Proactive Serving Decreases User Delay Exponentially

ZHANG, Shaoquan

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Information Engineering

The Chinese University of Hong Kong
August 2014

Abstract

In online service systems, delay experienced by a user from the service request to the service completion is one of the most critical performance metrics. To improve user delay experience, in this thesis, we investigate a novel aspect of system design: *proactive serving*, where the system can predict future user request arrivals and allocate its capacity to serve these upcoming requests proactively. This approach is complementary to the conventional capacity boosting mechanism and is motivated by recent industrial and academic advances. In particular, we investigate the fundamentals of proactive serving from a queuing theory perspective.

First, most importantly, we show that under proactive serving the average user delay decreases *exponentially* (in the prediction window size) for a wide range of queuing models. Furthermore, the delay reduction is robust against prediction errors. We also show that both the variance of user delay and the tail of user delay decrease exponentially under proactive serving, which are also important user delay experience metrics.

We then show that proactive serving is more effective in decreasing user delay than capacity boosting in light workload regime. In particular, the average user delay decays inverse-proportionally in system capacity, but exponentially in the prediction window size in proactive serving.

Finally we demonstrate how to leverage proactive serving in system design from a optimization point of view, *e.g.*, how many resources are dedicated to proactive serving. The results provide useful engineering insights to system

designers.

Our trace-driven simulation results demonstrate the practical power of proactive serving: for example, under the YouTube data trace of 1000 different videos, the average user delay can be decreased by 50% when the system predicts 100 seconds ahead. Our results provide useful insights for proactive serving and justify its increasing applications in practical systems.

摘要

對於在線服務系統，由於系統服務造成的用戶延遲是衡量系統性能的重要指標。提高用戶的延遲體驗的傳統方法是提高系統中服務器的性能。在本論文中，我們研究一種新穎的叫做“前瞻性服務”的方法用來提高用戶的延遲體驗。前瞻性服務是指系統在預測用戶需求的基礎上，在用戶產生需求之前，系統已經將服務送到用戶手中。前瞻性服務是傳統方法的有力補充。我們從排隊論的角度研究前瞻性服務對用戶的延遲的提高。

首先，對於多种排隊系統，我們證明前瞻性服務能夠指數性降低用戶的平均延遲。而且前瞻性服務對於用戶需求預測的誤差具有魯棒性。 我們同時也證明了前瞻性服務能夠指數性降低用戶的延遲方差和尾概率。

然後，我們證明前瞻性服務在系統低负载时比傳統方法在降低用戶的平均延遲上更加有效。 前瞻性服務能夠指數性降低用戶的平均延遲。而通過提高系統服務器性能，只能反比例降低用戶的平均延遲。

最後，我們從優化的角度分析怎樣在系統設計中利用前瞻性服務，給系統設計者提供有用的建議。

我們基於實際數據的仿真結果驗證了前瞻性服務在實際系統的作用。例如，基於Youtube數據的仿真表明，如果系統能提前一百秒預測用戶的需求，那麼前瞻性服務能夠降低一半的用戶延遲。

Acknowledgement

PhD study is a long journey. It is my good fortune to have Prof. Minghua Chen as my supervisor in my journey. I would like to take this opportunity to express my deep gratitude to him. During my PhD study, he shows me a good example of excellent researcher. His enthusiasm in research inspires me to pursue what makes me excited and tackle all the fronting challenges. What I have learned from him is not just how to do research, but his way of living and his insights towards the world. He has always been my source of encouragement and spiritual support. Thanks to him, my PhD life is challenging and enjoyable.

I am also sincerely grateful to Prof. Longbo Huang who hosted me during my visit at Tsinghua University, and Dr. Laurent Massoulie and Dr. Fabio Picconi who hosted me during my internship in Technicolor Research Paris. I have wonderful off-campus times under their great help. Also their broad knowledge and deep insights in research benefit me a lot in my PhD study.

I also want to thank my collaborators in my PhD study: Prof. Mung Chiang, Prof. Xin Liu, Prof. Zongpeng Li, Dr. Wenjie Jiang, Dr. Ziyu Shao, Dr. Libin Jiang. Thanks to their great suggestions and big efforts, we get some good works published. It is my pleasure to work with all of them.

My friends at CUHK make my PhD life full of joy. They are Liang Chen, Wei Chen, Yue Wang, Ziyu Shao, Jihang Ye, Zhe Zhu, Tan Lu, Xiangwen Chen, Sheng Cai, Lian Lu, Jinlong Tu, Jincheng Zhang, Hanxu Hou, Yang Yang, Benedit Max, Wenjie Zhang, Guanglin Zhang, Xu Chen, Lingjie Duan

and Lei Zhu. Many thanks, guys.

Finally, I would like to thank my parents for their warm and endless love. Their love makes me stronger and keeps me forward forever. I dedicate this thesis to them.

This work is dedicated to my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The fast growing number of personal devices with Internet access, *e.g.*, smart mobile devices, has led to the blossom 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 a user from the service request to the service completion is one of the most critical performance metrics. For example, experiments at Amazon showed that every 100-millisecond increase in load time of Amazon.com would decrease revenues by 1% [31]. Google also found that an extra 0.5 seconds in search page generation time dropped traffic by 20% [31].

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

Recently, both industrial practice and academic studies suggest *proactive serving*, *i.e.*, serving future requests before they arrive, as a modern approach for decreasing user delay. Proactive serving is based on the key observation

on service request predictability. It is a technique that has been widely used in computer systems based on what is likely to happen next, such as cache pre-loading and command pre-fetching [1]. Similarly, in cloud service systems, it is not atypical to have predictable service requests. For example, in cloud computing platforms, service jobs, such as indexing, page-ranking, backup, crawling, and maintenance-related load, are often predictable. In fact, in an industrial grade cloud computing system, we observe a significant portion of the workload to be periodic and thus predictable [28]. Intuitively, if one user is observed to watch football news consistently in the morning, then such contents can be preloaded in the future.

In practice, Amazon launched *Amazon Silk*, a mobile web browser in its tablet Kindle Fire [4]. All the web traffic from the browser goes through the Amazon cloud. The cloud uses machine learning techniques to predict what users will browse and pre-loads web pages that are likely to be requested in the future. In this manner, when the user clicks on the corresponding content, the loading is instant and the user delay is decreased to zero. This technique speeds up request responses and improves user browsing experience.

All the above exciting developments suggest proactive serving as a new design mechanism for decreasing user delay: based on user request arrival prediction, the system can allocate its capacity proactively and pre-serve future requests to decrease the delay experienced by users. This observation naturally leads to two fundamental questions:

- How much user delay reduction can we obtain by proactive serving?
- How does proactive serving compare to capacity boosting in decreasing user delay?

In this thesis, we explore answers to the above open questions and investigate the fundamentals of proactive serving from a queuing theory perspective. In

[1] Detailed description can be found in section 2.1

particular, we study proactive serving with a prediction window of size ω, where one has the ability to predict future requests in a time window of ω and serve them if needed. We investigate the impact of proactive serving on decreasing user delay as a function of the prediction window size ω. We also consider the case of imperfect prediction.

Challenge. 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 (based on perfect or imperfect prediction) using a multi-dimensional Markov chain, and 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 the 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 observation suggests that we can first obtain the delay distribution without proactive serving, which is usually a less-complicated task, and then derive the desired one with proactive serving. The approach is simple yet can reveal insights behind the delay reduction by proactive serving. More importantly, the approach can be generalized to various queuing models.

Second, even with our new approach, characterizing user delay with proactive serving in the presence of prediction errors is still non-trivial. We carefully model the system behavior under two types of prediction errors, namely miss detection and false alarm, as priority queues, and apply residue theorem in a highly involved derivation process to obtain closed-form expressions for the average user delay.

Contribution. We make the following contributions.

$\triangleright$ In chapter 3, we present the first set of queuing models for service systems with proactive serving capability. In our models, there are user

request arrivals, servers, a queue, and a prediction window of size ω (modeled as a pipe). User requests pass through the pipe before entering the queue. The system can serve the requests in both the queue and the pipe. The time a request spent in the pipe is excluded from its delay computation. We develop corresponding taxonomy and notations. These models enable the use of queuing theory tools to characterize the delay reduction benefit of proactive serving. We focus on stable queuing systems where arrivals happen slower than service completions.

$\triangleright$ In chapter 4, we first show that, for stable M/M/1 queuing systems with proactive serving, the average user delay decreases *exponentially* in the prediction window size ω. More importantly, based on the insights from the M/M/1 systems, we then extend the analysis to G/G/1 queuing systems and show that proactive serving can still decrease the average queuing delay *exponentially*.

$\triangleright$ In chapter 5, we compare proactive serving to capacity boosting and show that proactive serving is more effective in decreasing user delay in light workload regime. In particular, for stable M/M/1 systems, the average user delay decays inverse-proportionally in system capacity, but exponentially in the prediction window size ω in proactive serving.

$\triangleright$ In chapter 6, we show that proactive serving can decrease both the variance of user delay and the tail of user delay exponentially in the prediction window size ω, which are also important factors that impact user delay experience.

$\triangleright$ In chapter 7, we incorporate prediction errors (in terms of miss detection and false alarm) into proactive serving and consider more realistic settings. We show that the effect of proactive serving on user delay reduction is robust against prediction errors.

$\triangleright$ In chapter 8, from a optimization point of view, we show how to leverage proactive serving in system design. In particular, we consider two specific

problems, namely how to balance the benefit and cost of proactive serving and how to leverage both proactive serving and capacity boosting to improve user delay experience. The results we obtain provide useful engineering insights to system designers.

▷ We evaluate the performance of proactive serving using simulations based on real-world traces in chapter 10. Specifically, under the *YouTube* data trace of 1000 different videos, user delay can be decreased by 50% when the system can predict 100 seconds ahead. Under *Youku* data trace of bursty request arrivals, user delay can be decreased by 50% when the system can predict 12 minutes ahead, which is highly feasible according to the study on predicting user requests for an industrial-grade VoD system [23]. Overall, the simulation results suggest that in addition to capacity boosting, proactive serving is indeed a powerful mechanism for improving user delay experience in practice.

Chapter 2

Background

2.1 Proactive Serving

Proactive serving is a technique, based on system's request predictability, that serves future service requests before they arrive at the system. This technique has seen industrial practices and also has been studied in academics. Proactive serving can be cache pre-loading, which means that the system can send contents to users' caches before users request them. A good example is Amazon's Kindle Fire [4]. All the web traffic from Kindle Fire goes through the Amazon cloud. Based on cached user traffic, the servers in the cloud use machine learning techniques to predict what users will browse and pre-load web pages to users' tablets. Proactive serving can be prefetching in computing systems [41], [34]. Prefetching means that data or instructions are preloaded into memory before they are actually requested. Another example of proactive serving is used in computing management where the computing resource managing unit pre-computes some information in case some later applications request them, *e.g.*, branch prediction in computer architecture [33], [22].

2.2 Related Work

In [47], the authors study how future information 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-traffic 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 equal to that when we know all the future. In [37], [18] [36], the authors design online request scheduling algorithms to reduce request waiting times based on request future information. The future information includes request arrival time and service time. In their settings, when the system sees future requests, the system cannot pre-serve them but the only choice is to decide whether to keep server idle and wait for them.

When the system can pre-serve future requests based on prediction, authors in [49] show that the probability of server outage in wireless networks decreases linearly with the size of look-ahead window. They also show that appropriate use of future information by primary users can improve the gain of the secondary network at no cost of primary users in cognitive networks. In [48] they explore the idea of proactive serving in data networks to shape user demand. In [15] and [16], authors provide a prediction model and develop techniques for proactively serving web contents.

In addition, many works exist on proactive serving mobile content based on predicting mobile user traffic and mobility patterns, *e.g.*, in [9], [38], [46], [50], [54]. In [24], the authors present a system architecture for mobile prefetching where informed prefetching is structured as a library to which any mobile application may link. Experiment results show significant benefit of prefetching in reducing access delay and energy consumption. Breadcrumb [40] and smarttransfer [51] address the issue of predicting future network conditions and user profile. These experimental studies motivate the

necessity to address the fundamentals of proactive serving.

Proactive serving is also a well-studied mechanism in general computer systems. Authors in [42, 32, 56] explore prediction and prefetching for file systems, databases, and DRAM respectively. [52, 19, 43] address the theoretical problem in prefetching including cost benefit analysis.

Instead of using proactive serving to decrease user delay by exploiting time domain information, some works exploit user domain information to improve user delay experience. For example, [39], [11], [45], [44], [27], [10] design algorithms by scheduling user traffic to improve user delay experience in wireless networks. Some works leverage coding scheme to decrease delay in various systems, *e.g.*, [21], [14], [35], [20], [25].

This thesis is motivated by the above mentioned advances in industry and academic. To the best of our knowledge, we are the first to investigate the fundamentals of proactive serving from a queuing theory perspective.

Chapter 3

Model

3.1 Service System without Proactive Serving

Consider a service system shown in Fig. 3.1. In the system, backend servers provide service to incoming user requests that arrive at the system according to a 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. When a user request arrives and there is idle service capacity, the request will be served. Otherwise, the request waits in the queue $Q(t)$ for service. After being served, the request leaves the system. We define user delay as the time from when the user request arrives at the system till it leaves the system.

Traditionally, queuing theory has been applied to model such a system and study its performance. In particular, queuing theoretic analysis suggests

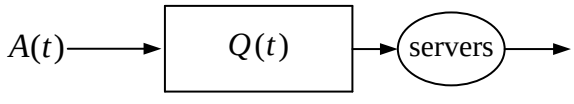

Figure 3.1: A single queue service system.

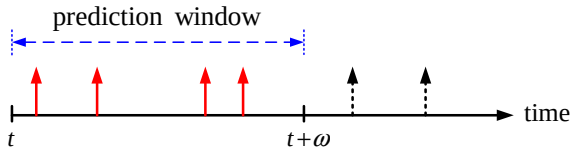

Figure 3.2: Prediction model: Each upright arrow represents a future request arrival. At time t, the system knows in $(t, t+\omega)$ the request arrival epochs (red solid arrows) and the corresponding users who generate these requests by its prediction mechanism.

that boosting the system capacity is a principal mechanism to decrease the average user delay. For example, one can use the standard M/M/1 queuing model to represent the service system in Fig. 3.1, and it is well known that for such a system the average user delay decreases inverse-proportionally in the service capacity.

3.2 Service System with Proactive Serving

We now consider a service system that can proactively serve future user requests based on arrival prediction. For the ease of presentation, we consider perfect prediction of arrivals in this chapter and chapter 4, and study imperfect prediction in chapter 7.

As shown in Fig. 3.2, we assume that the system can predict user request arrivals ω time ahead, or users indicate their arrivals to the system ω time ahead[1]. In either case, at time t, the system knows exactly in $(t, t+\omega)$ the request arrival epochs and the corresponding users who generate the requests. Meanwhile, we do not assume the knowledge of the workload of each user request.

[1]For instance, in video-on-demand systems a user can request the delivery of a movie that he or she wants to watch ω time ahead, for better quality of video-watching experience or discounted cost.

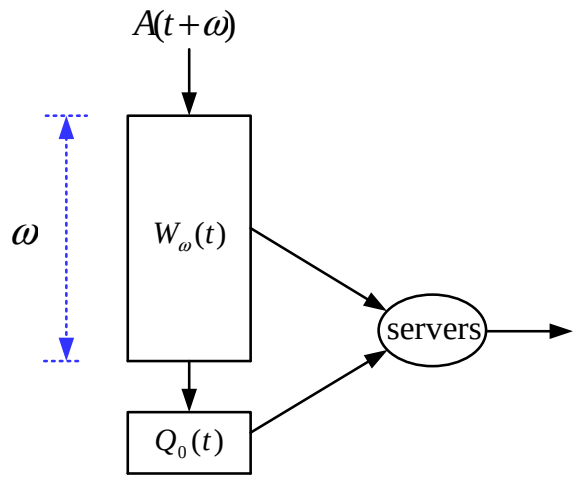

Figure 3.3: Service system with perfect prediction: $Q_0(t)$ represents the queue of the requests that have arrived at the system and are waiting for service at time t. $W_\omega(t)$ is the prediction window of size ω. Each arrived user request first goes through the prediction window $W_\omega(t)$ and then enters the queue $Q_0(t)$. The servers can observe and serve the requests in both $Q_0(t)$ and $W_\omega(t)$.

Based on the arrival prediction, the system can allocate its service capacity to serve future requests proactively. Specifically, the servers can provide service to the users who will generate requests in the future. The user requests that get pre-served will not enter the system. Such a proactive serving model captures service systems that can perform cache pre-loading or command pre-fetching. As a practical example of cache pre-loading, the Amazon cloud can predict web page requests and pre-load desired web pages to users' Kindle Fire tablets beforehand. When the user clicks on the predicted content, it gets the content immediately.

We depict the service system which can proactively serve future requests based on perfect prediction in Fig. 3.3. Let $Q_0(t)$ represent the queue of the requests that have arrived at the system and are waiting for service at time t, and $W_\omega(t)$ be 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 before that. The requests will not queue up in $W_\omega(t)$. Thus $W_\omega(t)$ should be viewed as a pipe. 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 $Q_0(t), W_\omega(t)$ to denote its corresponding size. For example, $W_\omega(t)$ can represent the number of arrivals within the prediction window which have not been pre-served at time t.

3.3 Queuing Models for Service System with Proactive Serving Capability

In this thesis, 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 classical queuing model, requests arrive randomly at the system. If all servers are busy, requests wait in the queue for service. Servers serve requests according to certain service policy. The time it takes 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 the requests in both the queue and the pipe according to certain service policy.

Remark that our new model is applied to service systems which can be modeled as queuing systems and have proactive serving capabilities.

In 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

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

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). In particular, $[\omega]$ denotes that the system can know future arrivals ω time ahead and serve them proactively.

For example, one can use $\text{M/M/1}^{[\omega]}$/FCFS to model the system in Fig. 3.3. Both the inter-arrival time and the service time follow exponential distributions. There is a single server in the system. The system can know future arrivals ω time ahead and serve them proactively. When the server becomes idle, it will first check $Q_0(t)$. If $Q_0(t) > 0$, the server serves the request at the head of $Q_0(t)$. If $Q_0(t) = 0$, the server will then check the prediction window $W_\omega(t)$, and pre-serve the earliest request in it.

A queuing system is stable if arrivals happen slower than service completions, and is unstable otherwise. Stability of a queuing system is thus determined by the average arrival and service rates, and is not affected by proactive serving. Unstable systems do not have steady-state distributions and consequently do not have well-defined average user delays. 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 thesis, unless mentioned otherwise. We evaluate the user delay performance under the Processor Sharing policy in simulation and observe results similar to that under the FCFS policy.

Chapter 4

Proactive Serving with Perfect Prediction

In this chapter, we study the user delay for two models: M/M/1$^{[\omega]}$ and G/G/1$^{[\omega]}$. We first start from the simple model M/M/1$^{[\omega]}$ and characterize the average user delay. Then we extend the analysis to G/G/1$^{[\omega]}$ based on the insights obtained from analyzing M/M/1$^{[\omega]}$. We observe that the average delay decreases exponentially under proactive serving.

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

In an M/M/1$^{[\omega]}$ system, 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 future arrivals ω time ahead and serve them proactively. M/M/1$^{[\omega]}$ can be used to model systems whose capacity is pooled together to serve incoming requests. For example, content distribution networks pool CPU cycles, bandwidth of servers in multiple data centers to load-balance and serve worldwide users [53]. Define $\rho = \frac{\lambda}{\mu}$ and D^ω as the user delay.

When $\omega = 0$, *i.e.*, without proactive serving, the system reduces to the classical M/M/1 queue. It's well known that the average user delay is given by

$$E\left[D^0\right] = \frac{1}{\mu - \lambda}. \tag{4.1}$$

The probability density function of D^0 is also known as [8]

$$f_{D^0}(t) = (\mu - \lambda)e^{-(\mu-\lambda)t}, t \geq 0. \tag{4.2}$$

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 both the 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 the Markov chain. The other is that it is hard to generalize the method to more complex models. Detailed discussions can be found in Appendix A.3.

Instead of applying the generic method, we develop a new approach to analyze the average user delay. To analyze the user delay for $\mathrm{M/M/1}^{[\omega]}$, we first prove an interesting result that $Q^{sum}(t) \triangleq Q_0(t) + W_\omega(t)$ evolves the same as an M/M/1 system with a properly initialized queue. Based on this observation, the distribution of user delay under proactive serving, *i.e.*, D^ω, turns out to be a "shifted" version of that of user delay without proactive serving as shown in (4.2). Once we know the distribution of D^ω, we can compute the average user delay $E\left[D^\omega\right]$.

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

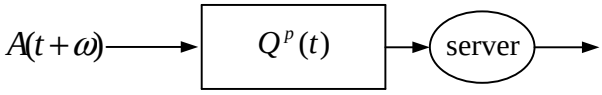

Figure 4.1: M/M/1: The arrival process is $\{A(t+\omega)\}_t$. Service times of requests are independent and identically exponentially distributed with mean $1/\mu$. The initial value $Q^p(0)$ is $|A(0:\omega)|+Q_0(0)$. The service policy is FCFS.

the FCFS policy. This essentially mimics an M/M/1 system, and intuitively, the size of the group evolves statistically the same as the queue size of the M/M/1 system. Our following lemma confirms this observation.

Lemma 1. *Define $Q^p(0) = |A(0:\omega)| + Q_0(0)$, where $A(0:\omega) = \{A(\tau), 0 < \tau \leq \omega\}$ is the set of arrivals from time 0 to time ω and $|A(0:\omega)|$ is the size of $A(0:\omega)$, and $Q^p(t)$ is an M/M/1 queue with initial value $Q^p(0)$ as shown in Fig. 4.1. We have*

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

Proof. See Appendix A.1. □

Lemma 1 reveals a useful observation: the total time that a user request spends in the M/M/$1^{[\omega]}$ system, *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 system. As discussed at the end of section 3.2, the time spent in $W_\omega(t)$ is excluded from the user delay calculation. Then the user delay in M/M/$1^{[\omega]}$ system, *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/$1^{[\omega]}$ system in the following lemma.

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

$$f_{D^\omega}(t) = f_{D^0}(t+\omega) \text{ when } t > 0$$

and

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

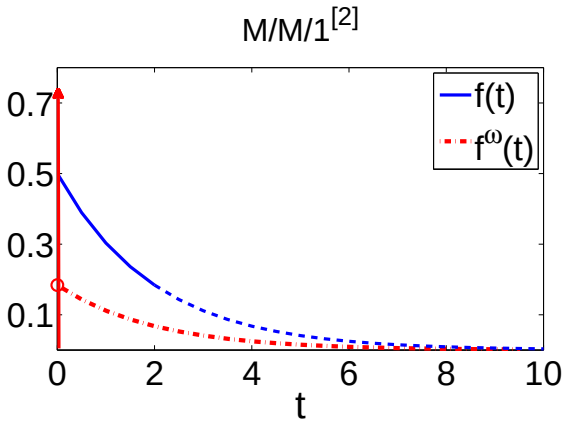

Figure 4.2: Probability density function (PDF) of user delay with/without future prediction: The PDF under perfect prediction (the vertical arrow at the origin and the dash curve) can be obtained by shifting the PDF without prediction (dash curve) ω units left, where $\omega = 2$.

Proof. See Appendix A.2. □

An illustrating example of the distribution derived in Lemma 2 is shown in Fig. 4.2.

Although Lemma 1 and 2 are established for M/M/$1^{[\omega]}$ system, they can be directly applied to G/G/$1^{[\omega]}$ system as shown in the following Corollary, which will be useful in next section.

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

$$f^G_{D^\omega}(t) = f^G_{D^0}(t+\omega) \text{ for } t > 0 \text{ and } \Pr(D^\omega = 0) = \int_0^\omega f^G_{D^0}(t)dt.$$

Proof. See Appendix A.4. □

Lemma 2 allows us to obtain the distribution of D^ω from that of D^0. Based on this lemma, we obtain the average user delay $E[D^\omega]$ as follows.

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

$$E[D^{\omega}] = \frac{1}{\mu - \lambda} e^{-(\mu-\lambda)\omega}. \tag{4.3}$$

Proof. See Appendix A.3. □

Theorem 1 reveals that the average user delay decays exponentially in the prediction window size ω[1]. This indicates that little future information can improve user delay experience tremendously.

From Theorem 1, in the heavy-load regime where the arrival rate λ is close to the service rate μ, (4.3) can be approximated by $\frac{1}{\mu-\lambda} - \omega$ when ω is small, which is linear in ω. As contrast, when the system is in the light-load regime, (4.3) 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 thus most of the serve capacity can be spared to serve future requests. Consequently, many requests get 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 time. As a result, the chance that a request gets served proactively by the server is small when going through a small prediction window. Therefore, proactive serving has limited delay reduction capability. When the system is in the heavy-load regime, to achieve small user delay, the system needs to guarantee a sufficient large prediction window so that the probability that a request gets served proactively is high enough before entering the queue.

[1]The amount of delay reduction by proactive serving is $\frac{1}{\mu-\lambda}(1 - e^{-(\mu-\lambda)\omega})$.

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

In this section, we extend the analysis to the general queuing system G/G/1$^{[\omega]}$. 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, and service times of user requests are also independent, identically and generally distributed with mean $\frac{1}{\mu}$ and variance σ_μ^2. Meanwhile, inter-arrival times and service times are independent. The system can predict future arrivals ω 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$^{[\omega]}$ systems, the distribution of service time is general. As a result, the distribution of user delay is general. As shown by Corollary 1, proactive serving essentially shifts the distribution left by ω unit. Therefore, we cannot observe exponential decreasing of user delay under proactive serving as we did in section 4.1[2]. Instead, in this section, we focus on the effect of proactive serving on queuing delay and show that proactive serving decreases the average queuing delay exponentially.

Let QD^ω denote the user queuing delay when the system predicts ω 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 open in queuing theory. However, people have shown that the tail of the distribution of QD^0 can be upper-bounded [30]. Let x denote the inter-arrival time random variable and y denote the service time random variable. Let $U(s)$ denote the Laplace-Stieltjes transform of the random variable $y - x$. Define

$$s_0 = \sup\{s > 0 : U(-s) \leq 1\}. \tag{4.4}$$

[2]Note that proactive serving can still clear the average user delay as long as the system predicts sufficiently far.

By [30], s_0 always exists. Given a positive $t > 0$, the tail of the distribution of QD^0, *i.e.*, $\Pr(QD^0 \geq t)$, can be upper-bounded as follows

$$\Pr(QD^0 \geq t) \leq e^{-s_0 t}. \tag{4.5}$$

This upper-bound (4.5) on the tail of the distribution of QD^0 enables us to prove that proactive serving can exponentially decrease the average queuing delay, *i.e.*, $E[QD^\omega]$. Based on the same idea as developed in section 4.1, we obtain the following theorem.

Theorem 2. *Assume $\mu > \lambda$. The average queuing delay of $G/G/1^{[\omega]}$ with perfect prediction is given by*

$$E[QD^\omega] \leq \frac{1}{s_0} e^{-s_0 \omega}. \tag{4.6}$$

Proof. See Appendix A.5. □

Theorem 2 reveals that, regardless of inter-arrival time distribution and service time distribution, proactive serving is so powerful that it can decrease the average queuing delay exponentially.

Same as the distribution of queuing delay, the explicit form of s_0 is also open. As a result, Theorem 2 does not tell how the inter-arrival time distribution and the service time distribution impact the average queuing delay under proactive serving. Instead, we simulate $G/G/1^{[\omega]}$ under different inter-arrival time distributions and service time distributions. The results of the average queuing delay are shown in Fig. 4.3. As seen from the figure, the average queuing delay always decreases exponentially in ω, which aligns with Theorem 2. Under the same mean, different distributions have different variances. For example, the variance of Weibull distribution[3] is larger than that of Exponential distribution, and the variance of Exponential distribution is larger than that of Uniform distribution. We can see from Fig. 4.3 that the average

[3]Note that the Weibull distribution has a heavy tail.

inter-arrival time distribution	Uniform	Exponential
variance	2.08	6.25

service time distribution	Uniform	Exponential	Weibull
variance	1.33	4	20

Table 4.1: Variance of inter-arrival time distributions and service time distributions. The upper table shows the variance of different inter-arrival time distributions. The lower table shows the variance of different service time distributions. The mean arrival rate is $\lambda = 0.4$. The mean service rate is $\mu = 0.5$. For the Uniform distribution, boundaries are 0 and double mean. For the Weibull distribution, the scale parameter is equal to 1.

queuing delay under the Weibull distribution is larger than other cases. The average queuing delay under the Uniform distribution is smaller than other cases. This indicates that the average queuing delay is an increasing function in variance.

In Tab. 4.2, we show the decaying rate of the average queuing delay under the same setting as Fig. 4.3. The decaying rate is defined as $-\frac{dE[QD^{\omega}]}{d\omega} \cdot \frac{1}{QD^{\omega}}$. The larger the decaying rate, the faster the delay decreases. As seen, the variance of the inter-arrival time distribution or the service time distribution impacts how fast the average queuing delay decays. Under the distribution with larger variance, the decaying rate is smaller. This indicates that the speed that the average queuing delay decreases is a decreasing function in variance.

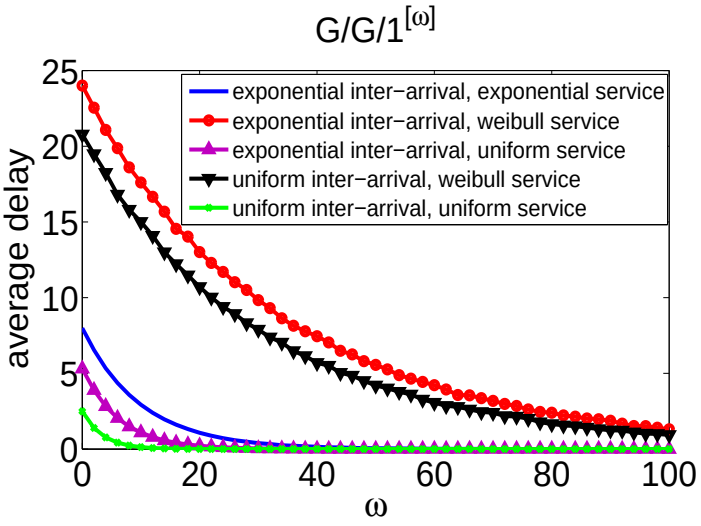

Figure 4.3: The average queuing delay vs how far we predict the future under perfect prediction under different inter-arrival time distributions and service time distributions. 'Exponential' means that the corresponding distribution is an Exponential distribution. 'Uniform' means that the corresponding distribution is a Uniform distribution. 'Weibull' means that the corresponding distribution is a Weibull distribution. We choose $\lambda = 0.4$ and $\mu = 0.5$. For the Uniform distribution, boundaries are 0 and double mean. For the Weibull distribution, the scale parameter is equal to 1. Under the settings, the Uniform distribution has the smallest variance. The Weibull distribution has the largest variance. The variance of the Exponential distribution is in the middle. The variance of different distributions under the settings are summarized in Tab. 4.1. As seen for the figure, first, the average queuing delay always decreases exponentially in ω. Second, the average queuing delay is small under inter-arrival time distributions with small variance or service time distributions with small variance.

inter-arrival time distribution, service time distribution	delay decaying rate
Uniform inter-arrival, Uniform service	0.28
Exponential inter-arrival, Uniform service	0.15
Exponential inter-arrival, Exponential service	0.1
Uniform inter-arrival, Weibull service	0.03
Exponential inter-arrival, Weibull service	0.028

Table 4.2: The decaying rate of the average queuing delay. Under the distribution with larger variance, the decaying rate is smaller. The table shows that the speed that the average queuing delay decreases is a decreasing function in variance.

Chapter 5

Comparison With Capacity Boosting

In this chapter, based on the results for the M/M/$1^{[\omega]}$ system, we compare capacity boosting and proactive serving (with perfect prediction) as two principal design mechanisms for decreasing user delays.

For the M/M/1 system without proactive serving, the average user delay with service capacity m is given by

$$1/(m \cdot \mu - \lambda). \tag{5.1}$$

By comparing it with the average delay of the M/M/$1^{[\omega]}$ system with proactive serving in (4.3), we obtain the amount of prediction (measured in prediction window size) needed to obtain the same delay reduction as boosting the capacity by $m - 1$ times. Denote the corresponding prediction window size as $\omega^*(m)$, we have the following theorem.

Theorem 3. *Assume $\lambda < \mu$. For the M/M/$1^{[\omega]}$ system with perfect prediction, $\omega^*(m)$ $(m \geq 1)$ is given by*

$$\omega^*(m) = \frac{1}{\mu - \lambda} \ln \frac{m - \rho}{1 - \rho}. \tag{5.2}$$

Proof. See Appendix A.6. □

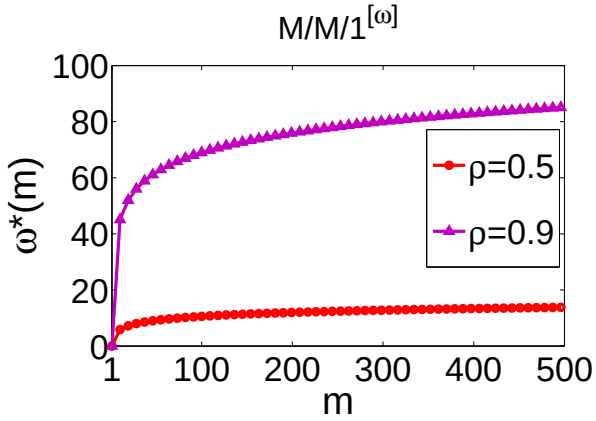

Figure 5.1: How far do we need predict in order to achieve the same delay performance as compared to the server capacity being increased by $m-1$ times in the M/M/1$^{[\omega]}$ system?

We plot the $\omega^*(m)$ as a function of m in Fig. 5.1 under different ρ (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, which is critical for online services. 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 about **500** times when $\rho = 0.5$.

In Fig. 5.1, 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 serve capacity can be spared to serve future requests. So, the power of proactive serving gets fully performed. As a result, most of requests get served proactively and thus experience zero delay. To achieve the same effect, the system capacity needs to be increased

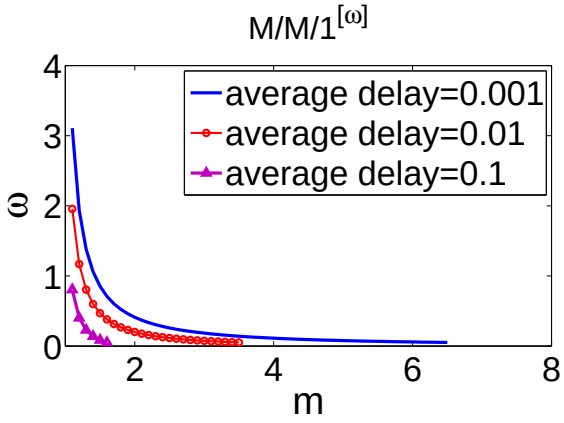

Figure 5.2: To achieve the same delay performance, the system can select different combinations of proactive serving and system capacity boosting.

significantly.

It is also conceivable to combine proactive serving and capacity boosting to achieve a desired average user delay for a system. For example, when $\lambda = 9$ and $\mu = 10$, to obtain that the average user delay of 0.1 unit time, the system can serve proactively 2.3 unit time ahead without boosting the service capacity, or it can serve proactively 0.8 unit time ahead and boosting the service capacity by 10%. We plot different combinations of ω (representing proactive serving capability) and m (representing service capacity) in Fig. 5.2 when the system has different desired delay targets. All the combinations of ω and m on the same isoline give the same average user delay, and the system designer can select the best combination based on the average user delay requirement and various resource constraints. Details are discussed in chapter 8.

Chapter 6

Other Delay Performance Measures

In chapter 4, we show that proactive serving can decrease the average user delay exponentially. In this chapter, we study the impact of proactive serving on the user delay performance from other important aspects. Specifically, we study the variance of user delay and the tail of user delay under proactive serving.

As shown by Lemma 2 and Corollary 1, proactive serving shifts the delay distribution left as a whole. As a result, for any given positive values d and τ, $\Pr(d < D^{\omega} < d + \tau)$ decreases when ω grows. Therefore, one can imagine that, under proactive serving, not only does the average user delay (the first moment) decrease, but also the variance (the second moment related) and the tail of user delay decrease. In this chapter, we show that proactive serving can decrease both the variance of user delay and the tail of user delay exponentially, which further demonstrates the power of proactive serving on improving the user delay experience.

6.1 Variance of User Delay

As suggested by [17], [26], delay variance is an important factor that affects user experience in Video-on-Demand systems. The variance of user delay reflects the fluctuation of delays that users experience on the system side. Small variance corresponds to low fluctuation and thus better user experience.

Let $Var[D^\omega]$ denote the variance of the user delay D^ω. For the M/M/$1^{[\omega]}$ system, Lemma 2 characterizes the distribution of D^ω. Based on that, we have the following lemma for the delay variance $Var[D^\omega]$.

Lemma 3. *Assume that $\lambda < \mu$. The variance of user delay in M/M/$1^{[\omega]}$ is given by*

$$Var[D^\omega] = Var[D^0] \cdot e^{-(\mu-\lambda)\omega} \cdot \left[2 - e^{-(\mu-\lambda)\omega}\right], \tag{6.1}$$

where $Var[D^0]$ is the delay variance without proactive serving.

Proof. See Appendix A.7. □

From Lemma 3, we can see that proactive serving can decrease the variance of user delay exponentially.

The delay variance without proactive serving is

$$Var[D^0] = \frac{1}{(\mu-\lambda)^2}. \tag{6.2}$$

As comparison, when boosting the server capacity, the delay variance decreases inverse-proportionally to the square of the service capacity. This suggests that proactive serving is more effective in decreasing the delay variance than capacity boosting as we observed for the average delay in chapter 5.

6.2 Tail of User Delay

The delay tail is defined as the probability that user delay is larger than certain value, *i.e.*, $DT^\omega(d) = \Pr(D^\omega > d)$ for given positive value d. By

the definition, it measures the fraction of users in the system that experience large delay. The larger the tail, the worse the user experience. It is important for system designers to decrease delay tail, especially long delay tail [55], [57].

For the M/M/$1^{[\omega]}$ system, we have the following lemma for the delay tail $DT^{\omega}(d)$ for any given positive d.

Lemma 4. *Assume that $\lambda < \mu$. The tail of user delay in M/M/$1^{[\omega]}$ is given by*

$$DT^{\omega}(d) = e^{-(\mu-\lambda)\omega} DT^{0}(d), d > 0, \tag{6.3}$$

where $DT^{0}(d)$ is the delay tail without proactive serving.

Proof. See Appendix A.8. □

From Lemma 4, we can see that proactive serving can decrease the delay tail exponentially.

Without proactive serving, the delay tail is

$$DT^{0}(d) = e^{-(\mu-\lambda)d}. \tag{6.4}$$

As seen, when boosting the server capacity, the delay tail also decreases exponentially. This suggests that both proactive serving and server capacity boosting are effective in decreasing the delay tail, which is different from what we observed from the average user delay and the delay variance.

Chapter 7

Proactive Serving with Imperfect Prediction

In chapter 4, we analyze the benefit of proactive serving under the assumption that perfect future arrival prediction is available. In this chapter, we look at two more realistic scenarios that correspond to two common types of prediction errors, and study the performance of proactive serving under these settings. For the ease of analysis and illustration, we consider a single server system.

7.1 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 arrival will be out of the system's vision. Therefore, it cannot be served proactively. Intuitively, such errors result in a "side flow" into the system and will affect the ultimate gain one can obtain by proactive serving. The other type of error is false alarm (also called false positive), which happens when the system mistakenly predicts the existence of non-existing arrivals. Such false arrivals will not eventually enter the system for service. However, the system

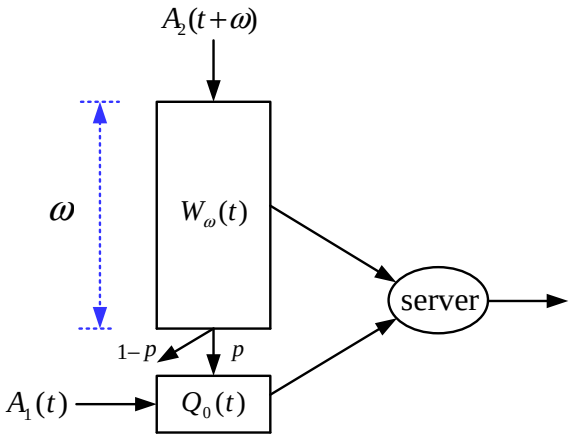

Figure 7.1: A service system with imperfect prediction: The miss detection process $\{A_1(t)\}_t$ can not be served proactively. Requests in $\{A_1(t)\}_t$ enters the system $Q_0(t)$ for service directly. The process $\{A_2(t)\}_t$ includes false alarms and actual arrivals that are predicted correctly. Requests in $\{A_2(t)\}_t$ go through the prediction window $W_\omega(t)$ and can be pre-served by the system. p is the probability that a request in $\{A_2(t)\}_t$ is an actual arrival.

may incorrectly allocate resources to serve them, resulting in wasted service opportunities.

We model the system with these two types of prediction errors by the model shown in Fig. 7.1. In this model, $\{A_2(t)\}_t$ represents the process of predicted arrivals, which include false alarms and actual arrivals that are predicted correctly. $\{A_1(t)\}_t$ instead represents the process of miss detections. $Q_0(t)$ stores requests that have already entered the system and are waiting for service at time t. $W_\omega(t)$ is the prediction window with length 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 $\{A_2(t)\}_t$ will not enter $Q_0(t)$ and thus disappear once they leave the prediction window.

For tractability, we make the following two assumptions on $\{A_1(t)\}_t$ and $\{A_2(t)\}_t$. First, we assume that each request in $\{A_2(t)\}_t$ is independently an

actual request with probability p. In this case, with probability p, a request of $\{A_2(t)\}_t$ will enter $Q_0(t)$ and with probability $1-p$, it will leave the system. The larger p is, the more accurate the prediction mechanism is. Second, we assume that $\{A_1(t)\}_t$ and $\{A_2(t)\}_t$ are independent Poisson processes. Note that our assumptions are not very restrictive. In systems where there are a large number of users, consecutive arrivals can well be from different users. In this case, the prediction about one arriving request can be quite independent from the other.

Denote $E[A_1(t)] = \lambda_1$ and $E[A_2(t)] = \lambda_2$. Since all the miss detections and the actual arrivals among the predicted arrivals compose the real arrivals, we have

$$\lambda_1 + p\lambda_2 = \lambda. \tag{7.1}$$

Here we recall that λ is the average rate for the real arrival process. Equation (7.1) shows the tradeoff between miss detection and false alarm: To predict more actual arrivals or equivalently to reduce the number of miss detections, the prediction mechanism must act more aggressively. However, this can lead to more false alarms. On the other hand, to reduce the number of false alarms, the prediction mechanism needs to act more conservatively, which may result in more miss detections.

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

7.2 Impact of Miss Detections

Now suppose that only miss detections exist in the system, *i.e.*, $p = 1$ and $\lambda_1 + \lambda_2 = \lambda$. Note that in this case, the delay improvement may 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 the delay distribution of a single queue system without proactive serving. After obtaining the distribution of D^ω, we then calculate the average user delay, *i.e.*, $E[D^\omega]$. In this case, miss detection creates a sub-arrival process into $Q_0(t)$ and mixes with the requests in $\{A_2(t)\}_t$. This makes it very challenging to derive the user delay distribution. To resolve this difficulty, we develop a single priority queue system which allows us to study the miss detection process $\{A_1(t)\}_t$ and the predicted arrival process $\{A_2(t)\}_t$ separately. Even so, in the deriving process, we need to apply residue theorem [5] with carefully designed branch cuts [3], which is highly non-trivial. In this way, we obtain the following result.

Theorem 4. *Assume $\lambda = \lambda_1 + \lambda_2 < \mu$. The average user delay under miss detections is given by:*

$$\begin{aligned} & E[D^\omega] \qquad (7.2) \\ = & \frac{\lambda_1}{(\mu-\lambda_1)\lambda} + \frac{\lambda_2}{\lambda}\left[\frac{\lambda^2-\lambda_1\mu}{\lambda_2^2(\mu-\lambda)}\cdot e^{\frac{\lambda_2(\lambda-\mu)}{\lambda}\omega}\cdot \mathbf{1}_{\lambda^2>\lambda_1\mu} + \right. \\ & \left. \frac{1}{2\pi}\int_0^{4\sqrt{\lambda_1\mu}} \frac{(\mu-\lambda)\sqrt{x(4\sqrt{\lambda_1\mu}-x)}e^{(-(\sqrt{\mu}-\sqrt{\lambda_1})^2-x)\omega}}{\left((\sqrt{\mu}-\sqrt{\lambda_1})^2+x\right)^2\cdot\left(\lambda x+(\sqrt{\lambda_1\mu}-\lambda)^2\right)}dx\right], \end{aligned}$$

which decreases exponentially in ω.

Proof. See Appendix A.9. □

The result in (7.2) consists of two components. The first component does not depend on w. This part is due to miss detections. As shown in Fig. 7.1, miss detections enter $Q_0(t)$ directly without going through the prediction window. Thus, proactive serving cannot serve miss detections. In this case, there will be a constant arrival rate entering the queue, resulting in a delay

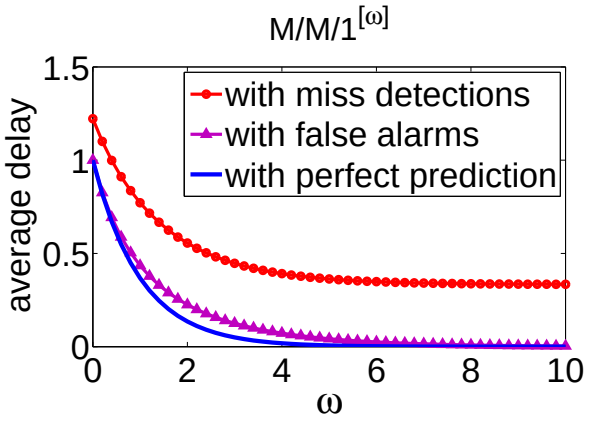

Figure 7.2: The average user delay vs how far we predict the future: The average user delay decreases exponentially when the system predicts further, under both miss detections and false alarms.

that cannot be decreased by proactive serving. The second component of the delay decreases exponentially with ω and diminishes as the system predicts sufficiently far. This is because the system can pre-serve predicted arrivals in $\{A_2(t)\}_t$, which in turn leads to exponential decrease of the second part, demonstrating the power of proactive serving.

Based on (7.2), we plot the average user delay under the impact of miss detections in Fig. 7.2. As seen, user delay decreases exponentially as the system predicts further, and it is finally dominated by the first part of (7.2) as discussed above. Compared to the scenario under perfect prediction, we note that the decay rate in w 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 detailed impact of miss detections in Fig. 7.3 based on (7.2). The average user delay increases when there are more miss detections in the system. At the same time, the increasing rate is enlarged when the number of miss detections increases. Although the gain is not as significant as in the perfect-prediction case, we can see from Fig. 7.2 that proactive serving still provides a large delay improvement.

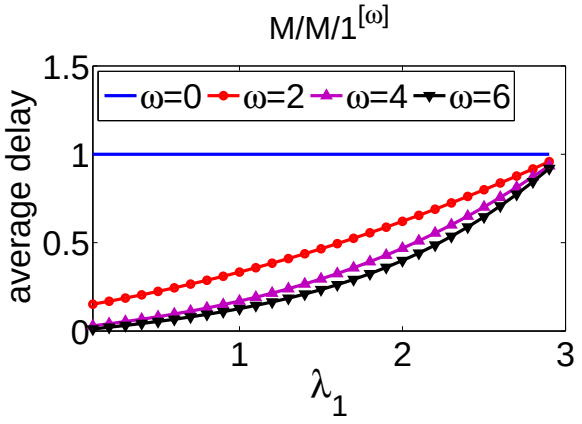

Figure 7.3: Impact of miss detection on the average user delay: The average user delay increases when there are more miss detections. The delay increasing rate is enlarged when the number of miss detections increases.

7.3 Impact of False Alarms

When there exist 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 power of proactive serving will be affected compared to the perfect prediction scenario. Despite this effect, as we will see, proactive serving still provides significant delay improvement.

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 7.2. Hence, the idea used in the miss detection scenario can not be applied here. To resolve 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. Then, by studying the stationary distribution of the Markov chain, we obtain the average user delay by applying Little's Law. Remark that the structure of the

Markov chain is very complicated which makes the analysis highly involved. We obtain the following result.

Theorem 5. *Assume $\lambda = p\lambda_2 < \mu$. The average user delay under the impact of false alarms is given by*

$$E[D^\omega] = \begin{cases} \frac{\mu-\lambda_2}{(\mu-\lambda)^2} \frac{1}{e^{(\mu-\lambda_2)\omega - \frac{\lambda_2-\lambda}{\mu-\lambda}}} & \lambda_2 \neq \mu \\ \frac{1}{(\mu-\lambda)[(\mu-\lambda)\omega+1]} & \lambda_2 = \mu \end{cases}. \tag{7.3}$$

The average delay decreases exponentially in ω when $\lambda_2 \neq \mu$.

Proof. See Appendix A.10. □

From (7.3), it is not immediately clear that $E[D^\omega]$ decreases exponentially in ω when $\lambda_2 \neq \mu$. Instead, we show in the proof that $E[D^\omega]$ can be lower and upper bounded by exponential functions which decrease exponentially in ω.

Based on (7.3), we plot the average user delay under the impact of false alarms in Fig. 7.2. We can see that the average delay decreases exponentially with ω. Compared to the perfect prediction case, the decay rate is smaller. This is because part of the server capacity dedicated to proactive serving is wasted by false alarms. More interestingly, we can derive the following from (7.3):

$$\lim_{\omega\to\infty} E[D^\omega] = \begin{cases} 0 & \text{when } \lambda_2 \leq \mu \\ \frac{\lambda_2-\mu}{(\mu-\lambda)(\lambda_2-\lambda)} & \text{when } \lambda_2 > \mu \end{cases}. \tag{7.4}$$

This equation shows that the system cannot push delay to zero if $\lambda_2 > \mu$. This is because a large fraction of the server capacity used to proactive serving is now consumed by false alarms, and remaining fraction for pre-serving the actual arrivals is not enough to clear the user delay. However, when $\lambda_2 \leq \mu$ the delay can always be decreased to zero as long as the system can predict sufficiently far.

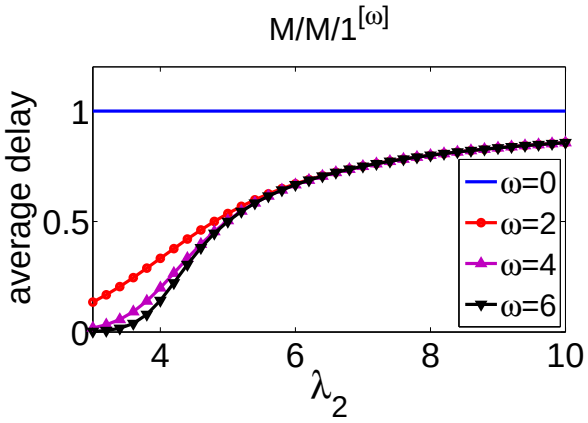

Figure 7.4: Impact of false alarm on the average user delay: The average user delay increases when there are more false alarms. When the average number of false alarms is smaller than the server capacity, the rate of delay increasing is enlarged as the number of false alarms increases. However, when the average number of false alarms is larger than the server capacity, the rate of delay increasing is diminished as the number increases.

In Fig. 7.4, we show the detailed impact of false alarms on user delay based on (7.3). The average user delay increases when more false alarms exist in the system. Different from the miss detection case in Fig. 7.3, when the average number of false alarms is small, the rate of delay increasing is enlarged as the number increases. When the average number of false alarms is large, the rate of delay increasing is gradually diminished as the number increases.

Discussion: Besides the impact on user delay reduction, false alarm may incur additional costs. For example, in content prefetching scenario, serving false alarms will waste the bandwidth resource on the system side and the bandwidth and storage resource on the user side. Thus system designers should take into account the consequences of false alarms when designing the request prediction algorithm. In next section, we will see that the ratio of miss detection and false alarm of the prediction algorithm impacts the

value of user delay. In practice system designers should adjust the ratio of miss detection and false alarm of the prediction algorithm so that user delay experience is satisfactory and the false alarm cost is acceptable.

7.4 Impact of Miss Detections and False Alarms

When miss detections and false alarms are both present in the system, it is very 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 works very aggressively and results in few miss detections but many false alarms. In the second case, the prediction mechanism works very conservatively, which 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 are shown in Fig. 7.5.

A few comments are in order for Fig. 7.5. First of all, the average user delay still decreases exponentially as the system predicts further. Second, when the system acts aggressively to predict future arrivals so that many false alarms occur, the system cannot clear delay totally, which aligns with (7.4). When the system acts conservatively so that many miss detections occur, the delay decay rate is weakened because the server is occupied most of time to deal with miss detections, which also aligns with the results in Fig. 7.3. When the number of miss detections and false alarms are moderate, compared to other two cases, proactive serving decreases user delay rapidly and ensures a small delay for the users.

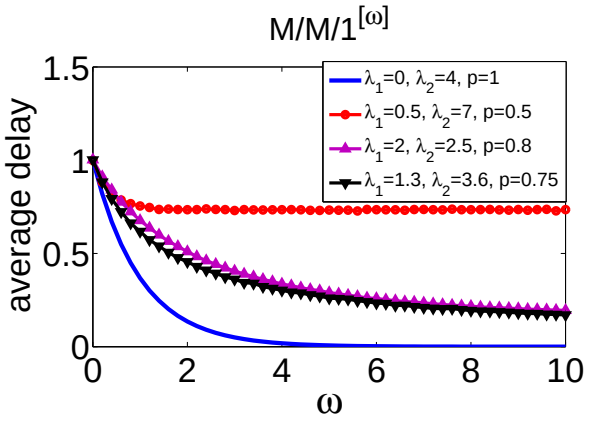

Figure 7.5: Impact of miss detection and false alarm on delay reduction: The curve marked with dot is the result of the case that the system predicts future arrivals aggressively. The curve marked with up-triangle is the result of the case that the system predicts future arrivals conservatively. The curve marked with down-triangle is the result of the cast that the prediction mechanism acts moderately.

Chapter 8

Utility Optimization

For service system designers, it is a critical task to decide how many resources to be deployed in the system based on the designing objective. If the resources are over-provisioned, it will lead to low resource utilization. If the resources are under-provisioned, on the other hand, the quality of service provided by the system will suffer. Both situations can cause loss of revenue. The results we obtained in previous chapters can help to provide guidance to system designers on how many resources should be deployed in the system. In this chapter, we introduce two specific examples under the $M/M/1^{[\omega]}$ model.

8.1 Trade-off Between Benefit and Cost of Proactive Serving

In chapter 4, we show that proactive serving is very powerful in improving user delay experience which can decrease the average user delay exponentially as a function of the prediction window size. The further we explore the future, the more we can decrease the average user delay and improve the user experience. While, on the other hand, the prediction algorithm needs more computation resources as predicting further and thus the prediction

cost increases. For system designers, it is necessary to balance the trade-off between the benefit and cost incurred by future prediction and decide how many resources are dedicated to proactive serving.

To understand how many resources should be dedicated to proactive serving from the optimization point of view, we consider the M/M/$1^{[\omega]}$ system which was studied in section 4.1. Let λ and μ denote the mean request arrival rate and the mean service rate respectively. Recall that for the traditional M/M/1 system without proactive serving, the average user delay is $\frac{1}{\mu-\lambda}$. For the M/M/$1^{[\omega]}$ system, as shown by Theorem 1, the average user delay is $\frac{1}{\mu-\lambda}e^{-(\mu-\lambda)\omega}$ when the system can predict future requests ω time ahead. Thus the delay improvement due to proactive serving is $\frac{1}{\mu-\lambda} - \frac{1}{\mu-\lambda}e^{-(\mu-\lambda)\omega}$. Let the function $U(\frac{1}{\mu-\lambda} - \frac{1}{\mu-\lambda}e^{-(\mu-\lambda)\omega})$ denote the benefit of proactive serving in delay reduction, and the function $C(\omega)$ denote the cost of predicting future requests ω time ahead. Assume that $U(x)$ is an increasing and twice differentiable concave function in variable x and $C(y)$ is an increasing and differentiable convex function in variable y.

Based on the above model, we can formulate the problem that how many resources should be dedicated to proactive serving as follows.

$$\textbf{PS-Only:} \max_{\omega \geq 0} U(\frac{1}{\mu-\lambda} - \frac{1}{\mu-\lambda}e^{-(\mu-\lambda)\omega}) - C(\omega), \tag{8.1}$$

where the objective function measures the net benefit due to proactive serving. The solution of this optimization problem tells how far the system should predict so that the net benefit gets maximized. The solution can be used to guide system designers in designing the request prediction algorithm.

Because $U(x)$ is an increasing and twice differentiable concave function in variable x and $\frac{1}{\mu-\lambda} - \frac{1}{\mu-\lambda}e^{-(\mu-\lambda)\omega}$ is concave in ω, by [12], $U(\frac{1}{\mu-\lambda} - \frac{1}{\mu-\lambda}e^{-(\mu-\lambda)\omega})$ is concave in ω. Consequently, the overall objective function in (8.1) is concave in ω.

Let $U^{'}(\cdot)$ and $C^{'}(\cdot)$ denote the derivative of $U(\cdot)$ and $C(\cdot)$ respectively.

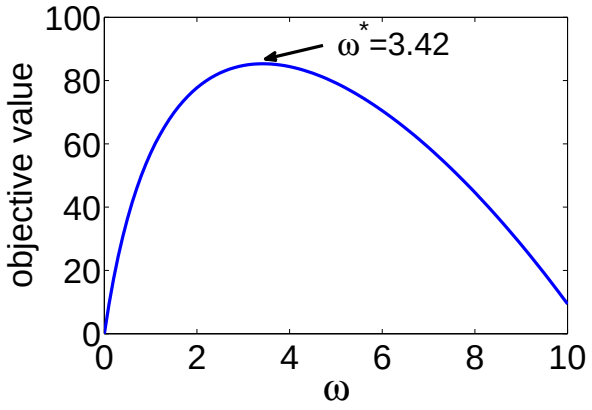

Figure 8.1: An illustrating example of the optimization problem (8.1) where $U(x) = 100 \cdot \ln(x+1)$ and $C(x) = x^2$. The optimal solution is $\omega^* = 3.42$.

Define

$$F(\omega) = U'(\frac{1}{\mu - \lambda} - \frac{1}{\mu - \lambda} e^{-(\mu-\lambda)\omega}) \cdot e^{-(\mu-\lambda)\omega} - C'(\omega). \tag{8.2}$$

The function $F(\omega)$ is the derivative of the objective function in (8.1) in ω. The concave problem (8.1) can be solved by studying $F(\omega)$ in three different cases.

Case 1: When $F(\omega) > 0$ for all $\omega \geq 0$, the objective function in (8.1) is an increasing function of ω, which means that the benefit of proactive serving outweighs the cost. Therefore, the system should predict future arrivals as far as possible.

Case 2: When $F(\omega) < 0$ for all $\omega \geq 0$, the objective function in (8.1) is a decreasing function of ω, which means that the cost of proactive serving outweighs the benefit. Therefore, the system should not predict future arrivals at all.

Case 3: When there exists $\omega^* \geq 0$ such that $F(\omega^*) = 0$, then ω^* is the solution of (8.1). Then, the system should predict ω^* time ahead so that the net utility of the system gets maximized.

As an illustrating example, we choose $U(x) = 100 \cdot \ln(x+1)$ and $C(x) = x^2$.

We plot the value of the objective function in ω in Fig. 8.1. As seen, this example belongs to **Case 3** and the maximum net utility is taken when the system predicts $\omega^* = 3.42$ time units ahead.

8.2 Trade-off Between Proactive Serving and Capacity Boosting

In chapter 5, we show that system designers can utilize both capacity boosting and proactive serving to improve user delay experience. Different combinations of capacity boosting and proactive serving can achieve the same delay performance. However, different combinations of two mechanisms may incur different operation costs. Then a natural question for the system designers is, given the average user delay requirement, how to choose the combination of two mechanisms so that the total cost is minimized?

To understand how to combine capacity boosting and proactive serving to meet the delay requirement from the optimization point of view, we consider the $\mathrm{M/M/1}^{[\omega]}$ system as in section 8.1. Let λ and μ denote the mean request arrival rate and the mean service rate respectively. When the server capacity is boosted by $m-1$ ($m \geq 1$) times, the mean service rate will increase to $m\mu$. Under the boosted server capacity, the average user delay is $\frac{1}{m\mu-\lambda}e^{-(m\mu-\lambda)\omega}$ when the system predicts future requests ω time ahead. Let the function $C_\mu(m)$ denote the cost incurred by boosting the server capacity by $m-1$ times. Let the function $C_\omega(\omega)$ denote the cost incurred by predicting future requests ω time ahead. Assume that $C_\mu(x)$ is an increasing and differentiable convex function in variable x and $C_\omega(y)$ is an increasing and twice differentiable convex function in variable y. Let d be the given average user delay requirement. Assume $d < \frac{1}{\mu-\lambda}$ which says that the requirement is unsatisfied without capacity boosting and proactive serving.

Based on the above model, the problem that how to combine capacity

boosting and proactive serving to meet the delay requirement can be formulated as follows.

$$\textbf{CB-PS:} \min_{\omega \geq 0, m \geq 1} \quad C_\mu(m) + C_\omega(\omega) \tag{8.3}$$

$$\text{s.t.} \quad \frac{1}{m\mu - \lambda} e^{-(m\mu - \lambda)\omega} = d, \tag{8.4}$$

where the objective function represents the total cost introduced by capacity boosting and proactive serving, and the constraint is to satisfy the delay requirement d. The solution of this optimization problem tells how far the system should predict and how many times the system capacity should be increased to satisfy the delay requirement while at the same time minimizing the total cost. The solution can be used to guide system designers in the request prediction algorithm design and server provision of the system.

Under the constraint in (8.4), the variable ω can be expressed as a function of m as follows

$$\omega = -\frac{1}{m\mu - \lambda} \ln\left[(m\mu - \lambda)d\right], \tag{8.5}$$

which is convex in m because its second derivative in m is positive. Since C_ω is an increasing and twice differentiable convex function, by [12], $C_\omega(\omega)$ is convex in m. Consequently, the overall objective function in (8.3) is convex in m.

Let $C'_\mu(\cdot)$ and $C'_\omega(\cdot)$ denote the derivative of $C_\mu(\cdot)$ and $C_\omega(\cdot)$ respectively. Let m^* and ω^* be the solution of (8.3). Define

$$F(m) = C'_\mu(m) + C'_\omega(-\frac{1}{m\mu - \lambda} \ln\left[(m\mu - \lambda)d\right]) \cdot \frac{(\ln\left[(m\mu - \lambda)d\right] - 1)\mu}{(m\mu - \lambda)^2}. \tag{8.6}$$

Since the objective function in (8.3) is convex in m, we can find m^* and then ω^* by studying its derivative $F(m)$ in three different cases.

Case 1: $F(m) > 0$ for all $m \geq 1$. Then the objective function in (8.3) is an increasing function of m. This means that, under the delay requirement, the cost of pure proactive serving is lower than that of any combination of

proactive serving and capacity boosting. As a result, the solution under this case is $m^* = 1$ and $\omega^* = -\frac{1}{\mu-\lambda}\ln[(\mu-\lambda)d]$, which indicates that the system should solely rely on proactive serving to satisfy the delay requirement.

Case 2: $F(m) < 0$ for all $m \geq 1$. Then the objective function in (8.3) is a decreasing function of m. This means that, under the delay requirement, the cost of pure capacity boosting is lower than that of any combination of proactive serving and capacity boosting. As a result, the solution under this case is $m^* = \frac{1}{\mu d} + \frac{\lambda}{\mu}$ and $\omega^* = 0$, which indicates that the system should solely rely on capacity boosting to satisfy the delay requirement.

Case 3: There exists $\bar{m} \geq 1$ such that $F(\bar{m}) = 0$. If $\bar{m} > \frac{1}{\mu d} + \frac{\lambda}{\mu}$, then $m^* = \frac{1}{\mu d} + \frac{\lambda}{\mu}$ and $\omega^* = 0$. Otherwise, $m^* = \bar{m}$ and $\omega^* = -\frac{1}{m^*\mu-\lambda}\ln[(m^*\mu-\lambda)d]$. Different from **Case 1** and **Case 2**, the cost of combination of proactive serving and capacity boosting is lower than that of either pure approach. This indicates that the system needs both proactive serving and capacity boosting to satisfy the delay requirement under this case.

As an illustrating example, let $\mu = 1$, $C_\mu(m) = (m-1)^2$ and $C_\omega(\omega) = \omega^2$. Define $d_0 = \frac{1}{\mu-\lambda}$. We summarize the solutions of the optimization problem (8.3) under different average request arriving rates λ and average user delay requirements d in Tab 8.1. In Fig. 8.2, 8.3, 8.4, we show the derivative of the objective function in ω and m respectively when $d = \frac{1}{10}d_0$ under different λ. Two variables ω and m are correlated by the constraint in (8.3). When calculating the derivative in ω, we consider m as a function of ω. Similarly, when calculating the derivative in m, we consider ω as a function of m. From the table and figures, we can obtain the following insights.

- Under different workload levels (corresponding to different values of $\frac{\lambda}{\mu}$), to achieve the relatively similar delay target, system designers should use different combinations of capacity boosting and proactive serving so that the total cost gets minimized.

- When workload level is low (small $\frac{\lambda}{\mu}$), as shown by Fig. 8.2, the deriva-

	$d = \frac{1}{2}d_0$	$d = \frac{1}{10}d_0$	$d = \frac{1}{100}d_0$
$\lambda = 0.1$	$m^* = 1.31$, $\omega^* = 0.33$	$m^* = 1.84$, $\omega^* = 0.95$	$m^* = 2.40$, $\omega^* = 1.60$
$\lambda = 0.5$	$m^* = 1.34$, $\omega^* = 0.21$	$m^* = 1.94$, $\omega^* = 0.87$	$m^* = 2.55$, $\omega^* = 1.56$
$\lambda = 0.9$	$m^* = 1.1$, $\omega^* = 0$	$m^* = 1.68$, $\omega^* = 0.33$	$m^* = 2.45$, $\omega^* = 1.21$

Table 8.1: Solutions (m^*, ω^*) of (8.3) under different average request arriving rates λ and average user delay requirements d. $d_0 = \frac{1}{\mu-\lambda}$ and $\mu = 1$.

tive value of the objective function in ω when ω is small is much smaller than that in m. So under the delay requirement, increasing ω can lead to more cost reduction. This suggests that system designers should rely more on proactive serving than capacity boosting to guarantee the delay performance while minimizing the operation costs. When workload level is high (large $\frac{\lambda}{\mu}$), in contrast, as shown by Fig. 8.4, the derivative value of the objective function in m when m is small is much smaller than that in ω. This suggests that system designers should rely more on capacity boosting than proactive serving. From Tab. 8.1, we can also get the same observation. The reason behind the observation is quite intuitive: When the workload level is high, the server will be busy most of time with serving requests that have been waiting in the system. Thus the probability that a request get pre-served is small. As a result, the delay improvement by proactive serving is small. In contrast, when the workload level is low, the delay improvement by proactive serving is prominent because the server proactively serves future requests most of time. Therefore, when the workload level is high, capacity boosting is more effective than proactive serving in decreasing request delay. When the workload level is low, proactive serving is more effective than capacity boosting in decreasing request delay.

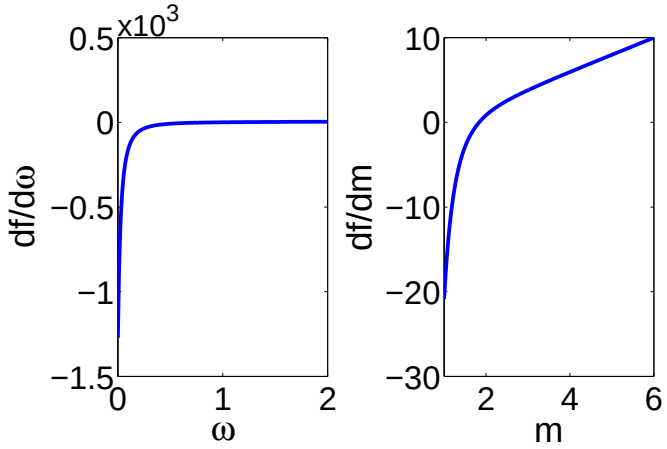

Figure 8.2: The left plot is the derivative of the objective function in ω. The right plot is the derivative of the objective function in m. The delay requirement $d = \frac{1}{10}d_0$. The request arriving rate $\lambda = 0.1$.

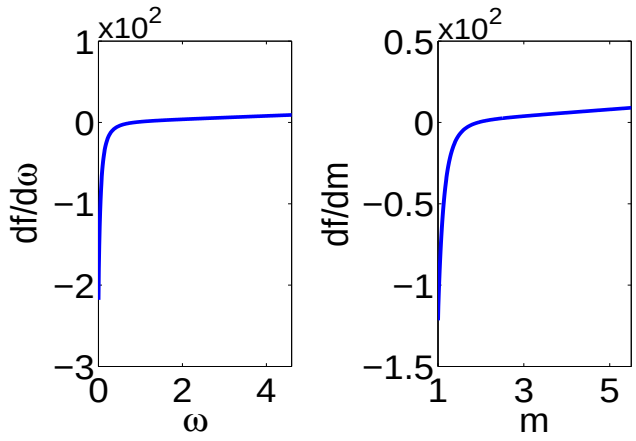

Figure 8.3: The left plot is the derivative of the objective function in ω. The right plot is the derivative of the objective function in m. The delay requirement $d = \frac{1}{10}d_0$. The request arriving rate $\lambda = 0.5$.

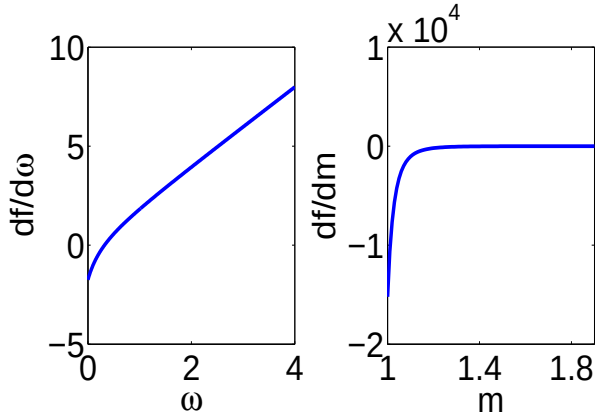

Figure 8.4: The left plot is the derivative of the objective function in ω. The right plot is the derivative of the objective function in m. The delay requirement $d = \frac{1}{10}d_0$. The request arriving rate $\lambda = 0.9$.

Chapter 9

Delay Reduction for M/M/$k^{[\omega]}$ and Markovian/Geo/$1^{[\omega]}$ under Proactive Serving

In this chapter, we provide results about user delay for two queuing models: M/M/$k^{[\omega]}$, and Markovian/Geo/$1^{[\omega]}$. M/M/$k^{[\omega]}$ extends M/M/$1^{[\omega]}$ from single server to multiple servers. Markovian/Geo/$1^{[\omega]}$ can be used to model flash crowd arrivals in practical systems [13].

9.1 M/M/k$^{[\omega]}$

In an M/M/$k^{[\omega]}$ system, there are k identical servers. 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 future arrivals ω time ahead and serve them proactively. A user request can be served by any server. A server is dedicated to one request at any given time. Such model can be used to model task management systems of smartphones with multi-core processor where each core represents a server to satisfy application

requests.

9.1.1 Average User Delay

Recall that $\rho = \frac{\lambda}{\mu}$. Define

$$q_0 = 1 - \frac{\rho^k}{\rho^k + k!(1-\frac{\rho}{k})\sum_{i=0}^{k-1}\rho^i\frac{1}{i!}},$$

which is the steady-state probability that the queue is empty in the M/M/k system. Following the same procedure as we develop Theorem 1, we obtain the average user delay of M/M/$k^{[\omega]}$ as follows.

Lemma 5. *Assume $\rho = \lambda/\mu < k$. The average user delay of M/M/$k^{[\omega]}$ with perfect prediction is given by*

$$E\left[D^{\omega}\right] = \begin{cases} \left(\frac{2-q_0}{\mu} + (1-q_0)\omega\right)e^{-\mu\omega}, & \text{if } \rho = k-1; \\ \eta_1 \cdot e^{-\mu\omega} + \eta_2 \cdot e^{-\mu(k-\rho)\omega}, & \text{if } 0 \le \rho < k \\ & \text{and } \rho \neq k-1, \end{cases} \tag{9.1}$$

where $\eta_1 = \frac{\rho+q_0-k}{\mu[\rho+1-k]}$ and $\eta_2 = \frac{1-q_0}{\mu(k-\rho)[\rho+1-k]}$.

Proof. See Appendix A.11. □

It is straightforward to verify that the expression of $E\left[D^{\omega}\right]$ for $\omega = 0$ reduces to the one for the classical M/M/k system [8].

We observe that $E\left[D^{\omega}\right]$ in (9.1) has different closed-form expressions, depending on the relationship between ρ and k. This is due to the fact that the probability distribution of D^0 (user delay in M/M/k) is case-dependent [8], and consequently the distribution of $D^{\omega} = \max\left(0, D^0 - \omega\right)$ and its expectation are also case-dependent.

From (9.1), it is not immediately clear that $E\left[D^{\omega}\right]$ decays exponentially in ω. For example, when $\rho < k-1$, $\eta_1 > 0$ and $\eta_2 < 0$. The sum of two exponential functions with different signs may be increasing and thus may

not be an exponential function. In the following lemma, we show that $E[D^\omega]$ indeed decreases exponentially in ω by deriving upper and lower bounds of $E[D^\omega]$ which decrease exponentially in ω.

Lemma 6. *Define* $\eta_3 = \frac{1+k-q_0-\rho}{(k-\rho)\mu}$.

- *When* $0 \leq \rho < k-1$, *we have* $\eta_1 > 0$, $\eta_3 > 0$, *and*

$$\eta_3 \cdot e^{-\mu\omega} < E[D^\omega] < \eta_1 \cdot e^{-\mu\omega}.$$

- *When* $\rho = k-1$, *there exists* $0 < \epsilon < \frac{e-1}{e}$ *such that*

$$\frac{2-q_0}{\mu} \cdot e^{-\mu\omega} < E[D^\omega] < \frac{3-2q_0}{\mu} \cdot e^{-\epsilon\mu\omega}.$$

- *When* $k-1 < \rho < k-q_0$, *we have* $\eta_2 > 0$, $\eta_3 > 0$, *and*

$$\eta_3 \cdot e^{-\mu(k-\rho)\omega} < E[D^\omega] < \eta_2 \cdot e^{-\mu(k-\rho)\omega}.$$

- *When* $k-q_0 \leq \rho < k$, *we have* $\eta_1 > 0$, $\eta_2 > 0$, *and*

$$(\eta_1+\eta_2) \cdot e^{-\mu\omega} < E[D^\omega] < (\eta_1+\eta_2) \cdot e^{-\mu(k-\rho)\omega}.$$

Proof. See Appendix A.12. □

We plot the average user delay in (9.1) as a function of ω for M/M/$5^{[\omega]}$ system under different workload levels (corresponding to different values of ρ/k) in Fig. 9.1. The results confirm that the average user delay decreases exponentially in the prediction window size ω.

9.1.2 Variance of User Delay

Similar to obtain Lemma 5, for the M/M/$k^{[\omega]}$ system, we have the following lemma for the delay variance $Var[D^\omega]$.

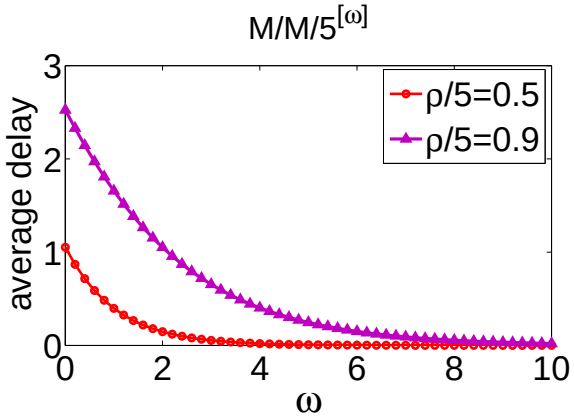

Figure 9.1: The average user delay vs how far we predict the future under perfect prediction: Under different workload levels, the average user delay decreases exponentially as the system predicts further.

Lemma 7. *Assume that* $\rho = \frac{\lambda}{\mu} < k$. *The variance of user delay in* $M/M/k^{[\omega]}$ *is given by*

$$\begin{aligned} & Var[D^{\omega}] \\ = \; & \frac{e^{-\mu\omega}}{\mu}\left[\frac{6-4q_0}{\mu} - \frac{(2-q_0)^2}{\mu}e^{-\mu\omega} - (2-q_0)(1-q_0)\omega e^{-\mu\omega}\right] + \\ & \omega e^{-\mu\omega}\left[\frac{2(1-q_0)}{\mu} - \frac{(2-q_0)(1-q_0)}{\mu}e^{-\mu\omega} - (1-q_0)^2\omega e^{-\mu\omega}\right] \end{aligned}$$

when $\rho = k-1$;

$$\begin{aligned} & Var[D^{\omega}] \\ = \; & \eta_1 e^{-\mu\omega}\left[\frac{2}{\mu} - \eta_1 e^{-\mu\omega} - \eta_2 e^{-\mu(k-\rho)\omega}\right] + \\ & \eta_2 e^{-\mu(k-\rho)\omega}\left[\frac{2}{\mu(k-\rho)} - \eta_1 e^{-\mu\omega} - \eta_2 e^{-\mu(k-\rho)\omega}\right] \end{aligned}$$

when $0 \le \rho < k$ *and* $\rho \neq k-1$. *And* $Var[D^{\omega}]$ *decreases exponentially in* ω.

Proof. See Appendix A.13. □

The lemma shows that proactive serving decreases the delay variance exponentially as we observed in Lemma 3 for the M/M/$1^{[\omega]}$ system. The reason

why the delay variance has different closed-form expressions depending on the relationship between ρ and k is the same as that for the average user delay.

9.1.3 Tail of User Delay

Similar to obtain Lemma 5, for the M/M/$k^{[\omega]}$ system, we have the following lemma for the delay tail $DT^{\omega}(d)$.

Lemma 8. *Assume that $\rho = \frac{\lambda}{\mu} < k$. The tail of user delay in M/M/$k^{[\omega]}$ is given by*

$$DT^{\omega}(d) = \begin{cases} q_0 e^{-\mu(\omega+d)} + (1-q_0)(\mu d + \mu\omega + 1)e^{-\mu(\omega+d)} & \rho = k-1 \\ \eta_1 \mu e^{-\mu(\omega+d)} + \eta_2 \mu (k-\rho) e^{-\mu(k-\rho)(\omega+d)} & 0 \le \rho < k \text{ and } \rho \neq k-1 \end{cases},$$

which decreases exponentially in ω.

Proof. See Appendix A.14. □

The lemma shows that proactive serving decreases the delay tail exponentially as we observed in Lemma 4 for the M/M/$1^{[\omega]}$ system. Due to the same reasons, the delay tail has different closed-form expressions as the average user delay and the delay variance.

9.1.4 Comparison With Capacity Boosting

For the M/M/k system without proactive serving, the average user delay with service capacity m is given by

$$\frac{1 - q_0^{(m)}}{\mu(mk - \rho)} + \frac{1}{m\mu}$$

where $q_0^{(m)} = 1 - \frac{(\frac{\rho}{m})^k}{(\frac{\rho}{m})^k + k!(1-\frac{\rho}{mk})\sum_{i=0}^{k-1}(\frac{\rho}{m})^i \frac{1}{i!}}$. To achieve the same delay reduction as boosting the capacity by m times, based on Lemma 6, we have the following theorem for $\omega^*(m)$.

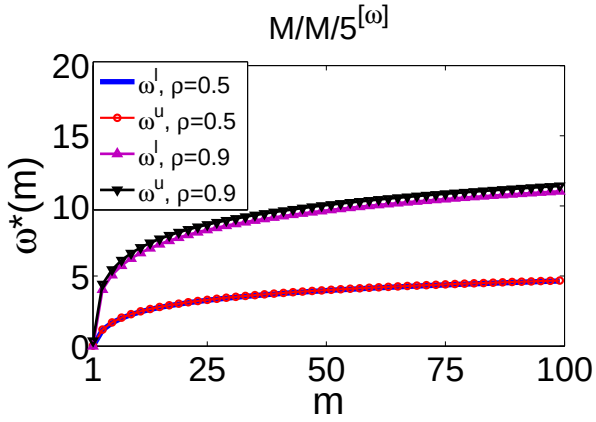

Figure 9.2: How far do we need predict in order to achieve the same delay performance as compared to the server capacity being increased by m times in the M/M/$k^{[\omega]}$ system?

Lemma 9. *Assume* $\rho = \frac{\lambda}{\mu} < k$. *Define* $\Delta = \frac{1-q_0^{(m)}}{\mu(mk-\rho)} + \frac{1}{m\mu}$. *For the M/M/*$k^{[\omega]}$ *system with perfect prediction,* $\omega^*(m)$ $(m \geq 1)$ *is given by*

$$\omega_1^l < \omega^*(m) < \omega_1^u$$

where $\omega_1^l = -\frac{1}{\mu}\ln\left(\frac{\triangle}{\eta_3}\right)$ *and* $\omega_1^u = -\frac{1}{\mu}\ln\left(\frac{\Delta}{\eta_1}\right)$ *when* $0 \leq \rho < k-1$;

$$\omega_2^l < \omega^*(m) < \omega_2^u$$

where $\omega_2^l = -\frac{1}{\mu}\ln\left(\frac{\mu\triangle}{2-q_0}\right)$ *and* $\omega_2^u = -\frac{1}{\epsilon\mu}\ln\left(\frac{\mu\Delta}{3-2q_0}\right)$ *when* $\rho = k-1$;

$$\omega_3^l < \omega^*(m) < \omega_3^u$$

where $\omega_3^l = -\frac{1}{\mu(k-\rho)}\ln\left(\frac{\triangle}{\eta_3}\right)$ *and* $\omega_3^u = -\frac{1}{\mu(k-\rho)}\ln\left(\frac{\Delta}{\eta_2}\right)$ *when* $k-1 < \rho < k-q_0$;

$$\omega_4^l < \omega^*(m) < \omega_4^u$$

where $\omega_4^l = -\frac{1}{\mu}\ln\left(\frac{\triangle}{\eta_1+\eta_2}\right)$ *and* $\omega_4^u = -\frac{1}{\mu(k-\rho)}\ln\left(\frac{\Delta}{\eta_1+\eta_2}\right)$ *when* $k-q_0 \leq \rho < k$.

Proof. See Appendix A.15. □

We plot the $\omega^*(m)$ as a function of m in Fig. 9.2 under different workload levels (represented by different values of ρ).

Discussion: The above results we get show that the effect of proactive serving on the user delay in M/M/$k^{[\omega]}$ system is similar to that in M/M/$1^{[\omega]}$ system. This observation suggests that the delay-reduction benefit of proactive serving is robust to resource pooling (*i.e.*, a system makes a collection of resources behave like a single pooled resource).

9.2 Markovian/Geo/$1^{[\omega]}$

In this section, we investigate the delay-reduction benefit of proactive serving for a discrete-time Markovian/Geo/$1^{[\omega]}$ queuing system with one server. The time is slotted, the service process is a Bernoulli process with parameter μ. In every time slot, with probability μ, the single server can serve a request in the system. The arrival process $\{A(t)\}_t$ is a two-state Markovian process with mean arrival rate $\lambda = \frac{\alpha}{\alpha+\beta}$, as shown in Fig. 9.3. We assume that user request arrives at the system at the end of each time slot. For the Markovian process, when α and β are small, user requests arrive in a bursty manner. Such process can be used to model flash crowd scenarios where a large number of user requests arrive at the system in a small time window. Flash crowds are common in practice. For example, in video on demand systems, views of TV show videos usually skyrocket once they get released and then decrease rapidly after a few days. Flash crowd creates bursty arrivals and is difficult to serve with satisfactory quality of service in practice. To accommodate bursty arrivals, a common exercise is to provision enough servers according to the peak arrival rate. Due to the highly-irregular arrival pattern, servers remain idle most of time after serving bursty requests, which leads to substantial resource and energy wastage.

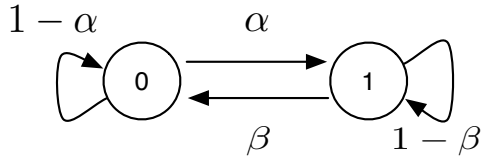

Figure 9.3: On-Off Markovian Arrival: The state of the Markov chain is $A(t)$. $A(t) = 1$ means that there is one request arrival at slot t. $A(t) = 0$ means that there is no request arriving at the system. The mean arrival rate is $\lambda = \frac{\alpha}{\alpha+\beta}$.

9.2.1 Average User Delay

Similar to the analysis for M/M/$1^{[\omega]}$ system, we first obtain the user delay distribution of Markovian/Geo/1 system, then "shift" it to obtain the user delay distribution of Markovian/Geo/$1^{[\omega]}$ system, and finally compute the average user delay as shown in the following theorem.

Lemma 10. *The average user delay of discrete-time Markovian/Geo/$1^{[\omega]}$ system with perfect prediction is given by*

$$E\left[D^{\omega}\right] = \frac{1-\mu}{(\alpha+\beta)(\mu-\lambda)}\left[\frac{1-\mu}{(1-\mu)+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega-1}, \tag{9.2}$$

where $\lambda = \frac{\alpha}{\alpha+\beta} < \mu$ and α, β, and μ are all in $(0, 1)$.

Proof. See Appendix A.16. □

From (9.2), we see that the average user delay decreases exponentially in the prediction window size ω, as also demonstrated in Fig. 9.4 for different mean arrival rates (corresponding to $\lambda = \alpha/(\alpha+\beta)$ and different degrees of bursty arrivals (corresponding to different combinations of α and β).

9.2.2 Variance of User Delay

Similar to obtain Lemma 10, for the Markovian/Geo/$1^{[\omega]}$ system, we have the following lemma for the delay variance $Var[D^{\omega}]$.

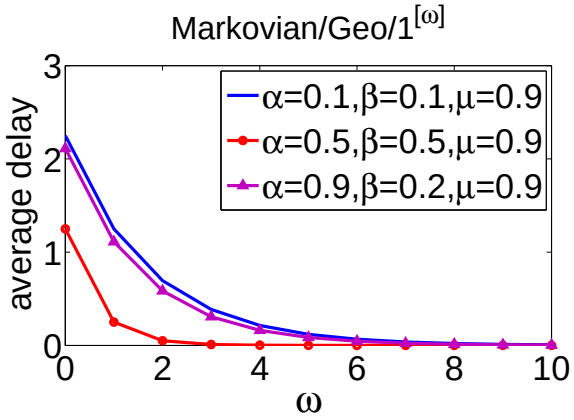

Figure 9.4: The average user delay vs how far we predict the future under perfect prediction: Under different workload levels, the average user delay decreases exponentially as the system predicts further.

Lemma 11. *Assume that* $\lambda = \frac{\alpha}{\alpha+\beta} < \mu$ *and* α, β *and* μ *are all in* $(0, 1)$. *The variance of user delay in Markovian/Geo/*$1^{[\omega]}$ *is given by*

$$\begin{aligned} & Var[D^{\omega}] \\ = \; & Var[D^{0}]\left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega}\left(\frac{2(1-\mu)+(\alpha+\beta)(\mu-\lambda)}{1-\mu}\right. \\ & \left.-\frac{1-\mu+(\alpha+\beta)(\mu-\lambda)}{1-\mu}\left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega}\right), \end{aligned}$$

which decreases exponentially in ω.

Proof. See Appendix A.17. □

The lemma shows that proactive serving decreases the delay variance exponentially.

As shown in the proof, we have

$$Var[D^{\omega}] \leq Var[D^{0}]\frac{2(1-\mu)+(\alpha+\beta)(\mu-\lambda)}{1-\mu}\left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega} \tag{9.3}$$

and

$$Var[D^{\omega}] \geq Var[D^{0}]\left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega}. \tag{9.4}$$

As seen, the decaying rate of the delay variance is dominated by

$$\ln\left[(1-\mu)/\left(1-\mu+(\alpha+\beta)(\mu-\lambda)\right)\right] \tag{9.5}$$

which is the same as that of the average user delay.

9.2.3 Tail of User Delay

Similar to obtain Lemma 10, for the Markovian/Geo/$1^{[\omega]}$ system, we have the following lemma for the delay tail $DT^{\omega}(d)$.

Lemma 12. *Assume that* $\lambda = \frac{\alpha}{\alpha+\beta} < \mu$ *and* α, β *and* μ *are all in* $(0, 1)$. *The tail of user delay in Markovian/Geo/*$1^{[\omega]}$ *is given by*

$$DT^{\omega}(d) = \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega} DT^{0}(d),$$

which decreases exponentially in ω.

Proof. See Appendix A.18. □

The lemma shows that proactive serving decreases the delay tail exponentially.

As seen, the decaying rate of the delay tail is

$$\ln\left[(1-\mu)/\left(1-\mu+(\alpha+\beta)(\mu-\lambda)\right)\right] \tag{9.6}$$

which is the same as that of the average user delay and the delay variance.

9.2.4 Comparison With Capacity Boosting

For the Markovian/Geo/1 system without proactive serving, the average user delay with service capacity m is given by

$$\frac{1-m\mu}{(\alpha+\beta)(m\mu-\lambda)}+1.$$

To achieve the same delay reduction as boosting the capacity by m times, based on (9.2), we have the following theorem for $\omega^{*}(m)$.

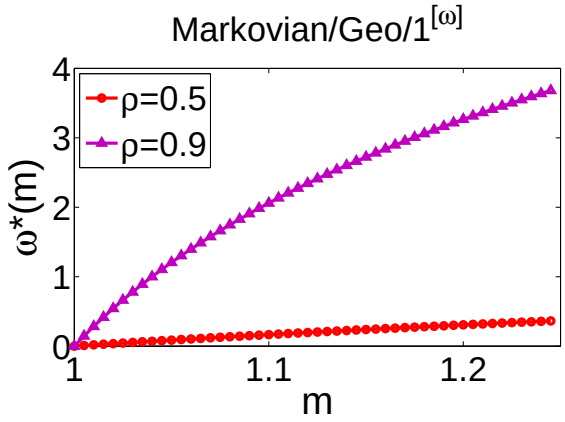

Figure 9.5: How far do we need predict in order to achieve the same delay performance as compared to the server capacity being increased by m times in the Markovian/Geo/$1^{[\omega]}$ system?

Lemma 13. *Assume* $\lambda = \frac{\alpha}{\alpha+\beta} < \mu$. *For the Markovian/Geo/*$1^{[\omega]}$ *system with perfect prediction,* $\omega^*(m)$ $(1 \leq m < \frac{1}{\mu})$ *is given by*

$$\omega^*(m) = \frac{\ln\left(\frac{(\alpha+\beta)(\mu-\lambda)}{1-\mu}\left[\frac{1-m\mu}{(\alpha+\beta)(m\mu-\lambda)}+1\right]\right)}{\ln\left(\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right)}+1.$$

Proof. See Appendix A.15. □

We plot the $\omega^*(m)$ as a function of m in Fig. 9.5 under different workload levels (represented by different values of ρ).

Discussion: Unlike the results of M/M/$1^{[\omega]}$ and M/M/$k^{[\omega]}$ systems, the effect of proactive serving on the user delay of Markovian/Geo/$1^{[\omega]}$ system depends not only on the mean arrival rate λ, but also on α and β, *i.e.*, the two transition probabilities that control the arrival burstiness. Specifically, the decaying rate in ω in Lemma 10, 11 and 12 is $\ln\left[(1-\mu)/\left(1-\mu+(\alpha+\beta)(\mu-\lambda)\right)\right]$. For the same μ and λ, the decaying rate is small when α and β are small, and is large with large α and β. These results match our intuitions that it is difficult to decrease user delay via proactive serving for bursty arrivals (such as flash crowd).

Chapter 10

Simulations

We carry out simulations to study the impact of proactive serving on decreasing 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? (ii) How does prediction error impact the effect of proactive serving? (iii) How about comparing with system capacity boosting?

10.1 Parameters and Settings

In our simulations, we consider three different data traces. Under each data trace, the simulation settings are as follows.

Synthetic Data Trace: We generate the data based on different distributions. We consider four different settings. In the first case, the request inter-arrival time follows Exponential distribution and the request service time follows Uniform distribution. In the second case, the request inter-arrival time follows Uniform distribution and the request service time follows Uniform distribution. In the third case, the request inter-arrival time follows Exponential distribution and the request service time follows Weibull distribution. In the fourth case, the request inter-arrival time follows Uniform

distribution and the request service time follows Weibull distribution. Under the four settings, the mean arrival rate is $\lambda = 0.4$ and the mean service rate is $\mu = 0.5$. For the Uniform distribution, boundaries are 0 and double mean. For the Weibull distribution, the scale parameter is equal to 1. There is a single server in the system under four settings. The server adopts FCFS as its service policy.

***Youku* Data Trace**: We collected the data of a weekly TV show video [1] from *Youku* [6]. Fig. 10.1 shows how the view number changes since its first release. As seen, on the first day after its release, the view number skyrockets. Then the view number decreases sharply and maintains relatively low afterwards. Such view request process is a typical bursty arrival process.

For ease of the simulation, we set the number of servers to be 2500 in the system. In one second, a single server can serve a video of 100MB which is about 10% of the observed video. Such setting is chosen to make sure that the average user delay is at reasonable level. To better model practical situation, the system adopts the Processor-Sharing (PS) service policy where each request is served by all the servers simultaneously.

***Youtube* Data Trace**: We collected the data from *YouTube* [7], which covers 1000 different videos and their view numbers during one week. The average video duration are 456 seconds. The view process of all videos is shown in Fig. 10.2. Compared to the request arrival process in Fig. 10.1, this process is more uniform.

In the simulation, we set the number of servers to be 1000 in the system. In one second, a single server can serve a video of 700 seconds which is about 100MB. Such setting is chosen to make sure that the average user delay is at reasonable level. As in the setting under the *Youku* trace, the system adopts the Processor-Sharing (PS) service policy.

Common Setting: To investigate the impact of prediction errors, we model the miss detections and false alarms in the simulation as follows. Each

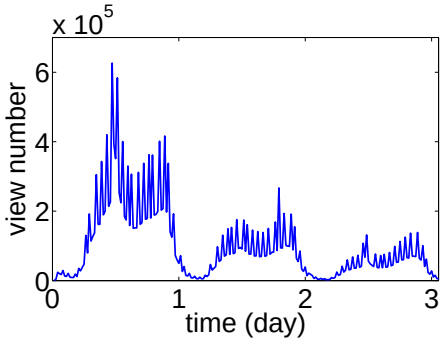

Figure 10.1: The number of views of a weekly TV show video in *Youku* during 3 days since its first release.

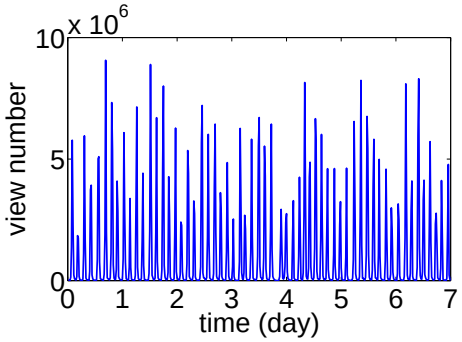

Figure 10.2: The number of views of 1000 different videos in *YouTube* during one week.

time unit, q fraction of the request arrivals are miss detections, and $\frac{1-p}{p}$ fraction of the rest request arrivals are false alarms. Therefore, fraction $\frac{1-q}{p}$ of all requests are predicted arrivals, which can be served proactively. Larger q means more miss detections, and smaller p means more false alarms in the system. For perfect prediction, $q = 0$ and $p = 1$.

10.2 Delay Reduction by Proactive Serving

Under synthetic data trace, the simulation results are shown in Fig. 10.3, 10.4, 10.5 and 10.6 respectively. We can see that the average user request delay decays significantly in ω, which is the same as what we observed in chapter 4. Under the *YouTube* data trace, the simulation results are shown in Fig. 10.7. As seen, under perfect prediction, the user request delay can be decreased by 50% when the system only predicts about 100 seconds ahead. Under the *Youku* data trace, the simulation results are shown in Fig. 10.8. When the system can predict perfectly about 12 mins ahead, the user request delay is decreased by 50%. The above simulation results all suggest that the system can improve user delay experience significantly by proactive serving.

The reason why the system needs to predict further under the *Youku*

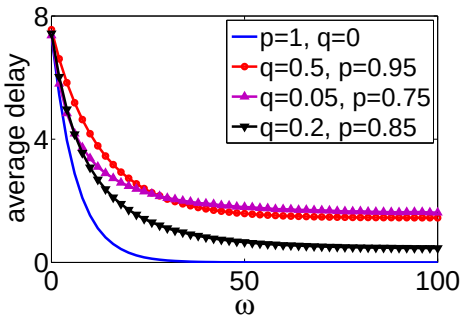

Figure 10.3: The average user request delay vs how far we predict the future when the request inter-arrival time follows Exponential distribution and the request service time follows Uniform distribution.

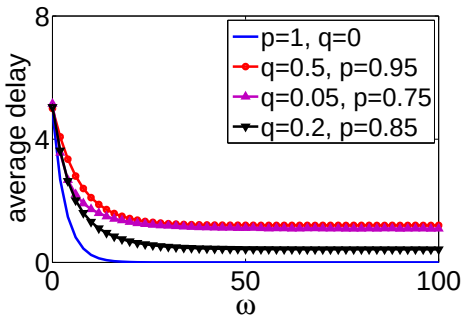

Figure 10.4: The average user request delay vs how far we predict the future when the request inter-arrival time follows Uniform distribution and the request service time follows Uniform distribution.

data trace compared to the *YouTube* data trace is due to the structure of bursty arrivals. In bursty arrivals, when flash crowd arrives, servers are busy most of time and hardly have spare capacity to server future requests. When flash crowd leaves, few future requests can be pre-served because there is almost no request coming. So, proactive serving is less effective under bursty arrivals. This aligns with the results in section 9.2.

In Fig. 10.9, 10.10, 10.11, 10.12, 10.13 and 10.14, we show the delay distributions without prediction and with perfect prediction. As seen from the figures, the delay distribution under proactive serving can be obtained by shifting the delay distribution without prediction ω units left, which aligns with Lemma 2 and Corollary 1 in section 4.1.

10.3 Impact of Miss Detections and False Alarms

The simulation results are shown in Fig. 10.3, 10.4, 10.5, 10.6, 10.7 and 10.8. As seen, both miss detection and false alarm weaken the effect of proactive

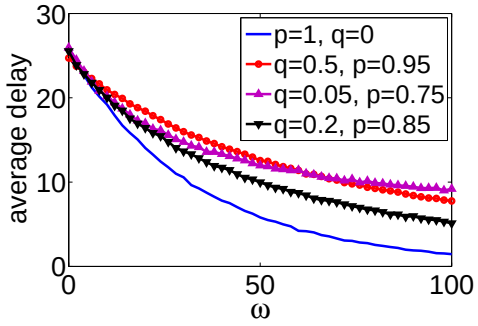

Figure 10.5: The average user request delay vs how far we predict the future when the request inter-arrival time follows Exponential distribution and the request service time follows Weibull distribution.

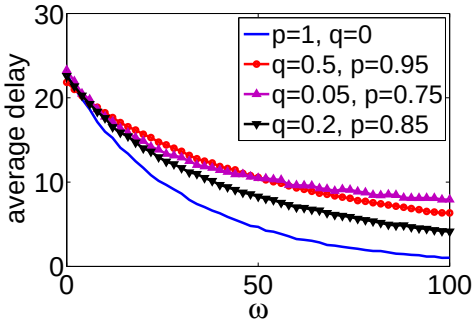

Figure 10.6: The average user request delay vs how far we predict the future when the request inter-arrival time follows Uniform distribution and the request service time follows Weibull distribution.

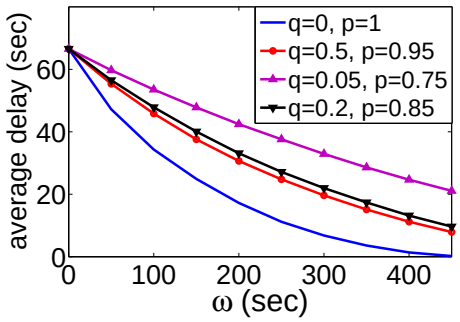

Figure 10.7: The average user request delay vs how far we predict the future under the *YouTube* data trace.

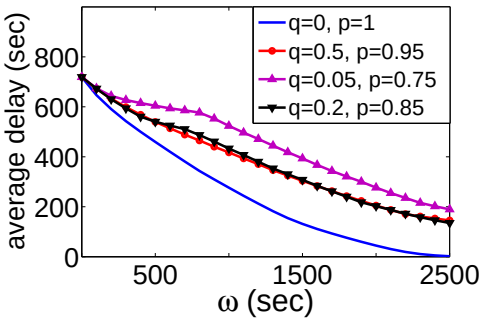

Figure 10.8: The average user request delay vs how far we predict the future under the *Youku* data trace.

serving on delay reduction. But proactive serving can still decrease user delay significantly under the impact of prediction errors.

Under all the data traces, proactive serving is more sensitive to false alarms than miss detections, which matches what we observed in section 7.4. When there are few miss detections and many false alarms, the effect

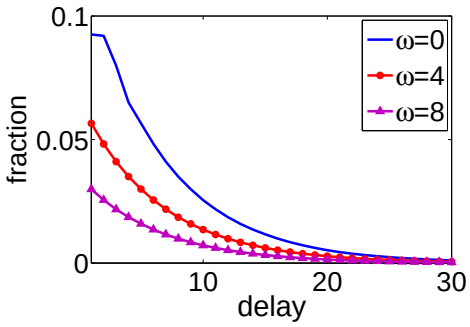

Figure 10.9: Delay distributions when the request inter-arrival time follows Exponential distribution and the request service time follows Uniform distribution.

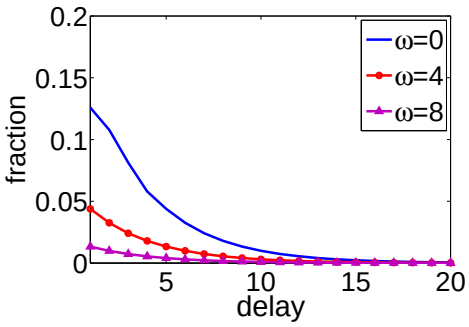

Figure 10.10: Delay distributions when the request inter-arrival time follows Uniform distribution and the request service time follows Uniform distribution.

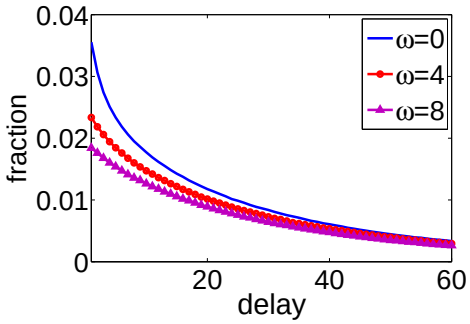

Figure 10.11: Delay distributions when the request inter-arrival time follows Exponential distribution and the request service time follows Weibull distribution.

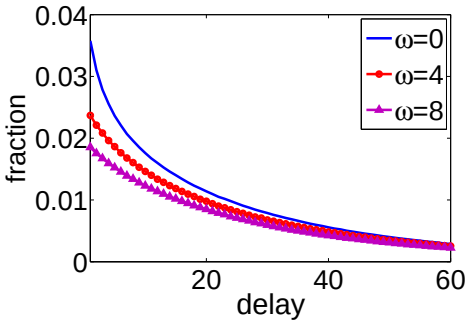

Figure 10.12: Delay distributions when the request inter-arrival time follows Uniform distribution and the request service time follows Weibull distribution.

of proactive serving on delay reduction has a sudden change when ω exceeds some value under bursty arrivals in Fig. 10.8. This is different from the other two cases. We believe that this is related to the statistics of bursty arrivals, which needs future investigations.

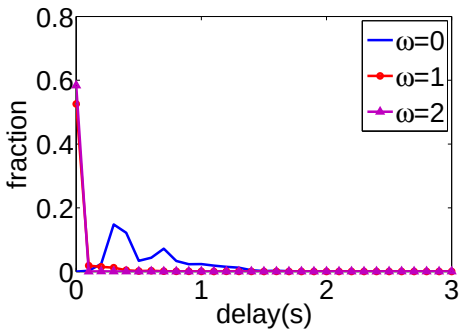

Figure 10.13: Delay distributions under the *YouTube* data trace.

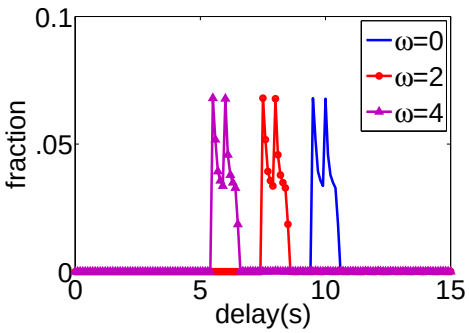

Figure 10.14: Delay distributions under the *Youku* data trace.

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

Table 10.1: Comparison with system capacity boosting under the *Youtube* data trace: The first column is how far the system serves proactively with perfect prediction. The second column is how many percent the number of servers is increased to achieve the same delay performance under proactive serving. The third column is percentage of the decrement of the utilization rate of each server, when the total number of servers increases. The forth column is how many percent the average user delay is decreased. For example, the first row says, serving proactively 120 seconds ahead can decrease the average user delay by 54.1%. To achieve the delay performance, the system can instead increase the number of servers by 10%, which results in that the utilization rate of each server is decreased by 9.1%.

10.4 Comparison with Capacity Boosting

We compare proactive serving under perfect prediction with system capacity boosting under real data traces. The results are shown in Tab. 10.1 and

ω^*(min.)	# of servers ↑	utilization ↓	average delay ↓
10	10%	9.1%	42.6%
18.3	20%	16.7%	66.2%
25	30%	23.1%	81.8%
31.5	40%	28.6%	90.1%

Table 10.2: Comparison with system capacity boosting under the *Youku* data trace: The meaning of each column is the same as Tab. 10.1. For example, the first row says, serving proactively 10 minutes ahead can decrease the average user delay by 42.6%. To achieve the delay performance, the system can instead increase the number of servers by 10%, which results in that the utilization rate of each server is decreased by 9.1%.

10.2. We first observe that, when the system capacity is boosted, the rate of ω^* increasing slows down, where ω^* is how far the system should predict to achieve the same delay performance as capacity boosting. This agrees with what we observed in Fig. 5.1. This suggests that proactive serving is more effective than capacity boosting to achieve low delay, which is very critical for online service systems. Specifically, under the *Youtube* data trace, boosting the capacity by 30% is equivalent to predicting 300 seconds ahead, which both decrease the average user delay by about 90%. Under the *Youku* data trace, boosting the capacity by 40% is equivalent to predicting 31.5 minutes ahead, which both decrease the average user delay by about 90%.

As discussed in chapter5, when the system employs more servers in the system, the server utilization is lower, which leads to resource waste. As observed in Tab. 10.1 and 10.2 under both data traces, the server utilization decreases inverse-proportionally when the system capacity increases.

Chapter 11

Conclusions

In this thesis, we investigate the fundamentals of proactive serving from a queuing theory perspective. We show that for service systems with proactive serving capability, the average user delay decreases exponentially in the prediction window size. More importantly, the delay reduction is robust against prediction errors. We also show that proactive serving decreases both the variance of user delay and the tail of user delay exponentially.

By comparing with conventional capacity boosting mechanism, we show that proactive serving is more effective in decreasing user delay in light workload regime. In particular, the average user delay decays inverse-proportionally in system capacity, but exponentially in the prediction window size under proactive serving.

From a optimization point of view, we demonstrate how to leverage proactive serving when designing systems. The results provide useful insights to system designers.

Our trace driven evaluation results demonstrate the practical power of proactive serving, *e.g.*, under the YouTube data trace of 1000 different videos, user delay can be decreased by 50% when the system can predict 100 seconds ahead. Our study applies to general online service models with prediction, such as preloading of interested content on mobile devices and prefetching of

Youtube videos. Thus, we believe that our results can be used to guide the design of proactive serving in practical online service systems.

We now conclude by discussing possible future directions. First, in section 4.2, we know that the exact impact of the moments of inter-arrival time and service time on delay reduction in $G/G/1^{[\omega]}$ system is open. How to investigate this impact can be an interesting future topic. Second, in our model in section 3.2, we assume that the system has no knowledge of the workload of each user request. A future direction can be how to design the queuing policy to optimize the delay performance if the workload information can be predicted. Third, in section 3.2, besides based on actual prediction algorithms, the system can predict future user requests by users indicating their requests ahead of time. How to design the incentive mechanism to encourage users to report their requests ahead of time can be investigated in the future.

Appendix A

Proofs

In this chapter, we present the detailed proofs of lemmas and theorems in previous chapters.

A.1 Proof of Lemma 1

Under the condition of $Q^p(0) = |A(0:\omega)| + Q_0(0)$, $Q^{sum}(0)$
$= Q^p(0)$. Now consider any time point t. Up 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, at time t, $Q^{sum}(t) = Q^p(t)$.

A.2 Proof of Lemma 2

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 which is (4.2).

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. 7.1, if not getting served beforehand, a request goes through the prediction window $W_\omega(t)$ before it enters Q_0, which costs ω time. If a request spends T time in

Q^p, it will spend $[T-\omega]^+$ slots in Q^0. ($[T-\omega]^+ = 0$ means the request gets pre-served by the system.) Therefore, the delay distribution of requests in Q_0 is a shifted version of that of Q^p.

A.3 Proof of Theorem 1

Based on Lemma 2,

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

Discussion: An alternative approach to obtain $E[D^\omega]$ in Theorem 1 is applying the Markov chain model, which works 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 $\{w_i(t)\}_{1\leq i\leq \omega/\delta}$. 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 to the approach we used in the thesis, 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,

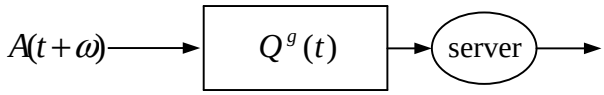

Figure A.1: G/G/1: The arrival process is $\{A(t+\omega)\}_t$. Service times of requests are independent and identically distributed with mean $1/\mu$. The initial value $Q^g(0)$ is $|A(0:\omega)| + Q_0(0)$. The service policy is FCFS.

for the system in chapter 7, the Markov chain that models the discretized system is much more complex, which is rather challenging to solve.

A.4 Proof of Corrolary 1

Consider a general queuing system as shown in Fig. 3.3. We can always find a queuing system without proactive serving as in Fig. A.1, where the arrival process is delayed ω units, 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 [29], the delay distribution of the system in Fig. A.1 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 units.

A.5 Proof of Theorem 2

Let f_{QD^ω} denote the distribution of QD^ω. Let QD^0 denote the queuing delay and f_{QD^0} 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

$$f_{QD^{\omega}}(t) = f_{QD^0}(t+\omega) \text{ for } t > 0 \text{ and } \Pr(QD^{\omega} = 0) = \int_0^{\omega} f_{QD^0}(t)dt.$$

Now we are ready to show that the average queuing delay decreases exponentially in ω. From [30], we have the following about the tail of the distribution of QD^0

$$\Pr(QD^0 \geq t) \leq e^{-s_0 t}. \tag{A.1}$$

Then we have

$$\begin{aligned}
& E[QD^{\omega}] \\
=& \int_0^{\infty} \Pr(QD^{\omega} \geq t)dt \\
=& \int_0^{\infty} \int_t^{\infty} f_{QD^{\omega}}(\tau)d\tau dt \\
=& \int_0^{\infty} \int_t^{\infty} f_{QD^0}(\tau+\omega)d\tau dt \\
=& \int_0^{\infty} \int_{t+\omega}^{\infty} f_{QD^0}(\tau)d\tau dt \\
=& \int_0^{\infty} \Pr(QD^0 \geq t+\omega)dt \\
\leq& \int_0^{\infty} e^{-s_0(t+\omega)}dt \\
=& \frac{1}{s_0} e^{-s_0\omega}.
\end{aligned}$$

A.6 Proof of Theorem 3

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 $\rho = \frac{\lambda}{\mu}$.

A.7 Proof of Lemma 3

From Lemma 2, the distribution of user delay D^ω is, for $t > 0$,

$$f_{D^\omega}(t) = (\mu-\lambda)e^{-(\mu-\lambda)(t+\omega)}.$$

Then the variance of $D^\omega$is

$$\begin{aligned} & Var[D^\omega] \\ = & \int_0^\infty t^2(\mu-\lambda)e^{-(\mu-\lambda)(t+\omega)}dt - \frac{1}{(\mu-\lambda)^2}e^{-2(\mu-\lambda)\omega} \\ = & \frac{2}{(\mu-\lambda)^2}e^{-(\mu-\lambda)\omega} - \frac{1}{(\mu-\lambda)^2}e^{-2(\mu-\lambda)\omega} \\ = & Var[D^0] \cdot e^{-(\mu-\lambda)\omega} \cdot \left[2 - e^{-(\mu-\lambda)\omega}\right]. \end{aligned}$$

Since $1 \leq 2 - e^{-(\mu-\lambda)\omega} \leq 2$, we have the upper and lower bounds of $Var[D^\omega]$ as follows

$$\frac{1}{(\mu-\lambda)^2} \cdot e^{-(\mu-\lambda)\omega} \leq Var[D^\omega] \leq \frac{2}{(\mu-\lambda)^2} \cdot e^{-(\mu-\lambda)\omega}.$$

The upper bound and the lower bound both decrease exponentially in ω. So $Var[D^\omega]$ decreases exponentially in ω.

A.8 Proof of Lemma 4

From Lemma 2, the distribution of user delay D^ω is, for $t > 0$,

$$f_{D^\omega}(t) = (\mu-\lambda)e^{-(\mu-\lambda)(t+\omega)}.$$

Then the tail of user delay is

$$\begin{aligned} & DT^{\omega}(d) \\ = & \int_{d}^{\infty} (\mu-\lambda)e^{-(\mu-\lambda)(t+\omega)}dt \\ = & e^{-(\mu-\lambda)\omega}DT^{0}(d), \end{aligned}$$

which decreases exponentially in ω.

A.9 Proof of Theorem 4

We apply the same idea as we prove Theorem 1.

First, we alter the queuing policy of Q_0. Instead of FCFS, the system gives preemptive priority to the requests of A_1, and within the same arrival process FCFS is adopted. The reasons for doing this are as follows. First, it can simplify the proof of the theorem. Second, most importantly, the average size of Q_0 stays the same and so is the average user delay. Because the average service times of requests in A_1 and A_2 are same, it costs the server the same amount of time to serve a request of A_1 or A_2 on average. During the same amount of time, same amount of new requests enter Q_0 on average. So, under the new queuing policy of Q_0, the average size of Q_0 stays the same. Note that the queuing policy of W_ω stays unchanged.

Under the new queuing policy, $Q^{sum}(t) = Q_0(t) + W_\omega(t)$ evolves the same as the system in Fig. A.2. The proof is similar to that of Lemma 1 and thus omitted. Consider a request of A_2. 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 that A_2 requests 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 A_2 requests 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 $f_2^{\omega}(t)$ easily. Instead of calculating it directly, based on existing knowledge on busy period of

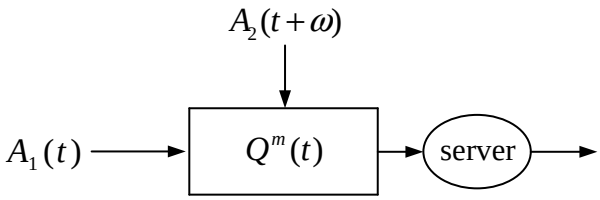

Figure A.2: M/M/1 with preemptive priority: Two arrival processes are $A_1(t)$ and $A_2(t+\omega)$ respectively. Service times of requests are independent and identically exponentially distributed with mean μ. The initial value of Q^m is $|A_2(0:\omega)|+Q_0(0)$. Requests of A_1 have preemptive priority over those of A_2.

M/M/1, we get the Laplace transform of $f_2^m(t)$.

Now we focus on the system in Fig. A.2. Let π_i^m be the stationary probability that the size of Q^m is i. By analyzing the Markov chain which models the evolution of Q^m, we can have $\pi_i^p = (1-\frac{\lambda_1+\lambda_2}{\mu})(\frac{\lambda_1+\lambda_2}{\mu})^i$. Consider a request of A_2. 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

$$P(T \le t) = \sum_{i=0}^{\infty} \pi_i P(T \le t | N = i) = \sum_{i=0}^{\infty} \pi_i P(T_{i+1} \le t). \qquad \text{(A.2)}$$

T_{i+1} is equal to the time till the system is empty again, which is usually called busy period in queuing theory. Denote the probability density functions of T_{i+1} by $f_{i+1}(t)$. By (A.2), we get $f_2^m(t) = \sum_{i=0}^{\infty} \pi_i f_{i+1}(t)$. From [2], we know that the Laplace transform of $f_{i+1}(t)$ 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 user delay of A_2 requests in Q_0, which is denoted by $E[D_2^\omega]$. By the definition of the Laplace transform,

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

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

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

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 inverse $\frac{F(s)}{s}$ and get an 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 $\mathrm{Res}(s_3)$ 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 $\mathrm{Res}(s_4)$ is 1. Consider the closed contour $L + C_R$ shown in Fig. A.3.

$$
\begin{aligned}
&\oint \frac{F(s)}{s}e^{s\omega}ds \\
=&\int_L \frac{F(s)}{s}e^{s\omega}ds + \int_{C_R}\frac{F(s)}{s}e^{s\omega}ds \\
=&\int_{\sigma-jR}^{\sigma+jR}\frac{F(s)}{s}e^{s\omega}ds + \int_{C_R}\frac{F(s)}{s}e^{s\omega}ds \\
=&\lim_{r\to 0}\int_{C_r}\frac{F(s)}{s}e^{s\omega}ds + 2\pi j\cdot\mathrm{Res}(s_4) + 2\pi j\cdot\mathrm{Res}(s_3)\cdot\mathbf{1}_{(\lambda_1+\lambda_2)^2>\lambda_1\mu}, \quad \text{(A.3)}
\end{aligned}
$$

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, $\int_{C_R}\frac{F(s)}{s}e^{s\omega}ds \to 0$ when $R\to\infty$. Thus, to calculate the inverse transform of $\frac{F(s)}{s}$, we only need to calculate $\lim_{r\to0}\int_{C_r}\frac{F(s)}{s}e^{s\omega}ds$.

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

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 (A.3), we obtain $\int_0^\omega f_2^m(t)dt$.

Finally, we get

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

Then we derive the average delay of A_2 requests

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

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 $E[D^\omega]=\frac{\lambda_1}{\lambda_1+\lambda_2}E[D_1]+\frac{\lambda_2}{\lambda_1+\lambda_2}E[D_2^\omega]$.

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. So by (A.4), $E[D_2^\omega]$ 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.

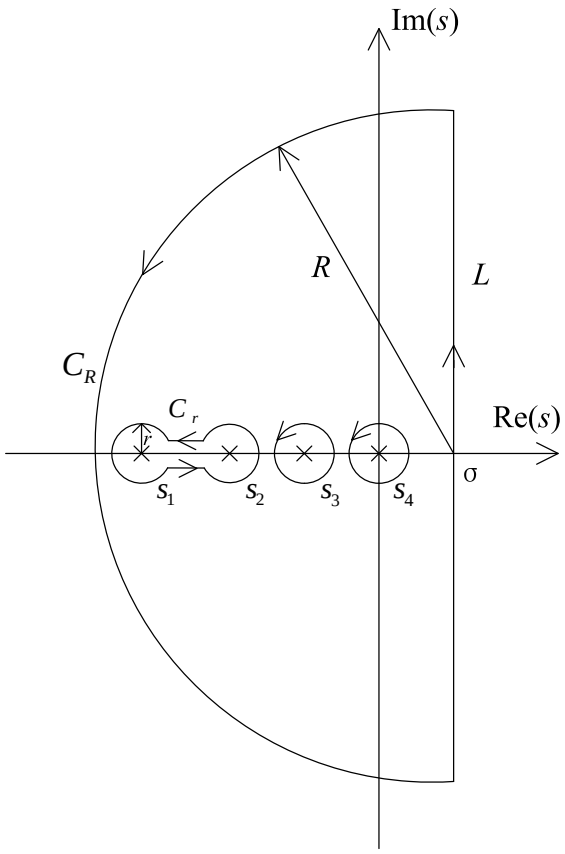

Figure A.3: Contour integration: The contour consists of L and C_R. s_1 and s_2 are branch points. s_3 and s_4 are simple poles.

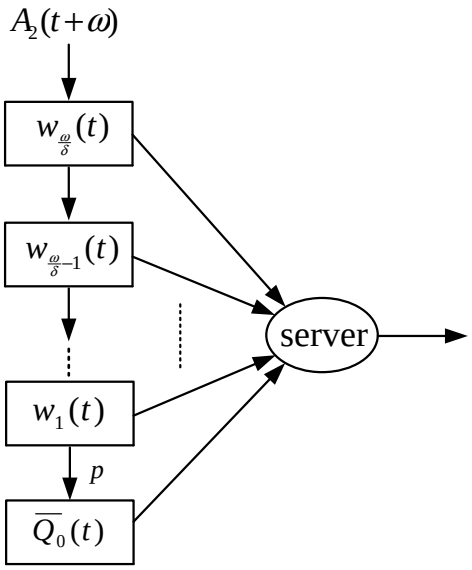

Figure A.4: A discrete service system with only false alarms: The request arrival process $A_2(t)$ and the service process are independent Bernoulli processes. Each request first goes through a pipeline of these small windows from $w_{\omega/\delta}(t)$ to $w_1(t)$ before entering $\bar{Q}_0(t)$. If $A_2(t+i\delta) = 1$, then the system can observe a request in the window $w_i(t)$, which can be served proactively. A request stays in each small window for exact 1 slot. Each request in $\{A_2(t)\}_t$ is independently an actual request with probability p.

A.10 Proof of Theorem 5

First, we discretize the system as shown in Fig. A.4. We chop the time into slots of equal length. Let δ denote the slot length. We choose the value of δ so that ω/δ is an integer. At time slot t, $A_2(t)$ becomes a Bernoulli random variable with probability $\lambda_2 \cdot \delta$. We assume that each request arrives at the end of the time slot. Note that this assumption does not affect the final results. 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. $\bar{Q}_0(t)$ 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 $\{w_i(t)\}_{1\leq i\leq \omega/\delta}$. Each request first goes through a pipeline of these small windows from $w_{\omega/\delta}(t)$ to $w_1(t)$ before entering $\bar{Q}_0(t)$. If $A_2(t+i\delta)=1$, then the system can observe a request in the window $w_i(t)$, which can be served proactively. A request stays in each small window for exact 1 slot and then moves to next window. The false alarm will not enter $\bar{Q}_0(t)$ and thus disappear once they leave the window $w_1(t)$. Each request in $\{A_2(t)\}_t$ is independently an actual request with probability p. In this case, with probability p, the request in the window $w_1(t)$ will enter $\bar{Q}_0(t)$ and with probability $1-p$, it will leave the system. Slightly abusing the notations, we sometimes use $\bar{Q}_0(t), w_i(t)$ to denote the number of requests in it.

The FCFS service policy works as follows. At time slot t, when the server is on, it first checks $\bar{Q}_0(t)$. If $\bar{Q}_0(t)>0$, the server selects the request at the head of $\bar{Q}_0(t)$ and serves it in the slot. Otherwise the server finds the smallest i such that $w_i(t)>0$ and serves the request in $w_i(t)$ in the slot.

Under the FCFS service policy, the system can be modeled by a multi-dimensional Markov chain with state being $\big(w_{\omega/\delta}(t), w_{\omega/\delta-1}(t), ..., w_1(t), \bar{Q}_0(t)\big)^T$ where T means transpose. Define $\bar{\lambda}=\lambda_2\cdot\delta$, $\bar{\mu}=\mu\cdot\delta$ and $\bar{\omega}=\frac{\omega}{\delta}$. Also define $a=\bar{\lambda}(1-\bar{\mu})$, $b=(1-\bar{\lambda})\bar{\mu}$, $c=(1-\bar{\lambda})(1-\bar{\mu})$, $a^{'}=p\bar{\lambda}(1-\bar{\mu})$, $b^{'}=(1-p)(1-\bar{\lambda})\bar{\mu}$, $c^{'}=p(1-\bar{\lambda})(1-\bar{\mu})$, $d^{'}=(1-p)\bar{\lambda}\bar{\mu}$, $a^{''}=p\bar{\lambda}\bar{\mu}+(1-p)\bar{\lambda}(1-\bar{\mu})$, $b^{''}=(1-\bar{\lambda})\bar{\mu}+(1-p)(1-\bar{\lambda})(1-\bar{\mu})$, $c^{''}=p(1-\bar{\lambda})\bar{\mu}+(1-p)(1-\bar{\lambda})(1-\bar{\mu})$, $d^{''}=\bar{\lambda}\bar{\mu}+(1-p)\bar{\lambda}(1-\bar{\mu})$.

When we say the Markov chain is in state i at time slot t, we mean

$$i=\bar{Q}_0(t)+\sum_{j=1}^{\bar{\omega}} w_j(t).$$

Then the state space of the Markov chain is $\{0,1,2,3,...\}$. To calculate the stationary distribution of the Markov chain, as a first step, we derive the transition matrix. Now we first define some matrixes which serve as the

building blocks of the transition matrix. Define

$$\bar{B} = \begin{pmatrix} G \\ F'' \end{pmatrix}, \bar{A}_2 = \begin{pmatrix} F^{(\bar{\omega}-1)} \\ F' \end{pmatrix},$$

$$\bar{A}_1 = \begin{pmatrix} E \\ E'' \end{pmatrix}, \bar{A}_0 = \begin{pmatrix} O^{(2^{\bar{\omega}-1})} \\ E' \end{pmatrix},$$

where

$$O^{(n)} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}_{n \times 2^{\bar{\omega}}},$$

$$F^{(n)} = \begin{pmatrix} b & d & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & b & d & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & b & d & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & d \end{pmatrix}_{2^n \times 2^{\bar{\omega}}},$$

$$E = \begin{pmatrix} c & a & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & c & a & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & c & a & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & a \end{pmatrix}_{2^{\bar{\omega}-1} \times 2^{\bar{\omega}}},$$

$$E' = \begin{pmatrix} c' & a' & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & c' & a' & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & c' & a' & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c' & a' \end{pmatrix}_{2^{\bar{\omega}-1} \times 2^{\bar{\omega}}},$$

$$E'' = \begin{pmatrix} c'' & a'' & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & c'' & a'' & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & c'' & a'' & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c'' & a'' \end{pmatrix}_{2^{\bar{\omega}-1} \times 2^{\bar{\omega}}},$$

$$F^{\prime} = \begin{pmatrix} b' & d' & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & b' & d' & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & b' & d' & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & b' & d' \end{pmatrix}_{2^{\bar{\omega}-1} \times 2^{\bar{\omega}}},$$

$$F'' = \begin{pmatrix} b'' & d'' & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & b'' & d'' & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & b'' & d'' & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & b'' & d'' \end{pmatrix}_{2^{\bar{\omega}-1} \times 2^{\bar{\omega}}},$$

$$G = \begin{pmatrix} 1-\bar{\lambda} & \bar{\lambda} & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & c & a & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & c & a & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & a \end{pmatrix}_{2^{\bar{\omega}-1} \times 2^{\bar{\omega}}}$$

$+\mathbf{1}_{\{\bar{\omega} \geq 2\}} \cdot \sum_{m=0}^{\bar{\omega}-2} \begin{pmatrix} O^{(2^m)} \\ F^{(m)} \\ O^{(2^{\bar{\omega}-1}-2^{m+1})} \end{pmatrix}$, where n is a positive integer and $\mathbf{1}_{\{\cdot\}}$ is

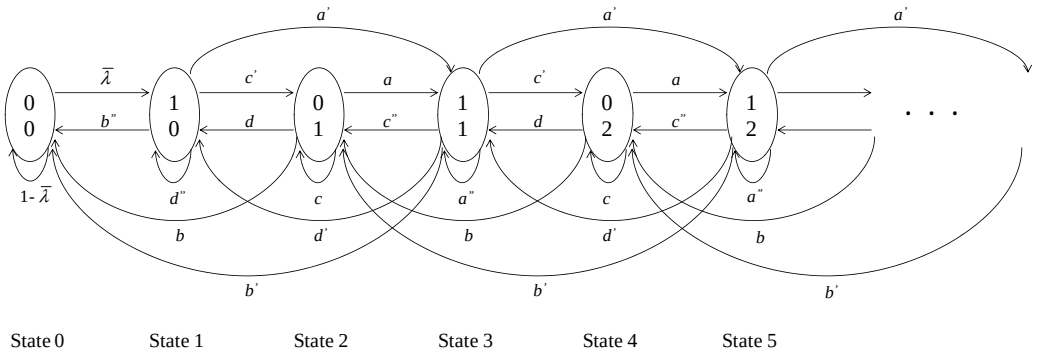

Figure A.5: State diagram when $\bar{\omega} = 1$: In each state, the lower value is the number of requests in $\bar{Q}_0(t)$, and the upper is the number of requests in $w_1(t)$. The state number is calculated by $2 \cdot \bar{Q}_0(t) + w_1(t)$.

the indicator function. Now, the transition matrix of the Markov chain is

$$\bar{\boldsymbol{P}} = \begin{pmatrix} \bar{B} & \bar{A}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \bar{A}_2 & \bar{A}_1 & \bar{A}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \bar{A}_2 & \bar{A}_1 & \bar{A}_0 & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \bar{A}_2 & \bar{A}_1 & \bar{A}_0 & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

As an illustrating example, in Fig. A.5, we draw the state diagram of the Markov chain when $\bar{\omega} = 1$. The matrix blocks which comprise the transition matrix are

$$\bar{B} = \begin{pmatrix} 1-\bar{\lambda} & \bar{\lambda} \\ b'' & d'' \end{pmatrix}, \bar{A}_0 = \begin{pmatrix} 0 & 0 \\ c' & a' \end{pmatrix},$$

$$\bar{A}_2 = \begin{pmatrix} b & d \\ b' & d' \end{pmatrix}, \bar{A}_1 = \begin{pmatrix} c & a \\ c'' & a'' \end{pmatrix}.$$

Now we are ready to calculate the stationary distribution of the Markov chain. Let $\bar{\boldsymbol{\pi}}$ be the stationary distribution which should satisfy $\bar{\boldsymbol{\pi}} \cdot \bar{\boldsymbol{P}} = \bar{\boldsymbol{\pi}}$ and $\sum \bar{\pi}_i = 1$. $\bar{\pi}_i$ is the probability that the system is in state i when the system is stable.

From $\bar{\boldsymbol{\pi}} \cdot \bar{\boldsymbol{P}} = \bar{\boldsymbol{\pi}}$, for states 0 and 1, we have the following balance equations

$$(1-\bar{\lambda})\bar{\pi}_0 + \sum_{j=0}^{\bar{\omega}-2} b\bar{\pi}_{2^j} + b^{''}\bar{\pi}_{2^{\bar{\omega}-1}} + b\bar{\pi}_{2^{\bar{\omega}}} + b^{'}\bar{\pi}_{2^{\bar{\omega}}+2^{\bar{\omega}-1}} = \bar{\pi}_0, \tag{A.5}$$

$$\bar{\lambda}\bar{\pi}_0 + \sum_{j=0}^{\bar{\omega}-2} d\bar{\pi}_{2^j} + d^{''}\bar{\pi}_{2^{\bar{\omega}-1}} + d\bar{\pi}_{2^{\bar{\omega}}} + d^{'}\bar{\pi}_{2^{\bar{\omega}}+2^{\bar{\omega}-1}} = \bar{\pi}_1. \tag{A.6}$$

For positive integers i and j, we define $\frac{ij}{2} = \frac{i}{2} + \lfloor \frac{j}{2} \rfloor$ and $ij = \frac{i}{2} + \lfloor \frac{j}{2} \rfloor$ where $\lfloor \cdot \rfloor$ is the floor opeartion. For state $i+j$ where $i = 2^{\bar{\omega}-k}$ and $0 \leq j < 2^{\bar{\omega}-k} (1 \leq k \leq \bar{\omega}-1)$, we have the following balance equations

$$\bar{\pi}_{i+j} = c\bar{\pi}_{\frac{ij}{2}} + \sum_{l=2}^{k} b\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}-l}} + b^{''}\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}-1}} + b\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}}} + b^{'}\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}}+2^{\bar{\omega}-1}} \tag{A.7}$$

when j is even;

$$\bar{\pi}_{i+j} = a\bar{\pi}_{\frac{ij}{2}} + \sum_{l=2}^{k} d\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}-l}} + d^{''}\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}-1}} + d\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}}} + d^{'}\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}}+2^{\bar{\omega}-1}} \tag{A.8}$$

when j is odd. For state $i+j$ where $i = n \cdot 2^{\bar{\omega}}$ and $0 \leq j < 2^{\bar{\omega}}$ $(n \geq 1)$, we have the following balance equations

$$\bar{\pi}_{i+j} = c^{'}\bar{\pi}_{ij-2^{\bar{\omega}-1}} + c\bar{\pi}_{ij} + c^{''}\bar{\pi}_{ij+2^{\bar{\omega}-1}} + b\bar{\pi}_{ij+2^{\bar{\omega}}} + b^{'}\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{\omega}-1}} \tag{A.9}$$

when j is even;

$$\bar{\pi}_{i+j} = a^{'}\bar{\pi}_{ij-2^{\bar{\omega}-1}} + a\bar{\pi}_{ij} + a^{''}\bar{\pi}_{ij+2^{\bar{\omega}-1}} + d\bar{\pi}_{ij+2^{\bar{\omega}}} + d^{'}\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{\omega}-1}} \tag{A.10}$$

when j is odd.

To solve the above equations, three useful lemmas that describe the relationships among states when the system is stable need to be established. As the first step, based on the equations (A.5)-(A.10), we reveal some direct relationships among states when the system is stable in the following lemma. We denote the set of natural numbers and the set of positive natural numbers by $\mathcal{N}$ and $\mathcal{N}^+$ respectively.

Lemma 14. *For* $i = 2^{\bar{\omega}-k}$ $(1 \leq k \leq \bar{\omega} - 1)$,

$$\frac{\bar{\pi}_{i+j}}{\bar{\pi}_{i+j+2^{l-1}}} = \frac{1-\bar{\lambda}}{\bar{\lambda}}$$

where $j = m \cdot 2^l$ $(m \in \mathcal{N}, l \in \mathcal{N}^+)$ *and* $j < i - 2^{l-1}$. *For* $i = n \cdot 2^{\bar{\omega}}$ $(n \in \mathcal{N}^+)$,

$$\frac{\bar{\pi}_{i+j}}{\bar{\pi}_{i+j+2^{l-1}}} = \frac{1-\bar{\lambda}}{\bar{\lambda}}$$

where $j = m \cdot 2^l$ $(m \in \mathcal{N}, l \in \mathcal{N}^+)$ *and* $j < 2^{\bar{\omega}} - 2^{l-1}$.

Proof. We conduct the proof by induction. First let's verify the result when $l = 1$. When $i = 2^{\bar{\omega}-k}$ $(1 \leq k \leq \bar{\omega} - 1)$, based on (A.7) and (A.8), we can directly get $\frac{\bar{\pi}_{i+j}}{\bar{\pi}_{i+j+1}} = \frac{1-\bar{\lambda}}{\bar{\lambda}}$ where $j = 2 \cdot m$ $(m \in \mathcal{N})$ and $j < i - 1$. Similarly, when $i = n \cdot 2^{\bar{\omega}}$ $(n \in \mathcal{N}^+)$, based on (A.9) and (A.10), we get $\frac{\bar{\pi}_{i+j}}{\bar{\pi}_{i+j+1}} = \frac{1-\bar{\lambda}}{\bar{\lambda}}$ where $j = 2 \cdot m$ $(m \in \mathcal{N})$ and $j < 2^{\bar{\omega}} - 1$.

Suppose when $2 \leq l < \bar{l}$

$$\frac{\bar{\pi}_{i+j}}{\bar{\pi}_{i+j+2^{l-1}}} = \frac{1-\bar{\lambda}}{\bar{\lambda}} \tag{A.11}$$

for $i = 2^{\bar{\omega}-k}$ or $i = n \cdot 2^{\bar{\omega}}$. Now let $l = \bar{l}$. Because of the condition $j < i - 2^{\bar{l}-1}$, $\bar{\omega} - \bar{l} < k \leq \bar{\omega} - 1$ is excluded from $1 \leq k \leq \bar{\omega} - 1$. For $i = 2^{\bar{\omega}-k}$ $(1 \leq k \leq \bar{\omega} - \bar{l})$, $j = m \cdot 2^{\bar{l}}$ $(m \in \mathcal{N})$ and $j < i - 2^{\bar{l}-1}$,

$$\begin{aligned}
&\bar{\pi}_{i+j+2^{\bar{l}-1}} \\
=&c\bar{\pi}_{\frac{ij}{2}+2^{\bar{l}-2}} + \sum_{l'=2}^{k} b\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}-l'}+2^{\bar{l}-2}} + b''\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}-1}+2^{\bar{l}-2}} \\
&+ b\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}}+2^{\bar{l}-2}} + b'\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}}+2^{\bar{\omega}-1}+2^{\bar{l}-2}} \\
=&\frac{\bar{\lambda}}{1-\bar{\lambda}} \left(c\bar{\pi}_{\frac{ij}{2}} + \sum_{l'=2}^{k} b\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}-l'}} + b''\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}-1}} + b\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}}} + b'\bar{\pi}_{\frac{ij}{2}+2^{\bar{\omega}}+2^{\bar{\omega}-1}} \right) \\
=&\frac{\bar{\lambda}}{1-\bar{\lambda}}\bar{\pi}_{i+j},
\end{aligned}$$

where the first and third equality are based on (A.7) and the second equality is derived from (A.11) when $l = \bar{l} - 1$. For $i = n \cdot 2^{\bar{\omega}} (n \in \mathcal{N}^+)$, $j = m \cdot 2^{\bar{l}}$ $(m \in \mathcal{N})$ and $j < 2^{\bar{\omega}} - 2^{\bar{l}-1}$,

$$\begin{aligned}
&\bar{\pi}_{i+j+2^{\bar{l}-1}} \\
=&c^{'}\bar{\pi}_{ij-2^{\bar{\omega}-1}+2^{\bar{l}-2}} + c\bar{\pi}_{ij+2^{\bar{l}-2}} + c^{''}\bar{\pi}_{ij+2^{\bar{\omega}-1}+2^{\bar{l}-2}} \\
&+ b\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{l}-2}} + b^{'}\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{\omega}-1}+2^{\bar{l}-2}} \\
=&\frac{\bar{\lambda}}{1-\bar{\lambda}}\left(c^{'}\bar{\pi}_{ij-2^{\bar{\omega}-1}} + c\bar{\pi}_{ij} + c^{''}\bar{\pi}_{ij+2^{\bar{\omega}-1}} + b\bar{\pi}_{ij+2^{\bar{\omega}}} + b^{'}\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{\omega}-1}}\right) \\
=&\frac{\bar{\lambda}}{1-\bar{\lambda}}\bar{\pi}_{i+j},
\end{aligned}$$

where the first and third equality are based on (A.9) and the second equality is derived from (A.11) when $l = \bar{l} - 1$. □

When $\bar{\omega} = 1$, Lemma 14 tells that $\frac{\bar{\pi}_i}{\bar{\pi}_{i+1}} = \frac{1-\bar{\lambda}}{\bar{\lambda}}$ when i is even.

The Lemma 14 can help us simplify $\sum_{0 \le j < 2^{\bar{\omega}}} \bar{\pi}_{i+j}$ for any $i = n \cdot 2^{\bar{\omega}}$ $(n \in \mathcal{N}^+)$ as follows

$$\begin{aligned}
&\sum_{0 \le j < 2^{\bar{\omega}}} \bar{\pi}_{i+j} \\
=&\frac{1}{1-\bar{\lambda}} \sum_{0 \le j < 2^{\bar{\omega}} : j = 2 \cdot m, m \in \mathcal{N}} \bar{\pi}_{i+j} \\
=&\frac{1}{(1-\bar{\lambda})^2} \sum_{0 \le j < 2^{\bar{\omega}} : j = 4 \cdot m, m \in \mathcal{N}} \bar{\pi}_{i+j} \\
=&\cdots \\
=&\frac{1}{(1-\bar{\lambda})^{\bar{\omega}}}\bar{\pi}_i.
\end{aligned} \tag{A.12}$$

Similarly for any $i = 2^{\bar{\omega}-k}$ $(1 \le k \le \bar{\omega} - 1)$,

$$\sum_{0 \le j < 2^{\bar{\omega}-k}} \bar{\pi}_{i+j} = \frac{1}{(1-\bar{\lambda})^{\bar{\omega}-k}}\bar{\pi}_i. \tag{A.13}$$

As the second step, the following lemma tells the relationship between state 0 and other states when the system is in steady state.

Lemma 15. *For* $i = 2^{\bar{\omega}-k}$ $(1 \leq k \leq \bar{\omega})$, $\bar{\pi}_i = \frac{\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}-k}}{1-\bar{\lambda}}\bar{\pi}_0$. *And* $\bar{\pi}_{2^{\bar{\omega}}} = \frac{p\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}}}{(1-p\bar{\lambda})\bar{\mu}}\bar{\pi}_0$.

Proof. We prove the first part of the lemma by induction. From (A.5) and (A.6), we can easily get $\bar{\pi}_1 = \frac{\bar{\lambda}}{1-\bar{\lambda}}\bar{\pi}_0$. Suppose, when $0 < \bar{k} < k \leq \bar{\omega}$, $\bar{\pi}_{2^{\bar{\omega}-k}} = \frac{\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}-k}}{1-\bar{\lambda}}\bar{\pi}_0$. Now let $k = \bar{k}$. From (A.7),

$$\bar{\pi}_i = c\bar{\pi}_{\frac{i}{2}} + \sum_{l=2}^{\bar{k}} b\bar{\pi}_{\frac{i}{2}+2^{\bar{\omega}-l}} + b^{''}\bar{\pi}_{\frac{i}{2}+2^{\bar{\omega}-1}} + b\bar{\pi}_{\frac{i}{2}+2^{\bar{\omega}}} + b^{'}\bar{\pi}_{\frac{i}{2}+2^{\bar{\omega}}+2^{\bar{\omega}-1}}.$$

At the same time,

$$\bar{\pi}_1 = \bar{\lambda}\bar{\pi}_0 + \sum_{j=0}^{\bar{\omega}-2} d\bar{\pi}_{2^j} + d^{''}\bar{\pi}_{2^{\bar{\omega}-1}} + d\bar{\pi}_{2^{\bar{\omega}}} + d^{'}\bar{\pi}_{2^{\bar{\omega}}+2^{\bar{\omega}-1}}.$$

From Lemma 14, $\frac{\bar{\pi}_{2^{\bar{\omega}}-l}}{\bar{\pi}_{\frac{i}{2}+2^{\bar{\omega}}-l}} = \frac{1-\bar{\lambda}}{\bar{\lambda}}$ when $0 \leq l \leq \bar{k}-1$ and $\frac{\bar{\pi}_{2^{\bar{\omega}}+2^{\bar{\omega}-1}}}{\bar{\pi}_{\frac{i}{2}+2^{\bar{\omega}}+2^{\bar{\omega}-1}}} = \frac{1-\bar{\lambda}}{\bar{\lambda}}$. So

$$\begin{aligned}
&\bar{\pi}_i \\
=&\frac{c\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}-\bar{k}-1}}{1-\bar{\lambda}}\bar{\pi}_0 + \bar{\pi}_1 - \bar{\lambda}\bar{\pi}_0 - \sum_{j=0}^{\bar{\omega}-\bar{k}-1} d\bar{\pi}_{2^j} \\
=&\left[\frac{c\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}-\bar{k}-1}}{1-\bar{\lambda}} + \frac{\bar{\lambda}^2}{1-\bar{\lambda}} - \frac{d\bar{\lambda}}{1-\bar{\lambda}}\sum_{j=0}^{\bar{\omega}-\bar{k}-1}(1-\bar{\mu})^j\right]\bar{\pi}_0 \\
=&\frac{\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}-\bar{k}}}{1-\bar{\lambda}}\bar{\pi}_0.
\end{aligned}$$

From (A.6), we can get $\bar{\pi}_{2^{\bar{\omega}}} = \frac{p\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}}}{(1-p\bar{\lambda})\bar{\mu}}\bar{\pi}_0$ since $\frac{\bar{\pi}_{2^{\bar{\omega}}}}{\bar{\pi}_{2^{\bar{\omega}}+2^{\bar{\omega}-1}}} = \frac{1-\bar{\lambda}}{\bar{\lambda}}$. □

When $\bar{\omega} = 1$, Lemma (15) tells that $\bar{\pi}_2 = \frac{p\bar{\lambda}(1-\bar{\mu})}{(1-p\bar{\lambda})\bar{\mu}}\bar{\pi}_0$.

In the final step, based on Lemma (14) and (15), some hidden relationships among states are disclosed in the following lemma.

Lemma 16. *For any* $2^{\bar{\omega}-1} \leq j < 2^{\bar{\omega}}$,

$$\frac{\bar{\pi}_j}{\bar{\pi}_{2^{\bar{\omega}}+j}} = \frac{\bar{\mu}(1-p\bar{\lambda})}{p\bar{\lambda}(1-\bar{\mu})}.$$

For any $i = n \cdot 2^{\bar{\omega}}$ *and* $0 \leq j < 2^{\bar{\omega}}$ $(n \in \mathcal{N}^+)$,

$$\frac{\bar{\pi}_{i+j}}{\bar{\pi}_{i+2^{\bar{\omega}}+j}} = \frac{\bar{\mu}(1-p\bar{\lambda})}{p\bar{\lambda}(1-\bar{\mu})}.$$

Proof. We first show that any $2^{\bar{\omega}-1} \leq j < 2^{\bar{\omega}}$, we have $\frac{\bar{\pi}_j}{\bar{\pi}_{2^{\bar{\omega}}+j}} = \frac{\bar{\mu}(1-p\bar{\lambda})}{p\bar{\lambda}(1-\bar{\mu})}$. When $j = 2^{\bar{\omega}-1}$, from Lemma 15, $\bar{\pi}_j = \frac{\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}-1}}{1-\bar{\lambda}}\bar{\pi}_0$. From Lemma 14,

$$\bar{\pi}_{2^{\bar{\omega}}+j} = \frac{\bar{\lambda}}{1-\bar{\lambda}}\bar{\pi}_{2^{\bar{\omega}}} = \frac{\bar{\lambda}}{1-\bar{\lambda}} \cdot \frac{p\bar{\lambda}(1-\bar{\mu})^{\bar{\omega}}}{(1-p\bar{\lambda})\bar{\mu}}\bar{\pi}_0.$$

So $\frac{\bar{\pi}_j}{\bar{\pi}_{2^{\bar{\omega}}+j}} = \frac{\bar{\mu}(1-p\bar{\lambda})}{p\bar{\lambda}(1-\bar{\mu})}$. When $2^{\bar{\omega}-1} < j < 2^{\bar{\omega}}$, based on Lemma 14, $\frac{\bar{\pi}_j}{\bar{\pi}_{2^{\bar{\omega}-1}}} = \frac{\bar{\pi}_{2^{\bar{\omega}}+j}}{\bar{\pi}_{2^{\bar{\omega}}+2^{\bar{\omega}-1}}}$. Therefore we get $\frac{\bar{\pi}_j}{\bar{\pi}_{2^{\bar{\omega}}+j}} = \frac{\bar{\mu}(1-p\bar{\lambda})}{p\bar{\lambda}(1-\bar{\mu})}$.

Now suppose, any $i = n \cdot 2^{\bar{\omega}}$ and $0 \leq j < 2^{\bar{\omega}}$ $(n \in \mathcal{N}^+)$, $\frac{\bar{\pi}_{i+j}}{\bar{\pi}_{i+2^{\bar{\omega}}+j}} = \frac{\bar{\mu}(1-p\bar{\lambda})}{p\bar{\lambda}(1-\bar{\mu})}$. Based on (A.9), we have

$$\begin{aligned}
&\bar{\pi}_{i+2^{\bar{\omega}}+j} \\
=&c^{'}\bar{\pi}_{ij+2^{\bar{\omega}}-2^{\bar{\omega}-1}} + c\bar{\pi}_{ij+2^{\bar{\omega}}} + c^{''}\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{\omega}-1}} + b\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{\omega}}} + b^{'}\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{\omega}}+2^{\bar{\omega}-1}} \\
=&\frac{p\bar{\lambda}(1-\bar{\mu})}{\bar{\mu}(1-p\bar{\lambda})} \cdot \left(c^{'}\bar{\pi}_{ij-2^{\bar{\omega}-1}} + c\bar{\pi}_{ij} + c^{''}\bar{\pi}_{ij+2^{\bar{\omega}-1}} + b\bar{\pi}_{ij+2^{\bar{\omega}}} + b^{'}\bar{\pi}_{ij+2^{\bar{\omega}}+2^{\bar{\omega}-1}}\right) \\
=&\frac{p\bar{\lambda}(1-\bar{\mu})}{\bar{\mu}(1-p\bar{\lambda})}\bar{\pi}_{i+j}
\end{aligned}$$

when j is even. When j is odd, we can derive $\frac{\bar{\pi}_{i+j}}{\bar{\pi}_{i+2^{\bar{\omega}}+j}} = \frac{\bar{\mu}(1-p\bar{\lambda})}{p\bar{\lambda}(1-\bar{\mu})}$ in the same way. So we verify that this relation is indeed true. □

When $\bar{\omega} = 1$, Lemma (16) tells that $\frac{\bar{\pi}_i}{\bar{\pi}_{i+2}} = \frac{\bar{\mu}(1-p\bar{\lambda})}{p\bar{\lambda}(1-\bar{\mu})}$.

With Lemma 14, 15 and 16, we are ready to calculate the stationary

distribution. We first calculate $\bar{\pi}_0$ in the following.

$$
\begin{aligned}
&\sum_{i=1}^{\infty} \bar{\pi}_i \\
=&\bar{\pi}_0 + \bar{\pi}_1 + \sum_{k=1}^{\bar{\omega}-1} \sum_{j=0}^{2^{\bar{\omega}-k}-1} \bar{\pi}_{2^{\bar{\omega}-k}+j} + \sum_{n=1}^{\infty} \sum_{j=0}^{2^{\bar{\omega}}-1} \bar{\pi}_{n\cdot 2^{\bar{\omega}}+j} \\
=&\bar{\pi}_0 + \bar{\pi}_1 + \sum_{k=1}^{\bar{\omega}-1} \frac{1}{(1-\bar{\lambda})^{\bar{\omega}-k}} \bar{\pi}_{2^{\bar{\omega}-k}} + \sum_{n=1}^{\infty} \frac{1}{(1-\bar{\lambda})} \bar{\pi}_{n\cdot 2^{\bar{\omega}}} \\
=&\bar{\pi}_0 + \frac{\bar{\lambda}}{1-\bar{\lambda}} \bar{\pi}_0 + \sum_{k=1}^{\bar{\omega}-1} \frac{1}{(1-\bar{\lambda})^{\bar{\omega}-k}} \bar{\pi}_{2^{\bar{\omega}-k}} + \bar{\pi}_{2^{\bar{\omega}}} \frac{1}{(1-\bar{\lambda})^{\bar{\omega}}} \sum_{n=1}^{\infty} \left(\frac{p\bar{\lambda}(1-\bar{\mu})}{\bar{\mu}(1-p\bar{\lambda})} \right)^{n-1} \\
=&\bar{\pi}_0 + \frac{\bar{\lambda}}{1-\bar{\lambda}} \bar{\pi}_0 + \frac{\bar{\lambda}}{1-\bar{\lambda}} \sum_{k=1}^{\bar{\omega}-1} \left(\frac{1-\bar{\mu}}{1-\bar{\lambda}} \right)^{\bar{\omega}-k} \bar{\pi}_0 + \frac{p\bar{\lambda}}{\bar{\mu}-p\bar{\lambda}} \left(\frac{1-\bar{\mu}}{1-\bar{\lambda}} \right)^{\bar{\omega}} \bar{\pi}_0 \\
=&1,
\end{aligned}
$$

where the first equality decompose all states into four parts, the second equality is derived from (A.13) and (A.12), the third and forth equality are based on Lemma 16 and 15 respectively. Then we get

$$
\bar{\pi}_0 = \begin{cases} \frac{\bar{\mu}-\bar{\lambda}}{\bar{\mu}} \frac{1}{\triangle} & \bar{\lambda} \neq \bar{\mu} \\ \frac{(1-\bar{\lambda})(\bar{\mu}-p\bar{\lambda})}{(\bar{\omega}-1)\bar{\lambda}\bar{\mu}+\bar{\mu}-p\bar{\omega}\bar{\lambda}^2} & \bar{\lambda} = \bar{\mu} \end{cases},
$$

where $\triangle = \left[1 - \frac{(1-p)\bar{\lambda}}{\bar{\mu}-p\bar{\lambda}} \left(\frac{1-\bar{\mu}}{1-\bar{\lambda}} \right)^{\bar{\omega}} \right]$. Further the average length of $\bar{Q}_0(t)$, *i.e.*,

$E[\bar{Q}_0(t)]$, is derived as follows.

$$\begin{aligned}
&E[\bar{Q}_0(t)] \\
=&\sum_{n=1}^{\infty} n \cdot \sum_{j=0}^{2^{\bar{\omega}}-1} \bar{\pi}_{n\cdot 2^{\bar{\omega}}+j} \\
=&\frac{1}{(1-\bar{\lambda})^{\bar{\omega}}} \bar{\pi}_{2^{\bar{\omega}}} \sum_{n=1}^{\infty} n \left(\frac{p\bar{\lambda}(1-\bar{\mu})}{\bar{\mu}(1-p\bar{\lambda})}\right)^{n-1} \\
=&\frac{p\bar{\lambda}(1-p\bar{\lambda})\bar{\mu}}{(\bar{\mu}-p\bar{\lambda})^2} \left(\frac{1-\bar{\mu}}{1-\bar{\lambda}}\right)^{\bar{\omega}} \bar{\pi}_0 \\
=&\begin{cases} \frac{p\bar{\lambda}(1-p\bar{\lambda})(\bar{\mu}-\bar{\lambda})}{(\bar{\mu}-p\bar{\lambda})^2} \left(\frac{1-\bar{\mu}}{1-\bar{\lambda}}\right)^{\bar{\omega}} \frac{1}{\triangle} & \bar{\lambda} \neq \bar{\mu} \\ \frac{p\bar{\lambda}(1-\bar{\lambda})(1-p\bar{\lambda})\bar{\mu}}{(\bar{\mu}-p\bar{\lambda})\left((\bar{\omega}-1)\bar{\lambda}\bar{\mu}+\bar{\mu}-p\bar{\omega}\bar{\lambda}^2\right)} & \bar{\lambda} = \bar{\mu} \end{cases}.
\end{aligned}$$

Now we consider the system before discretization. The average length of $Q_0(t)$, *i.e.*, $E[Q_0(t)]$, is given by

$$\begin{aligned}
&E[Q_0(t)] \\
=&\lim_{\delta \to 0} E[\bar{Q}_0(t)] \\
=&\begin{cases} \frac{\lambda(\mu-\lambda_2)}{(\mu-\lambda)^2} \frac{1}{e^{(\mu-\lambda_2)\omega} - \frac{\lambda_2-\lambda}{\mu-\lambda}} & \lambda_2 \neq \mu \\ \frac{\lambda}{(\mu-\lambda)[(\mu-\lambda)\omega+1]} & \lambda_2 = \mu \end{cases}.
\end{aligned}$$

Then by Little's Law, the average user delay

$$E[D^{\omega}] = \begin{cases} \frac{\mu-\lambda_2}{(\mu-\lambda)^2} \frac{1}{e^{(\mu-\lambda_2)\omega} - \frac{\lambda_2-\lambda}{\mu-\lambda}} & \lambda_2 \neq \mu \\ \frac{1}{(\mu-\lambda)[(\mu-\lambda)\omega+1]} & \lambda_2 = \mu \end{cases}.$$

To show that $E[D^{\omega}]$ decreases exponentially in ω, we need the following inequalities

$$\frac{\mu-\lambda_2}{\mu-\lambda} e^{(\mu-\lambda_2)\omega} < e^{(\mu-\lambda_2)\omega} - \frac{\lambda_2-\lambda}{\mu-\lambda} < e^{(\mu-\lambda_2)\omega}$$

when $\mu > \lambda_2$, and

$$\frac{\lambda_2-\mu}{\lambda_2-\lambda} e^{(\lambda_2-\mu)\omega} < e^{(\lambda_2-\mu)\omega} - \frac{\mu-\lambda}{\lambda_2-\lambda} < e^{(\lambda_2-\mu)\omega}$$

when $\mu < \lambda_2$. Then based on the first equality, we have

$$\frac{\mu - \lambda_2}{(\mu - \lambda)^2} e^{-(\mu-\lambda_2)\omega} < E[D^\omega] < \frac{1}{\mu - \lambda} e^{-(\mu-\lambda_2)\omega}$$

when $\mu > \lambda_2$. When $\mu < \lambda_2$,

$$\begin{aligned} & E[D^\omega] \\ =& \frac{\lambda_2 - \mu}{(\mu - \lambda)(\lambda_2 - \lambda)} + \frac{\lambda_2 - \mu}{(\lambda_2 - \lambda)^2} \frac{1}{e^{(\lambda_2-\mu)\omega} - \frac{\mu-\lambda}{\lambda_2-\lambda}}. \end{aligned}$$

Based on the second equality, we have

$$E[D^\omega] > \frac{\lambda_2 - \mu}{(\mu - \lambda)(\lambda_2 - \lambda)} + \frac{\lambda_2 - \mu}{(\lambda_2 - \lambda)^2} e^{-(\lambda_2-\mu)\omega}$$

and

$$E[D^\omega] < \frac{\lambda_2 - \mu}{(\mu - \lambda)(\lambda_2 - \lambda)} + \frac{1}{\lambda_2 - \lambda} e^{-(\lambda_2-\mu)\omega}.$$

Now we have obtained a lower and upper bound of $E[D^\omega]$ which both decrease exponentially in ω. Therefore, $E[D^\omega]$ decreases exponentially in ω when $\mu \neq \lambda_2$.

A.11 Proof of Lemma 5

In the same way as we prove Theorem 1, we can find a M/M/k system with a properly initialized queue so that the total time that a user request spends in the M/M/$k^{[\omega]}$ system is statistically the same as that in the M/M/k system. As a result, the delay distribution of M/M/$k^{[\omega]}$ is a "shifted" version of that of M/M/k.

The delay distribution of M/M/k is given by, for $t \geq 0$,

$$\begin{aligned} & f_{D^0}(t) \\ =& \begin{cases} q_0 \mu e^{-\mu t} + (1 - q_0)\mu^2 t e^{-\mu t} & \rho = k - 1 \\ \eta_1 \mu^2 e^{-\mu t} + \eta_2 \mu^2 (k - \rho)^2 e^{-\mu(k-\rho)t} & 0 \leq \rho < k \text{ and } \rho \neq k - 1 \end{cases}. \end{aligned}$$

Then the distribution of D^ω is, for $t > 0$,

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

and $\Pr(D^\omega = 0) = \int_0^\omega f_{D^0}(t)dt$. So the average user delay, *i.e.*, $E[D^\omega]$ is calculated as follows

$$\begin{aligned} & E[D^\omega] \\ =& \int_0^\infty t \cdot f_{D^\omega}(t)dt \\ =& \begin{cases} (\frac{2-q_0}{\mu} + (1-q_0)\omega)e^{-\mu\omega} & \rho = (k-1) \\ \eta_1 e^{-\mu\omega} + \eta_2 e^{-\mu(k-\rho)\omega} & 0 \le \rho < k \text{ and } \rho \neq k-1 \end{cases}. \end{aligned}$$

A.12 Proof of Lemma 6

First, it is easy to show that there exists $0 < \epsilon < \frac{e-1}{e}$ such that $\mu\omega < e^{(1-\epsilon)\mu\omega}$. Then, when $\rho = k - 1$, we have

$$E[D^\omega] < \frac{2-q_0}{\mu}e^{-\mu\omega} + \frac{1-q_0}{\mu}e^{-\epsilon\mu\omega} < \frac{3-2q_0}{\mu}e^{-\epsilon\mu\omega}.$$

Because $0 < q_0 < 1$, a lower bound of $E[D^\omega]$ is $\frac{2-q_0}{\mu}e^{-\mu\omega}$.

When $\rho \neq k - 1$, we derive the following relationship between $\eta_1$and η_2.

$$\eta_1 + \eta_2 = \frac{1 + k - q_0 - \rho}{(k-\rho)\mu} > 0,$$

where the inequality is due to the fact that $0 < q_0 < 1$ and $\rho < k$. Now we consider three different scenarios separately.

When $\rho < k - 1$, we have $\eta_1 > 0$ and $\eta_2 < 0$. So we can get a upper bound of $E[D^\omega]$ as follows

$$E[D^\omega] < \eta_1 e^{-\mu\omega}.$$

On the other hand, because $e^{-\mu(k-\rho)\omega} < e^{-\mu\omega}$, we have $\eta_2 e^{-\mu(k-\rho)\omega} > \eta_2 e^{-\mu\omega}$. Then we get a lower bound of $E[D^\omega]$ as follows:

$$E[D^\omega] > \eta_1 e^{-\mu\omega} + \eta_2 e^{-\mu\omega} = \eta_3 e^{-\mu\omega}.$$

When $k-1<\rho<k-q_0$, we have $\eta_1<0$ and $\eta_2>0$. So we can get a upper bound of $E[D^\omega]$ as follows

$$E[D^\omega]<\eta_2 e^{-k\mu(1-\rho)\omega}.$$

On the other hand, because $e^{-\mu\omega}<e^{-\mu(k-\rho)\omega}$, we have $\eta_1 e^{-\mu\omega}>\eta_1 e^{-\mu(k-\rho)\omega}$. Then we get a lower bound of $E[D^\omega]$ as follows:

$$E[D^\omega]>\eta_1 e^{-\mu(k-\rho)\omega}+\eta_2 e^{-\mu(k-\rho)\omega}=\eta_3 e^{-\mu(k-\rho)\omega}.$$

When $k-q_0\leq\rho<k$, we have $\eta_1\geq 0$, $\eta_2>0$ and $e^{-\mu\omega}<e^{-\mu(k-\rho)\omega}$. Then we can obtain

$$(\eta_1+\eta_2)e^{-\mu\omega}<E[D^\omega]<(\eta_1+\eta_2)e^{-\mu(k-\rho)\omega}.$$

A.13 Proof of Lemma 7

From Theorem 5, the distribution of user delay D^ω is, for $t>0$,

$$f_{D^\omega}(t)=\begin{cases} q_0\mu e^{-\mu(t+\omega)}+(1-q_0)\mu^2(t+\omega)e^{-\mu(t+\omega)} & \rho=k-1 \\ \eta_1\mu^2 e^{-\mu(t+\omega)}+\eta_2\mu^2(k-\rho)^2 e^{-\mu(k-\rho)(t+\omega)} & 0\leq\rho<k \text{ and } \rho\neq k-1 \end{cases}.$$

Then we calculate the variance of D^ωas follows.

When $0\leq\rho<k$ and $\rho\neq k-1$,

$$\begin{aligned} & Var[D^\omega] \\ = & \int_0^\infty t^2\left[\eta_1\mu^2 e^{-\mu(t+\omega)}+\eta_2\mu^2(k-\rho)^2 e^{-\mu(k-\rho)(t+\omega)}\right]dt-E^2[D^\omega] \\ = & \eta_1 e^{-\mu\omega}\left[\frac{2}{\mu}-\eta_1 e^{-\mu\omega}-\eta_2 e^{-\mu(k-\rho)\omega}\right]+ \\ & \eta_2 e^{-\mu(k-\rho)\omega}\left[\frac{2}{\mu(k-\rho)}-\eta_1 e^{-\mu\omega}-\eta_2 e^{-\mu(k-\rho)\omega}\right]. \end{aligned}$$

We can easily get a upper bound of $Var[D^\omega]$, which is

$$Var[D^\omega]\leq\frac{2\eta_1}{\mu}e^{-\mu\omega}\cdot\mathbf{1}_{\eta_1>0}+\frac{2\eta_2}{\mu(k-\rho)}e^{-\mu(k-\rho)\omega}\cdot\mathbf{1}_{\eta_2>0}.$$

The upper bound decreases exponentially in ω.

On the other hand, we have

$$Var[D^{\omega}] \geq \eta_1 e^{-\mu\omega}\left[\frac{2}{\mu} - \eta_1 - \eta_2\right] + \eta_2 e^{-\mu(k-\rho)\omega}\left[\frac{2}{\mu(k-\rho)} - \eta_1 - \eta_2\right].$$

Define $\eta_4 = \frac{2}{\mu} - \eta_1 - \eta_2$ and $\eta_5 = \frac{2}{\mu(k-\rho)} - \eta_1 - \eta_2$. Then we have

$$\begin{aligned}
& \eta_1\eta_4 + \eta_2\eta_5 \\
= & \frac{\rho + q_0 - k}{\mu(\rho + 1 - k)}\left[\frac{2}{\mu} - \frac{1 + k - q_0 - \rho}{(k-\rho)\mu}\right] + \\
& \frac{1 - q_0}{\mu(k-\rho)(\rho+1-k)}\left[\frac{2}{\mu(k-\rho)} - \frac{1+k-q_0-\rho}{(k-\rho)\mu}\right] \\
= & \frac{1}{\mu^2(k-\rho)^2(\rho+1-k)}\left[(\rho+q_0-k)(\rho-q_0-k+1)(\rho-k)\right. \\
& \left.+(\rho+q_0-k+1)(1-q_0)\right] \\
= & \frac{1}{\mu^2(k-\rho)^2}\left[(\rho-k)^2 + (1-q_0^2)\right] \\
> & 0.
\end{aligned}$$

Now we consider different scenarios separately.

When $\rho < k - 1 - q_0$, we have $\eta_1\eta_4 > 0$, $\eta_2\eta_5 > 0$ and $e^{-\mu(k-\rho)\omega} < e^{-\mu\omega}$. Then we obtain a lower bound of $Var[D^{\omega}]$ as follows

$$Var[D^{\omega}] > (\eta_1\eta_4 + \eta_2\eta_5)e^{-\mu(k-\rho)\omega}.$$

When $k - 1 - q_0 \leq \rho < k - 1$, we have $\eta_1\eta_4 > 0$, $\eta_2\eta_5 \leq 0$. On the other hand, because $e^{-\mu(k-\rho)\omega} < e^{-\mu\omega}$, we have $\eta_2\eta_5 e^{-\mu(k-\rho)\omega} \geq \eta_2\eta_5 e^{-\mu\omega}$. Then we obtain a lower bound of $Var[D^{\omega}]$ as follows

$$Var[D^{\omega}] \geq (\eta_1\eta_4 + \eta_2\eta_5)e^{-\mu\omega}.$$

When $k - 1 < \rho < k - q_0$, we have $\eta_1\eta_4 < 0$, $\eta_2\eta_5 > 0$. On the other hand, because $e^{-\mu(k-\rho)\omega} > e^{-\mu\omega}$, we have $\eta_1\eta_4 e^{-\mu\omega} > \eta_1\eta_4 e^{-\mu(k-\rho)\omega}$. Then we obtain a lower bound of $Var[D^{\omega}]$ as follows

$$Var[D^{\omega}] > (\eta_1\eta_4 + \eta_2\eta_5)e^{-\mu(k-\rho)\omega}.$$

When $k-q_0 \leq \rho < k-1+q_0$, we have $\eta_1\eta_4 \geq 0$, $\eta_2\eta_5 > 0$ and $e^{-\mu(k-\rho)\omega} > e^{-\mu\omega}$. Then we obtain a lower bound of $Var[D^\omega]$ as follows

$$Var[D^\omega] > (\eta_1\eta_4 + \eta_2\eta_5)e^{-\mu\omega}.$$

Note that here we only consider the case when $k - q_0 < k - 1 + q_0$. Similar result holds when $k - q_0 \geq k - 1 + q_0$.

When $k - 1 + q_0 \leq \rho < k$, we have $\eta_1\eta_4 \leq 0$, $\eta_2\eta_5 > 0$. On the other hand, because $e^{-\mu(k-\rho)\omega} > e^{-\mu\omega}$, we have $\eta_1\eta_4 e^{-\mu\omega} \geq \eta_1\eta_4 e^{-\mu(k-\rho)\omega}$. Then we obtain a lower bound of $Var[D^\omega]$ as follows

$$Var[D^\omega] \geq (\eta_1\eta_4 + \eta_2\eta_5)e^{-\mu(k-\rho)\omega}.$$

As seen, the lower bound of $Var[D^\omega]$ decreases exponentially in ω.

When $\rho = k - 1$,

$$\begin{aligned}
& Var[D^\omega] \\
= & \int_0^\infty t^2 \left[q_0\mu e^{-\mu(t+\omega)} + (1-q_0)\mu^2(t+\omega)e^{-\mu(t+\omega)}\right] dt - E^2[D^\omega] \\
= & \frac{e^{-\mu\omega}}{\mu}\left[\frac{6-4q_0}{\mu} - \frac{(2-q_0)^2}{\mu}e^{-\mu\omega} - (2-q_0)(1-q_0)\omega e^{-\mu\omega}\right] + \\
& \omega e^{-\mu\omega}\left[\frac{2(1-q_0)}{\mu} - \frac{(2-q_0)(1-q_0)}{\mu}e^{-\mu\omega} - (1-q_0)^2\omega e^{-\mu\omega}\right].
\end{aligned}$$

We can easily get that

$$Var[D^\omega] \leq \frac{6-4q_0}{\mu^2}e^{-\mu\omega} + \frac{2(1-q_0)}{\mu}\omega e^{-\mu\omega}.$$

From Lemma 6, we know that there exists $0 < \epsilon < \frac{e-1}{e}$ such that $\mu\omega < e^{(1-\epsilon)\mu\omega}$. Then we have

$$Var[D^\omega] \leq \frac{6-4q_0}{\mu^2}e^{-\mu\omega} + \frac{2(1-q_0)}{\mu^2}e^{-\epsilon\mu\omega} \leq \frac{8-6q_0}{\mu^2}e^{-\epsilon\mu\omega},$$

which decreases exponentially in ω.

On the other hand, because the maximum value of $\omega e^{-\mu\omega}$ is $\frac{1}{e\mu}$, we have

$$Var[D^\omega] \geq \frac{e^{-\mu\omega}}{\mu^2}\left[2 - q_0^2 - \frac{(2-q_0)(1-q_0)}{e}\right] + \frac{\omega e^{-\mu\omega}}{\mu}\left[q_0 - q_0^2 - \frac{(1-q_0)^2}{e}\right].$$

Define $\eta_6 = 2 - q_0^2 - \frac{(2-q_0)(1-q_0)}{e}$ and $\eta_7 = q_0 - q_0^2 - \frac{(1-q_0)^2}{e}$. Because $0 < q_0 < 1$, we can get that $\eta_6 > 0$. At the same time,

$$\begin{aligned} & \eta_6 + \eta_7 \\ = \; & 2 - \frac{3}{e} + q_0 \left[1 + \frac{5}{e} - (2 + \frac{2}{e}) q_0 \right] \\ > \; & 2 - \frac{3}{e} + q_0 (\frac{3}{e} - 1) \\ > \; & 0. \end{aligned}$$

When $\frac{1}{e+1} \leq q_0 < 1$, we have $\eta_7 \geq 0$. Then we obtain a lower bound of $Var[D^\omega]$ as follows

$$Var[D^\omega] \geq \frac{\eta_6 e^{-\mu\omega}}{\mu^2}.$$

When $0 < q_0 < \frac{1}{e+1}$, we have $\eta_7 < 0$. As a result, we have $\frac{\eta_7 \omega}{\mu} e^{-\mu\omega} > \frac{\eta_7}{\mu^2} e^{-\epsilon\mu\omega}$. Then we obtain a lower bound of $Var[D^\omega]$ as follows

$$Var[D^\omega] > \frac{\eta_6}{\mu^2} e^{-\mu\omega} + \frac{\eta_7}{\mu^2} e^{-\epsilon\mu\omega} > \frac{\eta_6 + \eta_7}{\mu^2} e^{-\mu\omega}.$$

As seen, the lower bound of $Var[D^\omega]$ decreases exponentially in ω.

A.14 Proof of Lemma 8

From Theorem 5, the distribution of user delay D^ω is, for $t > 0$,

$$f_{D^\omega}(t) = \begin{cases} q_0 \mu e^{-\mu(t+\omega)} + (1-q_0)\mu^2 (t+\omega) e^{-\mu(t+\omega)} & \rho = k-1 \\ \eta_1 \mu^2 e^{-\mu(t+\omega)} + \eta_2 \mu^2 (k-\rho)^2 e^{-\mu(k-\rho)(t+\omega)} & 0 \leq \rho < k \text{ and } \rho \neq k-1 \end{cases}.$$

Then we calculate the tail of user delay as follows.

When $0 \leq \rho < k$ and $\rho \neq k-1$,

$$\begin{aligned} & DT^\omega(d) \\ = \; & \int_d^\infty \eta_1 \mu^2 e^{-\mu(t+\omega)} + \eta_2 \mu^2 (k-\rho)^2 e^{-\mu(k-\rho)(t+\omega)} dt \\ = \; & \eta_1 \mu e^{-\mu(\omega+d)} + \eta_2 \mu (k-\rho) e^{-\mu(k-\rho)(\omega+d)}. \end{aligned}$$

First note that

$$\begin{aligned} & \eta_1\mu + \eta_2\mu(k-\rho) \\ = & \frac{\rho + q_0 - k}{\rho + 1 - k} + \frac{1 - q_0}{\rho + 1 - k} \\ = & 1. \end{aligned}$$

Then when $\rho < k - 1$, we have $\eta_1\mu > 0$ and $\eta_2\mu(k-\rho) < 0$. So a upper bound of $DT^{\omega}(d)$ is given by

$$DT^{\omega}(d) < \eta_1\mu e^{-\mu(\omega+d)}.$$

On the other hand, because $e^{-\mu(k-\rho)(\omega+d)} < e^{-\mu(\omega+d)}$, we have $\eta_2\mu(k-\rho)e^{-\mu(k-\rho)(\omega+d)} > \eta_2\mu(k-\rho)e^{-\mu(\omega+d)}$. Then we get a lower bound of $DT^{\omega}(d)$ as follows

$$DT^{\omega}(d) > e^{-\mu(\omega+d)}.$$

When $k - 1 < \rho < k - q_0$, we have $\eta_1\mu < 0$ and $\eta_2\mu(k-\rho) > 0$. So a upper bound of $DT^{\omega}(d)$ is given by

$$DT^{\omega}(d) < \eta_2\mu(k-\rho)e^{-\mu(k-\rho)(\omega+d)}.$$

On the other hand, because $e^{-\mu(k-\rho)(\omega+d)} > e^{-\mu(\omega+d)}$, we have $\eta_1\mu e^{-\mu(\omega+d)} > \eta_1\mu e^{-\mu(k-\rho)(\omega+d)}$. Then we get a lower bound of $DT^{\omega}(d)$ as follows

$$DT^{\omega}(d) > e^{-\mu(k-\rho)(\omega+d)}.$$

When $k - q_0 \leq \rho < k$, we have we have $\eta_1\mu \geq 0$, $\eta_2\mu(k-\rho) > 0$ and $e^{-\mu(k-\rho)(\omega+d)} > e^{-\mu(\omega+d)}$. Then we get a upper bound and a lower bound of $DT^{\omega}(d)$ as follows

$$e^{-\mu(\omega+d)} < DT^{\omega}(d) < e^{-\mu(k-\rho)(\omega+d)}.$$

When $\rho = k - 1$,

$$\begin{aligned} & DT^{\omega}(d) \\ = & \int_d^{\infty} q_0\mu e^{-\mu(t+\omega)} + (1-q_0)\mu^2(t+\omega)e^{-\mu(t+\omega)}dt \\ = & q_0 e^{-\mu(\omega+d)} + (1-q_0)(\mu d + \mu\omega + 1)e^{-\mu(\omega+d)}. \end{aligned}$$

We can easily get a lower bound of $DT^{\omega}(d)$ as follows

$$DT^{\omega}(d) > [q_0 + (1-q_0)(\mu d+1)]\, e^{-\mu(\omega+d)}.$$

From Lemma 6, we know that there exists $0 < \epsilon < \frac{e-1}{e}$ such that $\mu\omega < e^{(1-\epsilon)\mu\omega}$. Then we have

$$DT^{\omega}(d) < [q_0 + (1-q_0)(\mu d+1)]\, e^{-\mu(\omega+d)} + e^{-\epsilon\mu\omega+\mu d} < [2 + (1-q_0)\mu d]\, e^{-\epsilon\mu\omega+\mu d}.$$

We can see that the lower and upper bound of $DT^{\omega}(d)$ both decrease exponentially in ω. Therefore, $DT^{\omega}(d)$ decreases exponentially in ω.

A.15 Proof of Lemma 9 and 13

In the same way as we prove Theorem 3, we can get the $\omega^*(m)$ for M/M/$k^{[\omega]}$ and Markovian/Geo/$1^{[\omega]}$ as shown in Lemma 9 and 13 respectively.

A.16 Proof of Lemma 10

Similar to the way that we prove Theorem 1 and 5, we can find a Markovian/Geo/1 system with a properly initialized queue so that the total time that a user request spends in the Markovian/Geo/$1^{[\omega]}$ system is statistically the same as that in the Markovian/Geo/1 system. As a result, the delay distribution of Markovian/Geo/$1^{[\omega]}$ is a "shifted" version of that of Markovian/Geo/1.

The delay distribution of Markovian/Geo/1 is given by, for $i \in \mathcal{N}^+$,

$$\Pr(D^0 = i) = \frac{(\alpha+\beta)(\mu-\lambda)}{1-\mu}\left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^i.$$

Then the distribution of D^{ω} is, for $i \in \mathcal{N}^+$,

$$\Pr(D^{\omega} = i) = \Pr(D^0 = i+\omega),$$

and $\Pr(D^\omega = 0) = \sum_{i=1}^{\omega} \Pr(D^0 = i)$. So the average user delay, *i.e.*, $E[D^\omega]$ is calculated as follows

$$\begin{aligned} & E[D^\omega] \\ =& \sum_{i=1}^{\infty} i \cdot \Pr(D^\omega = i) \\ =& \frac{1-\mu}{(\alpha+\beta)(\mu-\lambda)} \left[\frac{1-\mu}{(1-\mu)+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega-1}. \end{aligned}$$

A.17 Proof of Lemma 11

From Lemma 10, the distribution of user delay D^ω is, for $i \in \mathcal{N}^+$,

$$\Pr(D^\omega = i) = \frac{(\alpha+\beta)(\mu-\lambda)}{1-\mu} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{i+\omega}.$$

Then the variance of $D^\omega$is

$$\begin{aligned} & Var[D^\omega] \\ = & \sum_{i=1}^{\infty} i^2 \frac{(\alpha+\beta)(\mu-\lambda)}{1-\mu} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{i+\omega} - E^2[D^\omega] \\ = & \frac{(1-\mu)\left[2(1-\mu)+(\alpha+\beta)(\mu-\lambda)\right]}{(\alpha+\beta)^2(\mu-\lambda)^2} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega-1} \\ & -\frac{(1-\mu)^2}{(\alpha+\beta)^2(\mu-\lambda)^2} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{2\omega-2} \\ = & Var[D^0] \frac{2(1-\mu)+(\alpha+\beta)(\mu-\lambda)}{1-\mu} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega} \\ & -Var[D^0] \frac{1-\mu+(\alpha+\beta)(\mu-\lambda)}{1-\mu} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{2\omega} \\ = & Var[D^0] \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega} \left(\frac{2(1-\mu)+(\alpha+\beta)(\mu-\lambda)}{1-\mu}\right. \\ & \left. -\frac{1-\mu+(\alpha+\beta)(\mu-\lambda)}{1-\mu} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega}\right). \end{aligned}$$

Similar to Lemma 3, we have the upper and lower bounds of $Var[D^\omega]$ as follows

$$Var[D^\omega] \leq Var[D^0] \frac{2(1-\mu)+(\alpha+\beta)(\mu-\lambda)}{1-\mu} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^\omega$$

and

$$Var[D^\omega] \geq Var[D^0] \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^\omega.$$

The upper bound and the lower bound both decrease exponentially in ω. So $Var[D^\omega]$ decreases exponentially in ω.

A.18 Proof of Lemma 12

From Lemma 10, the distribution of user delay D^ω is, for $i \in \mathcal{N}^+$,

$$\Pr(D^\omega = i) = \frac{(\alpha+\beta)(\mu-\lambda)}{1-\mu} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{i+\omega}.$$

Then the tail of user delay is

$$\begin{aligned} & DT^\omega(d) \\ = & \sum_{d+1}^{\infty} \frac{(\alpha+\beta)(\mu-\lambda)}{1-\mu} \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{i+\omega} \\ = & \left[\frac{1-\mu}{1-\mu+(\alpha+\beta)(\mu-\lambda)}\right]^{\omega} DT^0(d), \end{aligned}$$

which decreases exponentially in ω.

Bibliography

[1] http://v.youku.com/v_show/id_XNjI2MzU5ODYw.html.

[2] http://www.win.tue.nl/~iadan/que/h4.pdf.

[3] Branch cut. http://en.wikipedia.org/wiki/Branch_point.

[4] Kindle fire. http://www.amazon.com/gp/product/b0051vvob2.

[5] Residue theorem. http://en.wikipedia.org/wiki/Residue_theorem.

[6] Youku. http://www.youku.com.

[7] Youtube. http://www.youtube.com.

[8] A. O. Allen. *Probability, statistics, and queueing theory: with computer science applications*. Gulf Professional Publishing, 1990.

[9] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, M. Kyriakakos, and A. Kalousis. Predicting the location of mobile users: a machine learning approach. In *International Conference on Pervasive Services*, 2009.

[10] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting. Scheduling in a queuing system with asynchronously varying service rates. *Probability in the Engineering and Informational Sciences*, 2004.

[11] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar. Providing quality of service over a shared wireless link. *IEEE Communications Magazine*, 2001.

[12] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[13] B. Chandrasekaran. Survey of network traffic models. *Informe téc*, 2009.

[14] C. Chen and S. J. Baek. Reducing delays by network coding for wireless broadcasting in networks using relay stations. In *International Symposium on Personal Indoor and Mobile Radio Communications*, 2012.

[15] X. Chen and X. Zhang. A popularity-based prediction model for web prefetching. *IEEE Computer*, 2003.

[16] X. Chen and X. Zhang. Coordinated data prefetching for web contents. *Computer Communications*, 2005.

[17] B. Cheng, X. Liu, Z. Zhang, and H. Jin. A measurement study of a peer-to-peer video-on-demand system. In *Proc. IPTPS*, 2007.

[18] B. Coleman. Quality vs. performance in lookahead scheduling. In *JCIS*, 2006.

[19] V. de Nitto Personè, V. Grassi, and A. Morlupi. Modeling and evaluation of prefetching policies for context-aware information services. In *International Conference on Mobile Computing and Networking*, 1998.

[20] A. Eryilmaz, A. Ozdaglar, and M. Medard. On delay performance gains from network coding. In *Annual Conference on Information Sciences and Systems*, 2006.

[21] A. Eryilmaz, A. Ozdaglar, M. Médard, and E. Ahmed. On the delay and throughput gains of coding in unreliable networks. *IEEE Trans. Information Theory*, 2008.

[22] M. U. Farooq and L. K. John. Store-load-branch (slb) predictor: A compiler assisted branch prediction for data dependent branches. In *Proceedings of International Symposium on High Performance Computer Architecture*, 2013.

[23] G. Gursun, M. Crovella, and I. Matta. Describing and forecasting video access patterns. In *Proc. INFOCOM*, 2011.

[24] B. D. Higgins, J. Flinn, T. J. Giuli, B. Noble, C. Peplin, and D. Watson. Informed mobile prefetching. In *Proc. MobiSys*, 2012.

[25] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran. Codes can reduce queueing delay in data centers. In *Proc. ISIT*, 2012.

[26] W. J. Jeon and K. Nahrstedt. Peer-to-peer multimedia streaming and caching service. In *Proc. ICME*, 2002.

[27] B. Ji, C. Joo, and N. B. Shroff. Delay-based back-pressure scheduling in multi-hop wireless networks. In *Proc. INFOCOM*, 2011.

[28] A. Khan, X. Yan, S. Tao, and N. Anerousis. Workload characterization and prediction in the cloud: A multiple time series approach. In *IEEE Network Operations and Management Symposium*, 2012.

[29] L. Kleinrock. *Queueing systems. volume 1: Theory.* Wiley-Interscience, 1975.

[30] L. Kleinrock. *Queueing systems, volume II: Computer applications.* Wiley-Interscience, 1976.

[31] R. Kohavi and R. Longbotham. Online experiments: Lessons learned. *IEEE Computer*, 2007.

[32] D. Kotz and C. S. Ellis. Practical prefetching techniques for parallel file systems. In *Parallel and Distributed Information Systems*, 1991.

[33] J. R. Larus and T. Ball. Branch prediction for free. In *SIGPLAN Conference on Programming Language Design and Implementation*, 1993.

[34] J. Lee, H. Kim, and R. Vuduc. When prefetching works, when it doesn't, and why. *ACM Trans. Architecture and Code Optimization*, 2012.

[35] D. E. Lucani, M. Médard, and M. Stojanovic. On coding for delay–network coding for time-division duplexing. *IEEE Trans. Information Theory*, 2012.

[36] M. Mandelbaum and D. Shabtay. Scheduling unit length jobs on parallel machines with lookahead information. *Journal of Scheduling*, 2011.

[37] W. Mao and R. K. Kincaid. A look-ahead heuristic for scheduling jobs with release dates on a single machine. *Computers & operations research*, 1994.

[38] R. Mayrhofer, H. Radi, and A. Ferscha. Recognizing and predicting context by learning from user behavior. In *The International Conference On Advances in Mobile Multimedia*, 2003.

[39] M. J. Neely. Delay-based network utility maximization. In *Proc. INFOCOM*, 2010.

[40] A. J. Nicholson and B. D. Noble. Breadcrumbs: forecasting mobile connectivity. In *International Conference on Mobile Computing and Networking*, 2008.

[41] V. N. Padmanabhan and J. C. Mogul. Using predictive prefetching to improve world wide web latency. *ACM SIGCOMM Computer Communication Review*, 1996.

[42] M. Palmer and S. B. Zdonik. *Fido: A cache that learns to fetch*. Brown University, Department of Computer Science, 1991.

[43] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka. *Informed prefetching and caching*. ACM, 1995.

[44] B. Sadiq and G. De Veciana. Throughput optimality of delay-driven maxweight scheduler for a wireless system with flow dynamics. In *Annual Allerton Conference on Communication, Control, and Computing*, 2009.

[45] S. Shakkottai and A. L. Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Translations of the American Mathematical Society-Series 2*, 2002.

[46] S. Sigg. *Development of a novel context prediction algorithm and analysis of context prediction schemes*. Kassel University Press, 2008.

[47] J. Spencer, M. Sudan, and K. Xu. Queueing with future information. *arXiv:1211.0618*, 2012.

[48] J. Tadrous, A. Eryilmaz, and H. El Gamal. Proactive data download and user demand shaping for data networks. *under submission*.

[49] J. Tadrous, A. Eryilmaz, and H. El Gamal. Proactive resource allocation: harnessing the diversity and multicast gains. *IEEE Trans. Information Theory*, 2013.

[50] V. S. Tseng and K. W. Lin. Efficient mining and prediction of user behavior patterns in mobile web systems. *Information and Software Technology*, 2006.

[51] Y. Wang, X. Liu, A. Nicoara, T.-A. Lin, and C.-H. Hsu. Smarttransfer: transferring your mobile multimedia contents at the right time. In *NOSSDAV*, 2012.

[52] T. Watson. Application design for wireless computing. In *Mobile Computing*. Springer, 1996.

[53] D. Wischik, M. Handley, and M. B. Braun. The resource pooling principle. *ACM SIGCOMM Computer Communication Review*, 2008.

[54] Y. Xu, M. Lin, H. Lu, G. Cardone, N. Lane, Z. Chen, A. Campbell, and T. Choudhury. Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In *Proc. ISWC*, 2013.

[55] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey. Bobtail: avoiding long tails in the cloud. In *Proc. USENIX Conference on Networked Systems Design and Implementation*, 2013.

[56] H. Yu and G. Kedem. Dram-page based prediction and prefetching. In *IEEE Computer Design*, 2000.

[57] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz. Detail: reducing the flow completion time tail in datacenter networks. *ACM SIGCOMM Computer Communication Review*, 2012.