega)$. We leverage this observation to obtain the distribution of D^ω for M/M/1$^{[\omega]}$ in the following lemma.

Lemma 2: Let $f_{D^\omega}(t)$ be the probability density function of D^ω. We have

$$f_{D^\omega}(t) = f_{D^0}(t+\omega), \quad \forall t > 0,$$

and

$$\Pr(D^\omega = 0) = \int_0^\omega f_{D^0}(t)dt = 1 - e^{-(\mu-\lambda)\omega}.$$

Proof: Because the arrival process $A(t)$ is stationary, the delay distribution of arrivals in Q^p is the same as that of M/M/1 in (2).

By Lemma 1, $Q^{sum}(t) = Q^p(t)$ for any t. Then a same request in $A(t)$ spends the same amount of time in Q^{sum} and Q^p. As shown in Fig. 6, if not getting pre-served, a request goes through the prediction window $W_\omega(t)$ before it enters Q_0, which costs ω amount of time. If a request spends T amount of time in Q^p, it will spend $[T-\omega]^+$ amount of time in Q^0. ($[T-\omega]^+ = 0$ means the request is pre-served completely.) Therefore, the delay distribution of requests in Q_0 is a "shifted" version of that of those in Q^p. ■

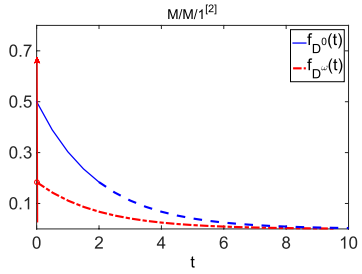


Fig. 5. Probability density function (PDF) of user delay with/without future prediction: The PDF under perfect prediction (the vertical arrow at the origin and the dash curve) can be obtained by shifting the PDF without prediction (dash curve) ω units left, where $\omega = 2$.

An illustrating example of the distribution derived in Lemma 2 is shown in Fig. 5. Note that the arguments in the proofs of Lemma 1 and 2 are also used in our sister paper [9]. We present the proofs here for completeness.

Although Lemma 1 and 2 are established for the M/M/1$^{[\omega]}$ model, they can be extended to the general G/G/1$^{[\omega]}$ model as shown in the following corollary, which will be used in the next subsection.

Corollary 1: In the G/G/1$^{[\omega]}$ model, the user delay distribution, denoted as $f^G_{D^\omega}(t)$, can be obtained from that of the G/G/1 model, denoted as $f^G_{D^0}(t)$, as follows

$$f^G_{D^\omega}(t) = f^G_{D^0}(t+\omega), \quad \forall t > 0,$$

and

$$\Pr(D^\omega = 0) = \int_0^\omega f^G_{D^0}(t)dt.$$

The proof is similar to that of Lemma 2 and is relegated to Appendix A.

Lemma 2 and Corollary 1 allow us to obtain the distribution of D^ω from that of D^0, which usually is well studied in queuing theory. Based on Lemma 2, for the M/M/1$^{[\omega]}$ model, we obtain the average request delay $E[D^\omega]$ as follows.

Theorem 1: Assume $\mu > \lambda$. The average user delay of M/M/1$^{[\omega]}$ model with perfect prediction is given by

$$E[D^\omega] = \frac{1}{\mu-\lambda} e^{-(\mu-\lambda)\omega}. \tag{3}$$

Proof: Based on Lemma 2,

$$E[D^\omega] = \int_0^\infty t \cdot f(t+\omega)dt = \frac{1}{\mu-\lambda} e^{-(\mu-\lambda)\omega}.$$

■

Theorem 1 says that the average user delay decreases exponentially in the prediction window size ω. This indicates that a little bit of future information can improve user delay experience significantly. In particular, this result suggests that, for Amazon's new mobile web-browser described in Section I, if the Amazon cloud can predict upcoming web requests actually, the request response time can be reduced significantly.

From Theorem 1, in the heavy-load regime where the arrival rate λ is close to the service rate μ, (3) can be approximated

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

by $\frac{1}{\mu-\lambda}-\omega$ when ω is small, which is linear in ω. In contrast, when the system is in the light-load regime, (3) decreases exponentially in ω. This indicates that proactive serving is more effective in decreasing delay in the light-load regime than in the heavy-load regime. The reason is as follows. In the light-load regime, the number of requests that enter $Q_0(t)$ for service is small and it is empty most of time. As such, most of the service capacity can be spared to serve upcoming requests. Consequently, many requests are served proactively and thus experience zero delay. In contrast, in the heavy-load regime, the server is busy with serving requests in $Q_0(t)$ most of the time. As a result, a request has little chance to be served proactively, especially when the prediction window size is small. Therefore, proactive serving has limited delay reduction capability.

B. Average User Delay of G/G/1$^{[\omega]}$

In this subsection, we extend the analysis to the general G/G/1$^{[\omega]}$ model. In G/G/1$^{[\omega]}$, inter-arrival times of user requests are independent, identically, and generally distributed with mean $\frac{1}{\lambda}$ and variance σ_λ^2. The service times of user requests are also independent, identically, and generally distributed with mean $\frac{1}{\mu}$ and variance σ_μ^2. Moreover, inter-arrival times and service times are independent. The system can predict upcoming arrivals ω amount of time ahead and serve them proactively.

By definition, under the FCFS queuing policy, user delay is the summation of queuing delay and service time, where queuing delay is defined as the time from when the user request arrives at the system till the system starts serving it. In G/G/1$^{[\omega]}$, service times follow general distribution. As a result, user delay distribution is also general. As shown by Corollary 1, proactive serving essentially "shifts" the delay distribution left by ω amount of time. Therefore, average user delay will decrease monotonically to zero as ω increases, yet the decrement may not be exponential in ω. Instead, we focus on the effect of proactive serving on reducing user queuing delay (*i.e.*, user delay subtracts service time) and show that proactive serving decreases the average user queuing delay exponentially as ω increases.

Let QD^ω denote the user queuing delay when the system predicts ω amount of time ahead and QD^0 denote the user queuing delay without proactive serving. The explicit form of the distribution of QD^0 in G/G/1 is still an open problem. However, the authors of [26] and [27] have derived an useful upper bound on the tail of the distribution of QD^0. Let random variable X be the inter-arrival time and Y be the service time. Let $U(s)$ denote the Laplace-Stieltjes transform of random variable $Y-X$. Define

$$s_0 = \sup\{s>0 : U(-s) \le 1\}. \tag{4}$$

The following lemma presents a sufficient condition for such s_0 to exist.

Lemma 3 [26]: s_0 defined in (4) exists if the following condition is satisfied:

$$\exists \bar{s} \in (-\infty, \infty), \quad \text{s.t. } E\left[e^{\bar{s}Y}\right] < \infty. \tag{5}$$

The condition in (5) requires that the convergence region of the moment generating function of Y includes an interval around the origin. Clearly, whether condition in (5) is satisfied depends on the distribution of Y, and it is satisfied for many popular distributions for modeling service time, including Exponential, Gamma, and Weibull [28], [29].

Assuming that the condition in (5) is satisfied and s_0 defined in (4) exists, the tail of the distribution of QD^0, *i.e.*, $\Pr(QD^0 \ge t)$, can be upper-bounded as follows [26], [27]

$$\Pr(QD^0 \ge t) \le e^{-s_0 t}, \quad \forall t > 0. \tag{6}$$

Combining the distribution shift effect under proactive serving, this upper-bound in (6) enables us to prove that the average user queuing delay, *i.e.*, $E[QD^\omega]$, decreases exponentially under proactive serving.

Theorem 2: Assume $\mu > \lambda$ and there exists a real $\bar{s}$ such that $E[e^{\bar{s}Y}] < \infty$. The average queuing delay of G/G/1$^{[\omega]}$ with perfect prediction is given by

$$E[QD^\omega] \le \frac{1}{s_0} e^{-s_0\omega}, \tag{7}$$

where s_0 exists and is defined in (4).

Proof: Let $f_{QD^\omega}(t)$ denote the distribution of QD^ω. Let QD^0 denote the queuing delay and $f_{QD^0}(t)$ denote the corresponding distribution without proactive serving. First, we show that the distribution of QD^ω can be obtained by shifting that of QD^0 by ω unit.

Although Corollary 1 is established for the distribution of user delay, it can be easily extended to the distribution of queuing delay based on the same proof. We can have

$$f_{QD^\omega}(t) = f_{QD^0}(t+\omega) \text{ for } t > 0$$

and

$$\Pr(QD^\omega = 0) = \int_0^\omega f_{QD^0}(t)dt.$$

Now we are ready to show that the average queuing delay decreases exponentially in ω. From [26], we have the following about the tail of the distribution of QD^0

$$\Pr(QD^0 \ge t) \le e^{-s_0 t}. \tag{8}$$

Then we have

$$\begin{aligned} E[QD^\omega] &= \int_0^\infty \Pr(QD^\omega \ge t)dt = \int_0^\infty \Pr(QD^0 \ge t+\omega)dt \\ &\le \int_0^\infty e^{-s_0(t+\omega)}dt = \frac{1}{s_0}e^{-s_0\omega}. \end{aligned}$$

■

Theorem 2 says that, as long as the service time distribution satisfies condition (5), the average user delay decreases exponentially under proactive serving. In general, computing the explicit form of s_0 is difficult. In Lemma 4 below, for the heavy-load regime ($\rho = \frac{\lambda}{\mu} \approx 1$ but remains strictly less than 1), one can leverage the results in [30] and [31] to show that higher-order terms of Taylor expansion (around $s=0$) of $U(s)$ can be neglected and an approximate expression of s_0 can be obtained. Since the arguments and details are similar to those in [30] and [31] (for analyzing queuing system

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

performance without proactive serving), we skip the details here and refer interested readers to [30] and [31].

Lemma 4: When $\rho = \frac{\lambda}{\mu} \approx 1$ but remains strictly less than 1, s_0 as defined in (4) can be approximated by

$$s_0 \approx 2\left(\frac{1}{\lambda} - \frac{1}{\mu}\right) / \left(\sigma_\lambda^2 + \sigma_\mu^2\right). \tag{9}$$

From (9), we can see how the inter-arrival time and the service time distributions affect the average queuing delay under proactive serving. First, we can see that s_0 is a decreasing function of λ and an increasing function of μ. This is intuitive. The incoming workload increases when λ increases or μ decreases. Consequently the server is more dedicated to serve requests in the queue and thus the number of pre-served requests decreases. Then the delay reduction by proactive serving is less significant. From (9), we can also see that s_0 is a decreasing function of σ_λ^2 or σ_μ^2. When the variance of interval-time or service time increases, the incoming workload becomes more bursty. In this case, the result in (9) suggests that proactive serving is less effective in reducing user delay.

V. Proactive Serving With Imperfect Prediction

In Section IV, we analyze the benefit of proactive serving under perfect arrival prediction. In this section, we look at two more realistic scenarios that correspond to two common types of prediction errors, and we study the performance of proactive serving under these settings. For the ease of analysis and illustration, we consider a single server setting.

A. Modeling

The first type of error is failing to predict actual arrivals, *i.e.*, miss detection (also called false negative). When miss detection happens, the missed arrivals cannot be proactively served. Intuitively, such errors result in a "side flow" into the system and will affect the benefit of proactive serving. The other type of error is false alarm (also called false positive), which happens when the system mistakenly predicts non-existing arrivals. Such false arrivals will not eventually enter the system for service. However, the system may incorrectly allocate resources to pre-serve them, resulting in wasted service capacity.

We represent the system with these two types of prediction errors using the model shown in Fig. 6. In this model, $\{A_1(t)\}_t$ represents the process of miss detections and $\{A_2(t)\}_t$ represents the process of predicted arrivals, which include false alarms and actual arrivals that are predicted correctly. $Q_0(t)$ stores requests that have already entered the system and are waiting for service at time t. $W_\omega(t)$ is the prediction window with size ω. Requests in $\{A_2(t)\}_t$ go through the prediction window and can be served proactively by the server. In contrast, requests in $\{A_1(t)\}_t$ enter $Q_0(t)$ directly and cannot be served proactively. False alarms in $\{A_2(t)\}_t$ disappear once they leave the prediction window and do not enter $Q_0(t)$.

For tractability, we make the following two assumptions on $\{A_1(t)\}_t$ and $\{A_2(t)\}_t$. First, we assume that the probability that a request in $\{A_2(t)\}_t$ is an actual arrival is p. With probability p, a request of $\{A_2(t)\}_t$ will enter $Q_0(t)$; with probability $1-p$, it will disappear once it leaves the prediction window. The larger p is, the more accurate the prediction is. Second, we assume that $\{A_1(t)\}_t$ and $\{A_2(t)\}_t$ are independent Poisson processes, which is reasonable for systems with large number of users.

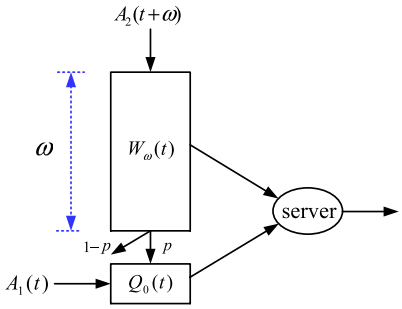


Fig. 6. Service system with imperfect prediction: The miss detection process $\{A_1(t)\}_t$ can not be served proactively. Requests in $\{A_1(t)\}_t$ enter $Q_0(t)$ for service directly. The process $\{A_2(t)\}_t$ includes false alarms and actual arrivals that are predicted correctly. Requests in $\{A_2(t)\}_t$ go through $W_\omega(t)$ and can be pre-served by the server. p is the probability that a request in $\{A_2(t)\}_t$ is an actual arrival.

Let λ_1 and λ_2 be the arrival rates of $\{A_1(t)\}_t$ and $\{A_2(t)\}_t$ respectively. Since all the miss detections and the actual arrivals among the predicted arrivals compose the actual arrivals, we have

$$\lambda_1 + p\lambda_2 = \lambda. \tag{10}$$

Recall that λ is the rate for the actual arrival process.

In this system, the server applies the FCFS service policy. Different from the case of perfect prediction, here we assume that the requests in $Q_0(t)$ and the arrivals from $\{A_1(t)\}_t$ have preemptive priority. That is, the arrivals in $W_\omega(t)$ will be pre-served only when the queue is empty and there is no new arrival entering $Q_0(t)$ from $\{A_1(t)\}_t$.

B. Impact of Miss Detection

Now suppose that there are only miss detections in the system, *i.e.*, $p = 1$ and $\lambda_1 + \lambda_2 = \lambda$. In this case, the delay reduction can be less significant as compared to the perfect-prediction case, because miss detection cannot be pre-served by the system.

To characterize the impact of miss detections, we follow the same idea used in the perfect prediction case. That is, linking the distribution of D^ω to that of a single queue system without proactive serving. After obtaining the distribution of D^ω, we then calculate the average user delay, *i.e.*, $E[D^\omega]$. Miss detection creates a sub-arrival process into $Q_0(t)$ and mixes with the requests in $\{A_2(t)\}_t$. This makes it challenging to derive the user delay distribution. In the derivation process, we need to apply residue theorem [32] with carefully designed branch cuts [33] to inverse Laplace transform to obtain the average user delay. We arrive at the following result.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

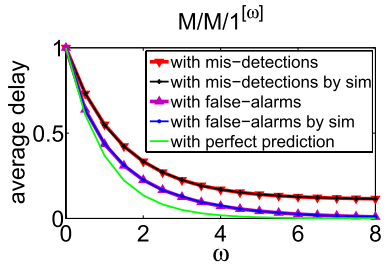


Fig. 7. The average request delay under perfect prediction, miss detection and false alarm.

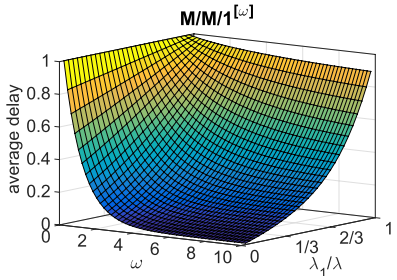


Fig. 8. Impact of miss detection on the average request delay with $\lambda = 3$ and $\mu = 4$. The fraction of miss detection is $\frac{\lambda_1}{\lambda}$.

Theorem 3: Assume $\lambda = \lambda_1 + \lambda_2 < \mu$. The average user delay in the presence of miss detections is given by:

$$E[D^\omega] = \frac{\lambda_1}{(\mu-\lambda_1)\lambda} + \frac{\lambda_2}{\lambda} \times \left[\frac{\lambda^2 - \lambda_1\mu}{\lambda_2^2(\mu-\lambda)} \cdot e^{\frac{\lambda_2(\lambda-\mu)}{\lambda}\omega} \cdot \mathbf{1}_{\lambda^2>\lambda_1\mu} + \frac{1}{2\pi}\int_0^{4\sqrt{\lambda_1\mu}} \times \frac{(\mu-\lambda)\sqrt{x(4\sqrt{\lambda_1\mu}-x)}e^{(-(\sqrt{\mu}-\sqrt{\lambda_1})^2-x)\omega}}{\left((\sqrt{\mu}-\sqrt{\lambda_1})^2+x\right)^2 \cdot \left(\lambda x + (\sqrt{\lambda_1\mu}-\lambda)^2\right)} dx \right], \quad (11)$$

which decreases exponentially in ω.

Proof: The proof is relegated to Appendix C. ∎

The average user delay expression in (11) consists of two terms. The first term is a positive constant representing the delay experienced by the miss-detection user requests. As shown in Fig. 6, miss detections enter $Q_0(t)$ directly without going through the prediction window. As such, these miss detections cannot be served proactively and will always experience positive delay. The second term in (11) decreases exponentially in ω and vanishes as the system predicts sufficiently far into the future, demonstrating the effectiveness of proactive serving.

We validate the closed-form expression of average user delay in Theorem 3 by plotting the average user delay by (11) (the red curve marked by down triangle) and by simulations (the black curve marked by '+') in Fig. 7 under the setting of $\lambda_1 = 1$, $\lambda = 3$, and $\mu = 4$. We observe that, (i) two curves coincide, which verifies Theorem 3, and (ii) user delay decreases exponentially as the system predicts further. The user delay is eventually dominated by the first term in (11) as expected. Compared to the scenario with perfect prediction, we note that the decay rate in ω is smaller. This is intuitive because miss detections occupy part of the server capacity and thus the capacity used for proactive serving is reduced. We plot the average user delay as a function of both miss detection fraction and prediction window size ω in Fig. 8, based on (11). The fraction of miss detection is $\frac{\lambda_1}{\lambda}$. As seen, proactive serving is more effective when there is few miss detection (*i.e.*, high prediction accuracy). This matches our intuition since more miss detections mean that more user requests go directly into the system without having the chance to be served proactively. The plot also suggests that when there are substantial miss detections, predicting further into the future is not effective in reducing user delay, and we are better off if we focus on improving the prediction accuracy (*i.e.*, reducing the fraction of miss detection).

C. Impact of False Alarm

When there are only false alarms in the system, *i.e.*, $\lambda_1 = 0$ and $p\lambda_2 = \lambda$, the server capacity will be wasted if a false alarm is pre-served. As a result, the benefit of proactive serving will be less significant as compared to the perfect prediction case.

Different from miss detections, false alarms do not enter $Q_0(t)$. Therefore, the system cannot be modeled by a single queue system without proactive serving. As a result, we cannot carry out the same equivalence argument as for the miss detection case in Section V-B. Hence, the idea used in the miss detection scenario can not be applied here. To address the difficulty, we first discretize the system and model its evolution by a multi-dimensional Markov chain where user requests passing through the window is captured by moving from one state to its subsequent state in the Markov chain. This Markov chain is non time-reversible. Then, by studying the stationary distribution of the Markov chain, we obtain the average user delay by applying Little's Law. We arrive at the following result.

Theorem 4: Assume $\lambda = p\lambda_2 < \mu$ $(0 < p \leq 1)$. The average user delay in the presence of false alarms is given by

$$E[D^\omega] = \begin{cases} \frac{p\mu-\lambda}{p(\mu-\lambda)^2}\left(e^{\frac{1}{p}(p\mu-\lambda)\omega} - \frac{(1-p)\lambda}{p(\mu-\lambda)}\right)^{-1}, & \text{when } \lambda < p\mu; \\ \frac{\lambda-p\mu}{(1-p)\lambda(\mu-\lambda)} + \frac{p(\lambda-p\mu)}{(1-p)^2\lambda^2}\left(e^{\frac{1}{p}(\lambda-p\mu)\omega} - \frac{p(\mu-\lambda)}{(1-p)\lambda}\right)^{-1}, & \text{when } p\mu < \lambda \text{ and } \lambda < \mu; \\ \frac{1}{(\mu-\lambda)\left[(\mu-\lambda)\omega+1\right]}, & \text{when } p\mu = \lambda. \end{cases} \quad (12)$$

Due to the space limitation, the proof is included in [24].

From (12), it is not immediately clear that $E[D^\omega]$ decreases with an exponential rate when $\lambda_2 \neq \mu$. Instead, we show in the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

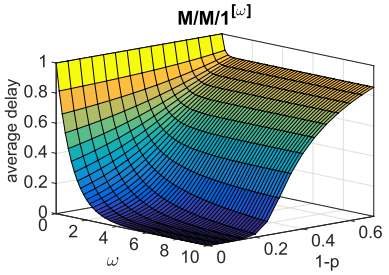


Fig. 9. Impact of false alarm on the average request delay with $\lambda = 3$ and $\mu = 4$. The fraction of false alarm is $1 - p$.

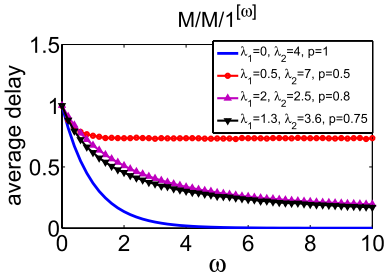


Fig. 10. The average request delay vs. how far we predict into the future under different prediction strategies.

proof in [24] that $E[D^\omega]$ can be lower and upper bounded by exponential functions which decrease exponentially in ω. The average delay decreases exponentially to zero when $\lambda < p\mu$ and decreases exponentially to a constant value when $\lambda > p\mu$. We call $p\mu$ "effective service rate". Theorem 3 says that the average user delay decreases exponentially to zero in the prediction window size, if the user request arrival rate is smaller than the effective service rate, *i.e.*, $\lambda < p\mu$, and otherwise to a positive constant determined by the fraction of false alarms.

We plot the average user delay under the impact of false alarms by (12) (the pink curve marked by up triangle) and by simulations (the blue curve marked by dot) in Fig. 7 under the setting of $\lambda_2 = 3.5, \lambda = 3$, and $\mu = 4$. The results clearly verify Theorem 3 and show that the average delay decreases exponentially in ω. As compared to the perfect prediction case, the decay rate is smaller. This is because part of the server capacity is wasted to serve false alarms. Moreover, we can derive the following from (12):

$$\lim_{\omega\to\infty} E[D^\omega] = \begin{cases} 0, & \text{when } \lambda < p\mu, \dfrac{\lambda}{\mu} < p \le 1; \\ \dfrac{\mu}{(\mu-\lambda)\lambda} - \dfrac{1}{(1-p)\lambda}, & \text{when } p\mu \le \lambda < \mu, 0 < p \le \dfrac{\lambda}{\mu}. \end{cases} \tag{13}$$

This equation shows that the system cannot reduce delay to zero if $p\mu < \lambda < \mu$. This is because statistically $1 - p$ fraction of the service capacity allocated for proactive serving is consumed by false alarms, and the remaining p fraction is not enough to pre-serve all the actual arrivals before they enter the system. As such, the user delay can not be reduced to zero no matter how far we can predict into the future. Meanwhile, when $\lambda \le p\mu$ the delay can always be reduced to zero as long as the system can predict sufficiently far.

In Fig. 9, we show the impact of false alarms on user delay based on (12). The fraction of false alarm in the predicted arrivals is $1 - p$. As seen from the figure, the average user delay increases when more false alarms exist in the system. When $\lambda > p\mu$ (corresponds to $1-p > 0.25$ in the simulation), the average delay does not decrease to zero. The larger $1 - p$ is, the less helpful proactive serving is in reducing user delay. In practice, system designers should improve their prediction algorithms to keep the fraction of false alarms less than the threshold $1-\frac{\lambda}{\mu}$ to extract the full potential of proactive serving. If the fraction of false alarms is inevitably larger than the threshold, then capacity boosting maybe a better choice for system designers to decrease the average delay.

Discussion: In addition to the impact on user delay reduction, false alarm may incur additional costs. For example, in the content pre-fetching scenario, serving false alarms will waste bandwidth on the system side, and bandwidth, storage, and energy on the user side. Thus system designers should take into account the consequences of false alarms when designing the request prediction algorithm.

D. Impact of Miss Detection and False Alarm

When miss detections and false alarms are both present in the system, it is difficult to analyze the user delay due to the coupling of the two effects. Instead, we conduct simulations to investigate the system behavior. We consider three cases. In the first case, the prediction mechanism results in few miss detections but many false alarms. In the second case, the prediction mechanism leads to few false alarms but many miss detections. The third case is in-between. The first two can be considered as extreme cases. The simulation results under the setting of $\mu = 5$ are shown in Fig. 10.

A few comments are in order for Fig. 10. First of all, user delay still decreases exponentially as the system predicts further. Second, when there are many false alarms, the system cannot remove delay completely, which aligns with (13). When there are many miss detections, the delay decay rate is smaller because the server is occupied with miss detections, which also aligns with the results in Fig. 8. When the number of miss detections and false alarms are moderate, compared to other two cases, user delay decreases rapidly and proactive serving leads to a small delay for users.

VI. Comparison With Capacity Boosting

In this section, we compare capacity boosting and proactive serving with perfect prediction as two principal design mechanisms for reducing user delays. In the case of imperfect prediction, the closed-form expression does not admit a clear comparison, and thus we resort to perform the comparison based on simulation results in Section VII.

For easy discussion, we focus on the M/M/1$^{[\omega]}$ system as discussed in Section IV-A. For the M/M/1 system without

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

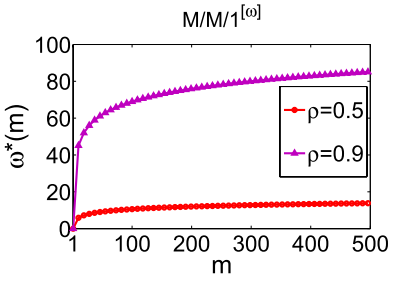


Fig. 11. How far do we need predict in order to achieve the same delay performance as compared to the server capacity being increased by m times?

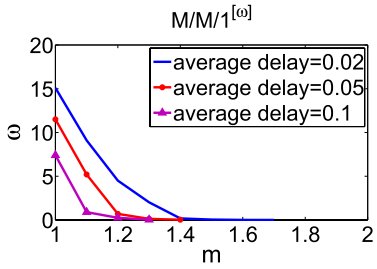


Fig. 12. To achieve the same delay performance, the system can select different combinations of proactive serving and system capacity boosting.

proactive serving capability, the average user delay with service capacity m is given by $1/(m \cdot \mu - \lambda)$. By comparing it with the average delay of the M/M/1$^{[\omega]}$ system with proactive serving capability in (3), we obtain the amount of prediction (measured in prediction window size) needed to obtain the same delay reduction as boosting the capacity by m times. Denote the corresponding prediction window size as $\omega^*(m)$, we have the following theorem.

Theorem 5: Assume $\lambda < \mu$. For the M/M/1$^{[\omega]}$ system with perfect prediction, $\omega^*(m)$ $(m \geq 1)$ is given by

$$\omega^*(m) = \frac{1}{\mu - \lambda} \ln \frac{m - \rho}{1 - \rho}.$$

Proof: For the M/M/1 system without proactive serving, the average user delay is $\frac{1}{m\mu - \lambda}$ when the service capacity is m. To achieve the same delay performance, $\omega^*(m)$ should satisfy

$$E[D^{\omega^*(m)}] = \frac{1}{\mu - \lambda} e^{-(\mu - \lambda)\omega^*(m)}.$$

Then we obtain $\omega^*(m) = \frac{1}{\mu - \lambda} \ln \frac{m - \rho}{1 - \rho}$ where $\rho = \frac{\lambda}{\mu}$. ■

We plot the $\omega^*(m)$ as a function of m in Fig. 11 under different ρ with $\mu = 1$ (recall that $\rho = \frac{\lambda}{\mu}$). We observe that $\omega^*(m)$ increases logarithmically in the system capacity m. This suggests that proactive serving is more effective than boosting the system capacity to keep the system in low delay regime. For example, to achieve the same average user delay, proactive serving with prediction window size of **14** unit time is equivalent to increasing the server capacity by **500** times when $\rho = 0.5$.

In Fig. 11, when $\rho = 0.9$, the system is in heavy-load regime when m is small and is in light-load regime when m is large. Then the logarithmic curve indicates that proactive serving is more effective in delay reduction than capacity boosting in light-load regime. The reason is as follows. When the workload is light, the number of requests that enter the queue for service is small and thus most of the service capacity can be spared to serve future requests. As a result, most of requests are served proactively and thus experience low delay.

It is also conceivable to combine proactive serving and capacity boosting to achieve a desired average user delay for a system. To evaluate this idea, we run simulations under the setting of $\lambda_1 = 0.4$, $\lambda_2 = 8$, $\mu = 10$ and $p = 0.95$ (for modeling imperfect prediction), as defined in Fig. 6. We plot different combinations of ω (representing proactive serving capability) and m (representing service capacity) to achieve the same desired delay target in Fig. 12. All the combinations of ω and m on the same isoline achieve the same average user delay. For example, to obtain the average user delay of 0.1 unit time, the system can serve proactively 7.4 unit time ahead without boosting the service capacity, or it can serve proactively 0.9 unit time ahead and boost the service capacity by 10%. The system designer can select the best combination according to the average user delay requirement and various resource constraints.

In practice, the choice of strategy to improve system delay performance may involve additional considerations. For capacity boosting, the incurred operation cost is a non-negligible factor. For proactive serving, the cost of bandwidth, storage, and energy due to false alarms are also needed to be considered. Along this line, we have obtained initial results by adapting the well-studied network utility maximization framework to achieve an optimized performance trade-off among delay reduction and multiple design considerations. Due to the space limitation, we skip the details and refer interested readers to our technical report [24].

VII. Simulations

We carry out simulations to study the impact of proactive serving on reducing the average user delay under different practical request arrivals and settings. Our objective is to evaluate: (i) How is the delay performance of the system with proactive serving capability? (ii) How does prediction error affect the delay reduction? (iii) How does proactive serving compare to capacity boosting?

A. Parameters and Settings

To carry out our simulations, we collect two sets of videos from YouTube [35]. One set of 557 videos are popular ones, and the other set of 443 videos are randomly chosen videos.

By studying the histogram of durations of the popular videos, we observe that the histogram fits light-tailed Gaussian distribution well, as shown in Fig. 13. This observation also confirms the measurement result in [36] that the video length of popular YouTube categories follows Gaussian distribution. For the randomly-chosen videos, we observe that the empirical CCDF (Complementary Cumulative Distribution Function) of their durations fits heavy-tailed power law distribution well, as shown in Fig. 14. Note that it is common to study CCDF to evaluate whether a distribution is heavy-tailed or not.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

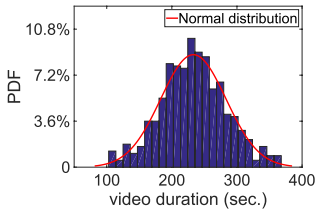


Fig. 13. The histogram fits well with a Gaussian distribution with R-squared value [34] of 92%, suggesting that the distribution of video duration is light-tailed.

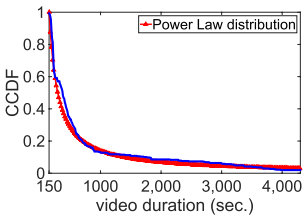


Fig. 14. The ccdf appears to fit well a Power Law distribution with R-squared value 97%, indicating that the distribution of video duration is heavy-tailed.

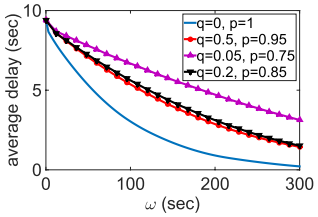


Fig. 15. The average request delay vs. how far we predict into the future under the light-tailed video set.

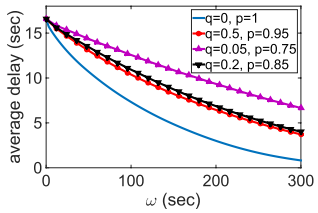


Fig. 16. The average request delay vs. how far we predict into the future under the heavy-tailed video set.

We evaluate the delay performance of popular YouTube videos, representing the light-tailed service time case, and that of randomly-chosen YouTube videos, representing the heavy-tailed serving time case.

In the simulations of popular Youtube videos, we set the number of servers to be 650 in the system. In the simulation of randomly-chosen videos, we set the number of servers to be 270 in the system. In one second, a single server can serve a video of 700 seconds, which is about 100MB in size. Such a setting is chosen for two reasons. First, under such setting, the workload level for two data sets is same so that we can make comparisons between simulation results under the two sets. Second, it is to make sure that the average user delay is reasonable. The system adopts the FCFS service policy.

To investigate the impact of prediction errors, we model the miss detections and false alarms in the simulation as follows. Each time unit, q fraction of the request arrivals are miss detections. The rest $1-q$ fraction compose actual arrivals in the predicted request arrivals that can be served proactively. In the predicted arrivals, $1-p$ fraction are false alarms and p fraction are actual arrivals. Larger q means more miss detections, and smaller p means more false alarms in the system. For perfect prediction, $q = 0$ and $p = 1$.

B. Delay Reduction by Proactive Serving

The simulation results under the popular videos (with light-tailed durations) and the randomly-selected (with heavy-tailed durations) are shown in Figs. 15 and 16, respectively. For the popular videos, under perfect prediction, the user delay can be reduced by 50% when the system predicts 60 seconds ahead. For the randomly-chosen videos, under prefect prediction, the user delay can be reduced by 50% when the system predicts 84 seconds ahead. The above simulation results suggest that the system can improve user delay experience significantly by proactive serving. Furthermore, for the popular videos with light-tailed durations, the delay curve under perfect prediction can be fitted nicely by the exponential function $9.91e^{-0.012\times\omega}$ with R-squared value of 99.74%. This verifies the theoretical results we obtain in Section IV-B.

In Fig. 16, we observe similar delay reduction as in Fig. 15. That is, when ω is small, increasing ω can lead to larger delay reduction than the case when ω is large. On the other hand, the delay reduction rate in Fig. 16 is smaller than that in Fig. 15. Request for elephant videos plays an important role on the reduction rate. In the randomly-chosen videos of heavy-tailed durations, elephant video is not rare. Request for such video will occupy the service for a long time. Therefore, the delay reduction under the heavy-tailed data set is less significant. It remains an interesting direction to characterize the delay performance under proactive serving for the case of heavy-tailed duration.

C. Impact of Miss Detection and False Alarm

The simulation results are shown in Fig. 15 and 16. As seen, both miss detection and false alarm leads to smaller delay reduction. But proactive serving can still reduce user delay significantly. For example, in Fig. 15, the user delay can be reduced by 50% as the system predicts about 123 seconds even when the prediction misses 50% of the actual arrivals (the red curve with circle maker).

Under all the data traces, proactive serving is relatively more sensitive to false alarms than miss detections, which matches what we observed in Section V-D. The reason is that some server capacity allocated to future requests is wasted due to false alarm, thus server capacity is not well utilized.

D. Comparison With Capacity Boosting

We compare proactive serving under perfect prediction with system capacity boosting using all 1000 Youtube video traces. The results are shown in Tab. I. In the table, the first column is how far in time the system serves proactively with

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
COMPARISON WITH SYSTEM CAPACITY BOOSTING UNDER THE YOUTUBE DATA TRACE

ω^*(sec.)	# of servers $\uparrow$	utilization $\downarrow$	average delay $\downarrow$
120	10%	9.1%	54.1%
219	20%	16.7%	75.3%
300	30%	23.1%	89.9%

perfect prediction. The second column is by how many percents the number of servers needs to increase to achieve the same delay performance. The third column is percentage of the decrement of the utilization ratio of each server, when the total number of servers increases. The fourth column is how many percents the average user delay is reduced. For example, in Tab. I, the first row says, serving proactively 120 seconds ahead can reduce the average user delay by 54.1%. To achieve the same delay performance, the system needs to increase the number of servers by 10%, which results in that the utilization rate of each server is decreased by 9.1%.

From Tab. I, we first observe that, when the system capacity increases, the rate of ω^* increasing slows down, where ω^* is how far the system needs to predict to achieve the same delay performance as capacity boosting. This agrees with what we observed in Fig. 11 that proactive serving is more effective than capacity boosting in the light-load regime. For example, to achieve the same delay reduction attained by boosting the capacity by 10%, one needs to predict request arrivals 120 seconds ahead. Meanwhile, suppose we already boost the capacity by 20%, to achieve the same delay reduction by boosting another 10% of the capacity (assuming incurring the same amount of expense), one only needs to increase the prediction window size by $300 - 219 = 81$ seconds, which is smaller than the 120-second increment required in the previous (heavier load) case. Results from Tab. I also show that, under the Youtube data trace, boosting the capacity by 30% is equivalent to predicting 300 seconds ahead; both reduce the average user delay by about 90%. Since predicting 300 seconds ahead with decent accuracy is very feasible according to the study on predicting user requests for an industrial-grade VoD system [6], this means one can achieve the same delay reduction by leveraging prediction, without significant investment to increase the service capacity by 30%.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we investigate the fundamentals of proactive serving from a queuing theory perspective. We show that proactive serving decreases average delay exponentially (as a function of the prediction window size) with perfect prediction. We show that in the presence of miss detections, average user delay still decreases exponentially in the prediction window size, but to a positive constant determined by the fraction of miss detections. Meanwhile, in the presence of false alarms, we show that there exists an "effective service rate" $p\mu$. Average user delay decreases exponentially to zero as prediction window size increases if the actual user request arrival rate is smaller than the effective service rate; otherwise it decreases to a positive constant determined by the fraction of false alarms. Compared with the conventional mechanism of capability boosting, we show that proactive serving is more effective in decreasing user delay in a light workload regime. Our trace driven evaluation results demonstrate the practical power of proactive serving, *e.g.*, for the data trace of light-tailed Youtube videos, user delay decreases by 50% when the system can predict 60 seconds ahead. Our study applies to general online service models that use prediction, such as the pre-loading interested content on mobile devices and the pre-fetching Youtube videos. Therefore, our results not only provide solid theoretical foundations for proactive serving, but also reveal insights into its practical applications.

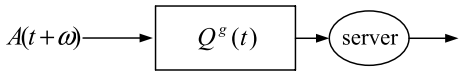


Fig. 17. G/G/1: The arrival process is $\{A(t+\omega)\}_t$. Service times of requests are independent and identically distributed with mean $1/\mu$. The initial value $Q^g(0)$ is $|A(0:\omega)| + Q_0(0)$. The service policy is FCFS.

Our research offers several interesting future directions. First, while in the paper we focus on characterizing the benefit of proactive serving, it would also be valuable to consider the costs/overhead involved in proactive serving, *e.g.*, bandwidth cost due to false alarms, to properly evaluate its effectiveness. Second, while this paper only considers proactively serving one predictive request at a time, it would be conceivable and interesting to generalize the analysis to consider serving multiple predictive requests simultaneously. Third, while we consider time-homogeneous prediction accuracy in this paper, it would be interesting to generalize the study to cases with time-heterogeneous prediction accuracy where short-time prediction is more accurate than long-term prediction. Last but not the least, it would be interesting to analyze the delay reduction benefit of proactive serving under work-conserving policies other than the FCFS used in our current study.

APPENDIX A
PROOF OF COROLLARY 1

Consider a general queuing system as shown in Fig. 3. We can always find a queuing system without proactive serving capability as in Fig. 17, where the arrival process is delayed ω amount of time, the initial value of queue is equal to $|A(0:\omega)| + Q_0(0)$ and the service policy is the same as adopted in the general queuing system. As such, we can get the same equivalence between two systems as Lemma 1. Based on how $f^G_{D^0}(t)$ is derived in [37], the delay distribution of the system in Fig. 17 is the same as $f^G_{D^0}(t)$. This is because that we consider the delay distribution when the system is in steady state. Then the initial value of Q^g (which is bounded) and the shifted arrival process will not alter the delay distribution. Then, based on the same argument as in Lemma 2, we can show that the delay distribution under proactive serving can be obtained by shifting the delay distribution without proactive serving left by ω amount of time.

APPENDIX B

Discussion: An alternative approach to obtain $E[D^\omega]$ in Theorem 1 is applying the Markov chain model, which works

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

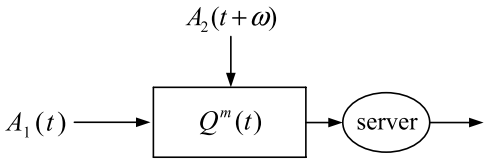


Fig. 18. M/M/1 with preemptive priority: Two arrival processes are $\{A_1(t)\}_t$ and $\{A_2(t+\omega)\}_t$ respectively. Service times of requests are independent and identically exponentially distributed with mean $1/\mu$. The initial value of Q^m is $|A_2(0:\omega)|+Q_0(0)$. Requests in $\{A_1(t)\}_t$ have preemptive priority over those in $\{A_2(t+\omega)\}_t$.

as follows. First, we discretize the system. We chop the time into slots of equal length. Let δ denote the slot length. At time slot t, $A(t)$ becomes a Bernoulli random variable with probability $\lambda\cdot\delta$. Each time slot, the server is on with probability $\mu\cdot\delta$ and off with probability $1-\mu\cdot\delta$. When the server is on, it can serve a request in one slot. $Q_0(t)$ stores requests that are waiting in the system for service at slot t. The prediction window $W_\omega(t)$ is chopped to ω/δ small windows which are denoted by $\{w_i(t)\}_{1\le i\le\omega/\delta}$. Each request first goes through a pipeline of these small windows from $w_{\omega/\delta}(t)$ to $w_1(t)$ before entering $Q_0(t)$. If $A(t+i\delta)=1$, then the system can observe a request in the window $w_i(t)$, which can be served proactively. Then, based on the efforts in the above step, the system can be modeled by a multi-dimensional Markov chain with state being $\big(w_{\omega/\delta}(t), w_{\omega/\delta-1}(t), ..., w_1(t), Q_0(t)\big)$. By solving the stationary distribution of the Markov chain, we can obtain the average user delay of the discretized system by applying Little's Law. Then, by taking limit in δ, we finally get $E[D^\omega]$. Remark that the structure of the Markov chain is very complicated which makes the derivation of the stationary distribution highly involved.

Compared the approach we used in the paper, the above approach is complicated and cannot provide intuitive insights. At the same time, the approach is hard to be generalized to more complex models. For example, for the system in Section V, the Markov chain that models the discretized system is much more complex, which is rather challenging to solve.

APPENDIX C
PROOF OF THEOREM 3

Instead of FCFS, the system under imperfect prediction gives preemptive priority to the requests of $\{A_1(t)\}_t$, and within the same arrival process FCFS is adopted.

Under this policy, $Q^{sum}(t)=Q_0(t)+W_\omega(t)$ evolves the same as the system in Fig. 18. The proof is similar to that of Lemma 1 and thus omitted. Consider a request in $\{A_2(t)\}_t$. Similar to Lemma 2, if it spends T time in Q^m, it will spend $[T-\omega]^+$ in Q_0. Then the distribution of delay of requests in $\{A_2(t)\}_t$ spend in Q_0 can be obtained by shifting that in Q^m left by ω units. Let $f_2^\omega(t)$ and $f_2^m(t)$ be the density function of delay that requests of $\{A_2(t)\}_t$ spend in Q_0 and Q^m, respectively. We have $f_2^\omega(t)=f_2^m(t+\omega)$ when $t>0$.

Next we first calculate $f_2^m(t)$, and then we can obtain $f_2^\omega(t)$. To do so, as the first step, we express $f_2^m(t)$ as a function of distributions of busy period of the M/M/1 system. By leveraging existing knowledge on busy period of the M/M/1 system (in particular, we know the Laplace transform of the distribution of length of the busy period), we get the Laplace transform of $f_2^m(t)$. Then, based on the relationship between $f_2^m(t)$ and $f_2^\omega(t)$, we get the Laplace transform of $f_2^\omega(t)$.

Now we focus on the system in Fig. 18. It can be modeled by a Markov chain and let π_i be the stationary probability that the number of requests in the system is i. By standard analysis, we can have $\pi_i=(1-\frac{\lambda_1+\lambda_2}{\mu})(\frac{\lambda_1+\lambda_2}{\mu})^i$. Consider a request of $\{A_2(t)\}_t$. Let us denote the request by a_2. Let T be the time that a_2 spends in the system. Let N be the total number of requests already in the system when the request enters the system. Let T_{i+1} be the time that a_2 spends in the system conditioning on $N=i$. Then we have

$$P(T\le t)=\sum_{i=0}^{\infty}\pi_i P(T\le t|N=i)=\sum_{i=0}^{\infty}\pi_i P(T_{i+1}\le t). \tag{14}$$

T_{i+1} is equal to the time till a standard M/M/1 system is empty again when there are $i+1$ requests in the M/M/1 system [38]. Note that T_1 is the length of a busy period. Denote the probability density functions of T_{i+1} by $f_{T_{i+1}}(t)$. By (14), we get $f_2^m(t)=\sum_{i=0}^{\infty}\pi_i f_{T_{i+1}}(t)$. From [38], we know that the Laplace transform of $f_{T_{i+1}}(t)$ is

$$F_{i+1}(s)=\left[\frac{1}{2\lambda_1}\left(\lambda_1+\mu+s-\sqrt{(\lambda_1+\mu+s)^2-4\lambda_1\mu}\right)\right]^{i+1}.$$

So the Laplace transform of $f_2^m(t)$ is

$$\begin{aligned}F(s)&=\sum_{i=0}^{\infty}\pi_i F_{i+1}(s)\\&=\frac{2(\mu-\lambda_1-\lambda_2)}{\mu-\lambda_1-2\lambda_2+s+\sqrt{(\lambda_1+\mu+s)^2-4\lambda_1\mu}}.\end{aligned}$$

By definition, the Laplace transform of $f_2^\omega(t)$ is

$$\begin{aligned}F^\omega(s)&=\int_0^\infty e^{-st}f_2^m(t+\omega)dt+\int_0^\omega f_2^m(t)dt\\&=e^{s\omega}\int_\omega^\infty e^{-st}f_2^m(t)dt+\int_0^\omega f_2^m(t)dt\\&=e^{s\omega}\left[F(s)-\int_0^\omega e^{-st}f_2^m(t)dt\right]+\int_0^\omega f_2^m(t)dt.\end{aligned}$$

Based on the Laplace transform, we are ready to calculate the average request delay of the requests of $\{A_2(t)\}_t$ in Q_0, which is denoted by $E[D_2^\omega]$. By the definition of the Laplace transform,

$$\begin{aligned}E[D_2^\omega]&=-\frac{dF^\omega(s)}{ds}|_{s=0}\\&=\frac{\mu}{(\mu-\lambda_1)(\mu-\lambda_1-\lambda_2)}-\omega+\omega\int_0^\omega f_2^m(t)dt\\&\quad-\int_0^\omega t\cdot f_2^m(t)dt.\end{aligned}$$

Then the derivative of $E[D_2^\omega]$ is

$$\frac{dE[D_2^\omega]}{d\omega}=\int_0^\omega f_2^m(t)dt-1.$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

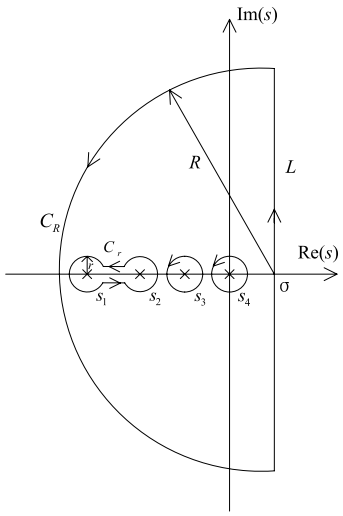


Fig. 19. Contour integration: The contour consists of L and C_R. s_1 and s_2 are branch points. s_3 and s_4 are simple poles.

Now consider $\int_0^\omega f_2^m(t)dt$ as a function of ω. The Laplace transform of it is equal to

$$\frac{F(s)}{s} = \frac{2(\mu-\lambda_1-\lambda_2)}{s\left(\mu-\lambda_1-2\lambda_2+s+\sqrt{(\lambda_1+\mu+s)^2-4\lambda_1\mu}\right)},$$

which is due to the integration property of the Laplace transform. Next, we invert $\frac{F(s)}{s}$ and get the expression of $\int_0^\omega f_2^m(t)dt$.

Define $s_1 = -(\sqrt{\mu}+\sqrt{\lambda_1})^2$, $s_2 = -(\sqrt{\mu}-\sqrt{\lambda_1})^2$, $s_3 = \frac{\lambda_2(\lambda_1+\lambda_2-\mu)}{\lambda_1+\lambda_2}$, $s_4 = 0$. $s_1$and s_2 are branch points of $\frac{F(s)}{s}$. s_4 is a simple pole of $\frac{F(s)}{s}$. When $(\lambda_1+\lambda_2)^2 > \lambda_1\mu$, s_3 is also a simple pole of $\frac{F(s)}{s}$. The residue at s_3 Res(s_3) is $\frac{\lambda_1\mu-(\lambda_1+\lambda_2)^2}{\lambda_2(\lambda_1+\lambda_2)}e^{\frac{\lambda_2(\lambda_1+\lambda_2-\mu)}{\lambda_1+\lambda_2}\omega}$. The residue at s_4 Res(s_4) is 1. Consider the closed contour $L+C_R$ shown in Fig. 19.

$$\begin{aligned}\oint \frac{F(s)}{s}e^{s\omega}ds &= \int_L \frac{F(s)}{s}e^{s\omega}ds + \int_{C_R}\frac{F(s)}{s}e^{s\omega}ds\\ &= \int_{\sigma-jR}^{\sigma+jR}\frac{F(s)}{s}e^{s\omega}ds + \int_{C_R}\frac{F(s)}{s}e^{s\omega}ds\\ &= \lim_{r\to0}\int_{C_r}\frac{F(s)}{s}e^{s\omega}ds\\ &\quad + 2\pi j\cdot\text{Res}(s_4) + 2\pi j\cdot\text{Res}(s_3)\\ &\quad \cdot\mathbf{1}_{(\lambda_1+\lambda_2)^2>\lambda_1\mu}, \end{aligned} \tag{15}$$

where the last equality is based on Cauchy's Theorem. When $R\to\infty$, $\frac{1}{2\pi j}\int_{\sigma-jR}^{\sigma+jR}\frac{F(s)}{s}e^{s\omega}ds$ becomes the inverse transform of $\frac{F(s)}{s}$. According to Jordan's lemma, $\int_{C_R}\frac{F(s)}{s}e^{s\omega}ds\to0$ when $R\to\infty$. Thus, to calculate the inverse transform of $\frac{F(s)}{s}$, we only need to calculate $\lim_{r\to0}\int_{C_r}\frac{F(s)}{s}e^{s\omega}ds$.

$$\begin{aligned}&\lim_{r\to0}\int_{C_r}\frac{F(s)}{s}e^{s\omega}ds\\ &= \lim_{r\to0}\int_{-\pi}^{\pi}\frac{F(s_1+re^{j\theta})}{s_1+re^{j\theta}}\cdot e^{(s_1+re^{j\theta})\omega}\cdot jre^{j\theta}d\theta\\ &\quad + \lim_{r\to0}\int_{-\pi}^{\pi}\frac{F(s_2+re^{j\theta})}{s_2+re^{j\theta}}\cdot e^{(s_2+re^{j\theta})\omega}\cdot jre^{j\theta}d\theta\\ &\quad - \int_0^{4\sqrt{\lambda_1\mu}}\\ &\quad \times\frac{2(\mu-\lambda_1-\lambda_2)e^{(s_2-x)\omega}}{(s_2-x)\left(\mu-\lambda_1-2\lambda_2+s_2-x+j\sqrt{x(4\sqrt{\lambda_1\mu}-x)}\right)}dx\\ &\quad + \int_0^{4\sqrt{\lambda_1\mu}}\\ &\quad \times\frac{2(\mu-\lambda_1-\lambda_2)e^{(s_2-x)\omega}}{(s_2-x)\left(\mu-\lambda_1-2\lambda_2+s_2-x-j\sqrt{x(4\sqrt{\lambda_1\mu}-x)}\right)}dx\\ &= \int_0^{4\sqrt{\lambda_1\mu}}\frac{j(\mu-\lambda_1-\lambda_2)\sqrt{x(4\sqrt{\lambda_1\mu}-x)}e^{(s_2-x)\omega}}{(s_2-x)\cdot\left((\lambda_1+\lambda_2)x+(\sqrt{\lambda_1\mu}-\lambda_1-\lambda_2)^2\right)}dx,\end{aligned}$$

where, in the first equality, the first two integrals are those around $s_1$and s_2 which are equal to 0 and the last two are those from s_2 to s_1 and from s_1 to s_2 respectively. By (15), we obtain $\int_0^\omega f_2^m(t)dt$.

Finally, we get

$$\begin{aligned}&\frac{dE[D_2^\omega]}{d\omega}\\ &= \int_0^\omega f_2^m(t)dt - 1\\ &= \frac{\lambda_1\mu-(\lambda_1+\lambda_2)^2}{\lambda_2(\lambda_1+\lambda_2)}e^{\frac{\lambda_2(\lambda_1+\lambda_2-\mu)}{\lambda_1+\lambda_2}\omega}\cdot\mathbf{1}_{(\lambda_1+\lambda_2)^2>\lambda_1\mu}\\ &\quad + \frac{1}{2\pi}\int_0^{4\sqrt{\lambda_1\mu}}\\ &\quad \times\frac{(\mu-\lambda_1-\lambda_2)\sqrt{x(4\sqrt{\lambda_1\mu}-x)}e^{(s_2-x)\omega}}{(s_2-x)\cdot\left((\lambda_1+\lambda_2)x+(\sqrt{\lambda_1\mu}-\lambda_1-\lambda_2)^2\right)}dx.\end{aligned} \tag{16}$$

Then we derive the average delay of requests of $\{A_2(t)\}_t$ as

$$\begin{aligned}&E[D_2^\omega]\\ &= \frac{\mu}{(\mu-\lambda_1)(\mu-\lambda_1-\lambda_2)}\\ &\quad - \frac{(\lambda_1+\lambda_2)^2-\lambda_1\mu}{\lambda_2^2(\mu-\lambda_1-\lambda_2)}(1-e^{\frac{\lambda_2(\lambda_1+\lambda_2-\mu)}{\lambda_1+\lambda_2}\omega})\cdot\mathbf{1}_{(\lambda_1+\lambda_2)^2>\lambda_1\mu}\\ &\quad - \frac{1}{2\pi}\int_0^{4\sqrt{\lambda_1\mu}}\\ &\quad \times\frac{(\mu-\lambda_1-\lambda_2)\sqrt{x(4\sqrt{\lambda_1\mu}-x)}(1-e^{(s_2-x)\omega})}{(s_2-x)^2\cdot\left((\lambda_1+\lambda_2)x+(\sqrt{\lambda_1\mu}-\lambda_1-\lambda_2)^2\right)}dx.\end{aligned}$$

Because of preemptive priority, the average delay of A_1 requests $E[D_1]=\frac{1}{\mu-\lambda_1}$. Then, we obtain the average delay of all requests by $E[D^\omega]=\frac{\lambda_1}{\lambda_1+\lambda_2}E[D_1]+\frac{\lambda_2}{\lambda_1+\lambda_2}E[D_2^\omega]$ based on the total expectation law.

Now consider $\frac{dE[D_2^\omega]}{d\omega}$.

$$\begin{aligned}&\frac{1}{2\pi}\int_0^{4\sqrt{\lambda_1\mu}}\frac{(\mu-\lambda_1-\lambda_2)\sqrt{x(4\sqrt{\lambda_1\mu}-x)}e^{(s_2-x)\omega}}{(s_2-x)\cdot\left((\lambda_1+\lambda_2)x+(\sqrt{\lambda_1\mu}-\lambda_1-\lambda_2)^2\right)}dx\\ &= C\cdot e^{(-(\sqrt{\mu}-\sqrt{\lambda_1})^2-\xi)\omega},\end{aligned}$$

where C is a negative constant and ξ is a constant between 0 and $4\sqrt{\lambda_1\mu}$ by the mean value theorem. Now in (16), the coefficients of two exponential terms are negative. At the same time, in both exponential terms, the coefficient of ω is

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

also negative. We can find positive constants C_1, C_2, ξ_1, ξ_2 so that $-C_1 \cdot e^{-\xi_1 \omega} < \frac{dE[D_2^\omega]}{d\omega} < -C_2 \cdot e^{-\xi_2 \omega}$. As a result, $E[D_2^\omega]$ decreases exponentially in ω. Because $\frac{dE[D^\omega]}{d\omega} = \frac{\lambda_2}{\lambda_1+\lambda_2}\frac{dE[D_2^\omega]}{d\omega}$, we get that $E[D^\omega]$ decreases exponentially in ω.

REFERENCES

[1] S. Zhang, L. Huang, M. Chen, and X. Liu, "Effect of proactive serving on user delay reduction in service systems," in *Proc. ACM SIGMETRICS*, 2014, pp. 1–2.

[2] S. Zhang, L. Huang, M. Chen, and X. Liu, "Proactive serving decreases user delay exponentially," in *Proc. Workshop Math. Perform. Modeling Anal.*, 2015, pp. 39–41.

[3] R. Kohavi and R. Longbotham, "Online experiments: Lessons learned," *Computer*, vol. 40, no. 9, pp. 103–105, Sep. 2007.

[4] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2012, pp. 1287–1294.

[5] *Kindle Fire*, accessed on Dec. 2013. [Online]. Available: http://www.amazon.com/gp/product/b0051vvob2

[6] G. Gürsun, M. Crovella, and I. Matta, "Describing and forecasting video access patterns," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 16–20.

[7] J. Spencer, M. Sudan, and K. Xu, "Queueing with future information," *Ann. Appl. Probab.*, vol. 24, no. 5, pp. 2091–2142, 2014.

[8] K. Xu and C. W. Chan, "Using future information to reduce waiting times in the emergency department," *Manuf. Service Oper. Manage.*, vol. 18, no. 3, pp. 314–331, 2016.

[9] L. Huang, S. Zhang, M. Chen, and X. Liu, "When backpressure meets predictive scheduling," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2237–2250, Aug. 2016.

[10] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive resource allocation: Harnessing the diversity and multicast gains," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 4833–4854, Aug. 2013.

[11] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive content download and user demand shaping for data networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1917–1930, Dec. 2015.

[12] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Joint smart pricing and proactive content caching for mobile services," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2357–2371, Aug. 2016.

[13] X. Chen and X. Zhang, "A popularity-based prediction model for Web prefetching," *Computer*, vol. 36, no. 3, pp. 63–70, Mar. 2003.

[14] X. Chen and X. Zhang, "Coordinated data prefetching for Web contents," *Comput. Commun.*, vol. 28, no. 17, pp. 1947–1958, Oct. 2005.

[15] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, M. Kyriakakos, and A. Kalousis, "Predicting the location of mobile users: A machine learning approach," in *Proc. Int. Conf. Pervasive Services*, 2009, pp. 65–72.

[16] R. Mayrhofer, H. Radi, and A. Ferscha, "Recognizing and predicting context by learning from user behavior," in *Proc. Int. Conf. Adv. Mobile Multimedia*, 2003, pp. 1–11.

[17] S. Sigg, *Development of a Novel Context Prediction Algorithm and Analysis of Context Prediction Schemes*. Kassel, Germany: Kassel Univ. Press, 2008.

[18] V. S. Tseng and K. W. Lin, "Efficient mining and prediction of user behavior patterns in mobile Web systems," *Inf. Softw. Technol.*, vol. 48, no. 6, pp. 357–369, Jun. 2006.

[19] Y. Xu *et al.*, "Preference, context and communities: A multi-faceted approach to predicting smartphone app usage patterns," in *Proc. ISWC*, 2013, pp. 69–76.

[20] B. D. Higgins *et al.*, "Informed mobile prefetching," in *Proc. MobiSys*, 2012, pp. 155–168.

[21] M. Palmer and S. B. Zdonik, "Fido: A cache that learns to fetch," Dept. Comput. Sci., Brown Univ., Providence, RI, USA, Tech. Rep., 1991.

[22] D. Kotz and C. S. Ellis, "Practical prefetching techniques for parallel file systems," in *Proc. Parallel Distrib. Inf. Syst.*, Dec. 1991, pp. 182–189.

[23] H. Yu and G. Kedem, "DRAM-page based prediction and prefetching," in *Proc. IEEE Comput. Design*, Sep. 2000, pp. 267–275.

[24] S. Zhang, L. Huang, M. Chen, and X. Liu, "Proactive serving decreases user delay exponentially: The light-tailed service time case," Inf. Eng. Dept., Chin. Univ. Hong Kong, Hong Kong, Tech. Rep., 2015. [Online]. Available: http://www.ie.cuhk.edu.hk/~mchen/papers/proactive_serving.tr.pdf

[25] A. O. Allen, *Probability, Statistics, and Queuing Theory: With Computer Science Applications*. Houston, TX, USA: Gulf, 1990.

[26] L. Kleinrock, *Queueing Systems: Computer Applications*, vol. 2. New York, NY, USA: Wiley, 1976.

[27] J. F. C. Kingman, "Inequalities in the theory of queues," *J. Roy. Statist. Soc.*, vol. 32, no. 1, pp. 102–110, 1970.

[28] *Moment Generating Function*, accessed on Dec. 2013. [Online]. Available: https://en.wikipedia.org/wiki/Moment-generating_function

[29] G. Muraleedharan, A. D. Rao, P. G. Kurup, N. U. Nair, and M. Sinha, "Modified Weibull distribution for maximum and significant wave height simulation and prediction," *Coastal Eng.*, vol. 54, no. 8, pp. 630–638, Aug. 2007.

[30] J. F. C. Kingman, "On queues in heavy traffic," *J. Roy. Statist. Soc.*, vol. 24, no. 2, pp. 383–392, 1962.

[31] H. Kobayashi, "Bounds for the waiting time in queuing systems," *Comput. Archit. Netw.*, pp. 263–274, 1974.

[32] *Residue Theorem*, accessed on Dec. 2013. [Online]. Available: http://en.wikipedia.org/wiki/Residue_theorem

[33] *Branch Cut*, accessed on Dec. 2013. [Online]. Available: http://en.wikipedia.org/wiki/Branch_point

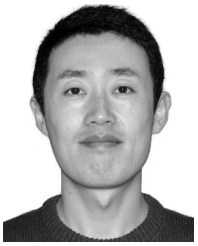
[34] N. R. Draper and H. Smith, *Applied Regression Analysis*. New York, NY, USA: Wiley, 1998.

[35] *Youtube*, accessed on Dec. 2013. [Online]. Available: http://www.youtube.com

[36] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of Internet short video sharing: A YouTube-based measurement study," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1184–1194, Aug. 2013.

[37] L. Kleinrock, *Queueing Systems: Theory*, vol. 1. New York, NY, USA: Wiley, 1975.

[38] I. Adan and J. Resing, *Queueing Theory*. Eindhoven, The Netherlands: Eindhoven Univ. Technology, 2002.



Shaoquan Zhang received the B.Eng. degree from the University of Science and Technology of China, in 2008, and the Ph.D. degree in information engineering from The Chinese University of Hong Kong, Hong Kong, in 2014. His research interests include networked system analysis and algorithm design, and distributed and stochastic network optimization.



Longbo Huang received the Ph.D. degree in electrical engineering from the University of Southern California in 2011. He was a Post-Doctoral Researcher with the Electrical Engineering and Computer Sciences Department, University of California (UC), Berkeley, CA, USA, from 2011 to 2012. He was a Visiting Scholar with the LIDS Lab, MIT, in 2012 and 2014 and with the EECS Department at UC Berkeley, in 2013. He was also a Visiting Professor with the Institute of Network Coding, The Chinese University of Hong Kong, in 2012 and at Microsoft Research Asia in 2015. Since 2012, he has been an Assistance Professor with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. His current research interests are in the areas of learning and optimization, network control, data center networking, and smart grid and mobile networks. He was selected into China's Youth 1000-talent Program in 2013 and received the Outstanding Teaching Award from Tsinghua University in 2014.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



Minghua Chen (S'04–M'06–SM'–13) received the B.Eng. and M.S. degrees from the Department of Electronic Engineering, Tsinghua University, in 1999 and 2001, respectively, and the Ph.D. degree from the Department of Electrical Engineering and Computer Sciences, University of California (UC), CA, USA, Berkeley, in 2006. He spent one year Visiting Microsoft Research Redmond as a Post-Doctoral Researcher. He joined the Department of Information Engineering, The Chinese University of Hong Kong in 2007, where he is currently an Associate Professor. He is also an Adjunct Associate Professor with the Institute of Interdisciplinary Information Sciences, Tsinghua University. His recent research interests include energy systems (e.g., smart power grids and energy-efficient data centers), distributed optimization, multimedia networking, wireless networking, network coding, and delay-constrained network information flow. He received the ELI JURY AWARD from UC Berkeley in 2007 (presented to a graduate student or recent alumnus for outstanding achievement in the area of systems, communications, control, or signal processing) and The Chinese University of Hong Kong Young Researcher Award in 2013. He also received several best paper awards, including the IEEE ICME Best Paper Award in 2009, the IEEE TRANSACTIONS ON MULTIMEDIA Prize Paper Award in 2009, and the ACM Multimedia Best Paper Award in 2012. He is currently an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING. He serves as a TPC Co-Chair of ACM e-Energy 2016 and a General Co-Chair of ACM e-Energy 2017.



Xin Liu received the Ph.D. degree in electrical engineering from Purdue University in 2002. She is currently a Professor with the Computer Science Department, University of California at Davis. From 2012 to 2014, she was with the Wireless Networking Group, Microsoft Research Asia. She has studied cellular scheduling algorithms, cognitive radio networks, and wireless mesh networks. Her current research focuses on data-driven approaches in networking. She has received the NSF CAREER Award (2005), the Outstanding Engineering Junior Faculty Award from the UC Davis College of Engineering (2005), and the Chancellor's Fellowship (2011).