1-1-2010

Filter Scheduling Function Model In Internet Server: Resource Configuration, Performance Evaluation And Optimal Scheduling

Minghua Xu
Wayne State University

FILTER SCHEDULING FUNCTION MODEL IN INTERNET SERVER: RESOURCE CONFIGURATION, PERFORMANCE EVALUATION AND OPTIMAL SCHEDULING

by

MINGHUA XU

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2010

MAJOR: COMPUTER ENGINEERING

Approved by:

____
Advisor   Date

____

____

____

____

ACKNOWLEDGEMENTS

I am grateful to my advisor, Dr. Cheng-Zhong Xu, for his encouragement, patience, persistence and invaluable guidance during these challenging years. It is an honor for me to have Dr. Nabil Sarhan, Dr. Jiang Song and Dr. Weisong Shi as my committee members, and to have their suggestions throughout this work. I am also thankful to friends in CIC Lab, Xilang Zhong, Jianbin Wei, and Jiayu Gong.

Finally I would like to thank my parents and my family for their sacrifices and unconditional support for my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The resources in the Internet include network bandwidth, CPUs, storage spaces, power and many other software or hardwares. How to use these limited resource to serve virtually unlimited user population posted a unique challenge for service providers. Multiple researches were carried out to study uniqueness of Internet traffics and Internet server scheduling policies with different queueing models. This chapter first defines the research direction for this thesis, and with introduction of unique background under Internet environment. Second, the major contributions from our research will be outlined, with highlights of each subsequent chapters.

1.1 Motivation and Problem Statement

Scheduling is one of the most important aspect in various activities during human history. One story dated back 3000 years ago for a smart general to use scheduling to win the horsing racing. While the scheduling in modern computer system is not as fancy as horse racing, it certainly has more financial stakes. The basic concepts are optimization between the order of services, quality of service (QoS) and constrained capacity. The quality measurements include system reliability, scalability, responsive time (Sojourn time), waiting time etc. Traditionally in Computer science and engineering, scheduling is studied with queueing model. Queueing model enables mathematical analysis of several related processes, including incoming traffic (arriving process), queue waiting, and the population actively in the service. It enabled the calculation and derivation of metrics such as average waiting time, queue length, probability of overflow, probability of starvation, etc.

A large body of work in the past ten years in Internet performance evaluation was stimulated by the discovery of self-similarity and long-range dependency (LRD) in the traffic[41]. The Poisson assumption in classical queueing model has been proved to be inadequate[55]. A lot of alternative models have been proposed to capture the correlation structures in the input traffic. However, due to the complicated nature of Internet traffic and diversified Internet applications, none of the model is entirely satisfactory. It is still in argument of whether researchers should go back to Poisson models to in small time-scale assumptions[34].

The classical queueing models assume renewal typed input traffic, and clearly is incapable to handle traffics with strong correlations. Numerous researches have extended queueing theorems to enable the correlated input traffic, such as Quasi Birth-and-Death model (QBD), Markovian Modulated Poisson Process (MMPP) queueing model, and more recently, LRD/GI/1 models. A lot of meaningful results have drawn from these models, such as empirical result of input-output relationship of second-order statistics from Li and Hwang [43, 44], negative impact of Hurst parameter in LRD traffic [52], and adaptiveness of queueing response to different GI and LRD in LRD/GI/1 queueing models [79].

In summary, these results are insightful but limited, for example, the study of negative impact of Hurst parameter ignored the service size distribution (assumed constant in [52]), or empirical, such as Li and Hwang were largely using simulation and numerical methods to study second-order in/out relationship in queueing server, or limited as even for excellent work like [79] only studied two LRD models, M/G/∞ and Fractional Gaussian Noise (FGN). As queueing model becomes more complex, its analytical solubility suffers dramatically. Results from these models are often referred as steady-state solution under assumption of infinite time horizon or buffer size. Analytically, the solution for queueing model is very hard to derive, even for simple low order Markovian modulated Poisson process. In practice, scalability of Internet server is still derived by stress-testing with application-specific benchmarks or synthetic workload [51]. These empirical studies have developed plentiful practical knowledge about scalable Internet servers. However, they are insufficient for the study of the impact of general Internet traffic, particularly its inherent bursty traffic patterns and auto-correlations on the server performance.

As consequence of complication in Internet traffic, A number of practically important questions are remain unresolved or partially unresolved, which include,

1. What resource capacity should a server be configured for a requirement of certain levels of QoS?

2. What level of QoS can a server with certain resource capacity support?

3. How to provide QoS guarantees by optimizing server scheduler?

4. What is the impact of LRD in servers?

In this thesis, we take an approach by constructing a decay function model for time varying request scheduling. The model assumes a fluid or random pulse modulated request arrival process. Each request has a randomly distributed size, and is scheduled by a decay function in the server. The decay function determines the amount of resource allocated at each scheduling epoch. The server utilization with given capacity in the server is the accumulation of individual allocations of all active requests. Decay function model reflects the relationship among service time of request, request size, autocorrelation in the traffic, and server workload. The first definition of decay function model is in [80].

From signal processing point of view, the proposed decay function model can be interpreted as filter, or referred as filtering function model. It models a server as a filter that transforms incoming traffic process to a process of server utilization, or server workload, with decay function as the transformer. Because there are rich theories of filter design in digital signal processing (DSP), the decay function model open a new path to study problems stated above.

In the time domain, the filtering relationship is charactered by a convolution operation $*$ (see 1.1) between scheduling function $h(t)$ and compound input traffic process $\tilde{w}(t)$. The relationship is contained naturally in the frequency domain as product of power spectral density (PSD) functions (1.2).

$$l(t) = \tilde{w}(t) * h(t), \tag{1.1}$$

where l(t) is the server workload at time t.

$$P_l(f) = P_{\tilde{w}}(f) \cdot |H(f)|^2, \tag{1.2}$$

where $P_l(f)$ is the PSD function.

Above derivation is under homogeneous assumption for $h(\cdot)$. When assumption individual requests r_k is served by $h_k(\cdot)$, a more general form of server load can be derived as following format.

$$P_l(f) = \{\lambda[\overline{w^2|H_k(f)|^2} - \overline{w}^2\overline{|H(f)|}^2]\} + \overline{w}^2 PSD_r(f)|H(f)|^2 \tag{1.3}$$

for server workload process $l(t)$, $P_{\tilde{w}}(f)$ is PSD for compound input process $\tilde{w}(t)$, $H(f)$ is Fourier transform of scheduling function $h(t)$, and f is the frequency variable.

Specifically, decay function model brings three immediate advantages. First, most of newly discovered second-order statistics, such as LRD, self-similarity, and more recent, time scaling [84, 19, 5] of Internet traffic are defined or studied in the frequency domain with Fourier or Wavelet transform. Decay function model makes it possible to apply these frequency-domain results directly to server performance modeling and optimization. In contrast, queueing analysis requires the conversion of frequency domain knowledge to the time domain. Second, frequency domain filter models provide a fresh look into the problem of server performance optimization. The server is modeled as a finite impulse response (FIR) filter. Filter theories are well defined and studied in DSP area. Third, the filter model is based on statistical signature of random processes — PSD function and autocorrelation function (ACF), instead of random processes themselves. This broadens the applicability of the model, and also make second-order statistics of Internet server analytically solvable with given second-order statistics from input traffic.

Using the decay function model, the Internet server is a real-time traffic processor. At any given time and with a wide-sense stationary process as input, we can derive the utilization process with convolution in time domain or point-wise multiplication in frequency domain. We proved that Internet server can be viewed as a non-high-pass filter. This extends the transitional low-pass filtering view of the sever. A number of properties, such as trade offs in time-scale — lobe-width relationship and lower bounds for lobe-width, lower-bounds for Accumulative Power Function (APF) are defined and characterized in the model.

With the model, we also capable of deriving the optimality of server schedulers. Traditionally, the scheduler performance is evaluated as fairness or effectiveness for individual requests (such as Lottery scheduler). Our model evaluates the optimality for scheduler from QoS point of view of the Internet server, which is in the sense of minimizing variance of the server utilization process, or the average power in server utilization process. It also provides a view of impact from scheduler to different component in the traffic from frequency domain, and from traffic correlation and sizing factors respectively. We show that queueing delay has effect on smoothing traffic from correlation (or PSD) of traffic point of view, it inadvertently increases the power of workload contributed by request sizing factor. As for sizing factor, it proofs the optimality of SRPT scheduler power spectrum of workload process from frequency domain.

For highly correlated Internet traffic, the optimality is proofed with decay function model to be convexity in the function. This applies to identical inter-independent traffic models such as Poisson, and traffic with monotonically decreasing PSD functions which applies to most of traffic processes with asymptotic second-order self-similarities.

Given the existence of optimal schedulers, algorithms can be constructed to solve the equations, and which in turn, give the optimal design of schedulers in theory. We further demonstrate its usefulness by adapting it to popular generalized processor sharing scheduler. The result [82, 81] shows great improvement in performance.

The other application of utilizing optimal schedulers is to study the capacity planning in the server, or QoS provisioning in the server. Such as, what amount of capacity is needed for a given service deadline and input traffic? or what kind of service deadline can server guarantee with capacity and input traffic? Several empirical formulas are derived in this research to address questions like above using general inequalities in probability.

In the ultimate, decay function model provides a nature platform to study LRD phenomena in the traffic, because LRD, as pointed out, is by itself defined with second-order statistics. The performance boundaries, both upper and lower, for asymptotically second-order self-similar traffic are derived here.

Decay function model under more general condition in time-variant variation and service differentiation environment is also valid and with meaningful optimal results for scheduling and performance evaluation [86].

1.2 Background

This section will highlight recent advancements in related research areas.

1.2.1 Internet Traffic Engineering

There are two major discoveries in Internet traffic that have brought great attention to performance analysis.

Leland, Taqqu, Willinger, and Wilson [41] observed that, over long time-scales, Internet traffic is self-similar. Self-similar is a fractal structure discovered in the nature. A lot of physical activities are shown to following a pattern, where a smaller portion of the object reassembles the larger pieces and repeating. one perfect example is the crystal structure in the snow flakes and other crystals. For traffics, the mathematical notation: Let $X(0, t]$ be the total amount of traffic passing through a particular point in the network in the time interval $(0, t]$. Write X for the overall traffic process. Define the speed-up version X_L by $X_L(0, t] = X(0, Lt]$. the rescaled process $aHX.a$ has approximately the same distribution whatever the value of a (for large a). The parameter H is the Hurst parameter and is good indication of self-similarity; they measured H and found it typically lay in (1/2 , 1). A self-similar process has bursts over all time-scales; i.e. the traffic does not become any smoother by averaging over any longer period of time. The common recognition is that the long-range dependency has strong negative impacts in the server performance.

From earlier work started in 1996, the size of static content in Internet has proved to be heavy tail distributed. Heavy tail is characterized as small percent of traffic contributes to most of workload in the Internet. In 2001, Cherkasova and Karlsson [14] revisited the 1996 invariants, showing several new trends in modern Web server workloads. Their work shows that 2-4% of files account for 90% of server requests. This level of skew (called concentration) is even more pronounced than claimed in 1996, when 10% of the files accounted for 90% of the activity.

1.2.2 Advancement in Scheduling Analysis

Three scheduling policies were studied widely in the computer science literatures: First-Come-First-Served (FCFS), Processor-Sharing (PS) and Shortest-Remaining-Processing-Time (SRPT). These policies are some of the most commonly used policies in computer systems. FCFS is used at a packet level in network and some functional servers. PS, which is basis for time-sharing applications, is the most common scheduling policy in computer systems. SRPT is known to optimize the mean response time and has been studied for performance improvement in various types of applications [74, 75, 29], such as OS and Web.

While a lot of researches focusing on mean value analysis with above policies, for Internet server, it is desirable to have stable and controllable services guarantee to the end users. There are many practical reasons for people to do so. Such as user may pay incentives for better and faster services, or providers have large financial stake with some user group. The statistical guarantees to the service profile, and with bounded capacity is what we are addressing in this thesis. On the other hand, second-order statistics on server workload (utilization) are directly associated with practical questions, such as scalability, stability and even power consumptions of server. This is another question that we use a function analysis to show structural optimality of schedulers.

Another systemic approach used feedback control for server performance optimization [2, 46]. Unlike feed-forward approaches like queueing and decay function models, feed-back approaches regard the input traffic variation as disturbance to the server, and operate by regulating admission and resource allocation according to the deviation of measurement from desired performance. For this approach, there is a non-negligible dead time (or delay) from the time when a correction action is taken to the time its effect is observed, and the delay is quite long in comparison with the change of input traffic in a heavily loaded server. The dead time sets a fundamental limit to the stability of the feedback control approaches.

1.2.3 Time Function and Unified Scheduling Framework

We point out that the idea of representing scheduling policies as a time varied function is not totally new. Fong and Squillante proposed a time-function scheduling to treat priority of jobs in computer systems as a function of time [27]. It is a generalization of the linear time-dependent priority discipline in queueing theories in which the priority of each job increases (linearly) according to a per-class function of some measure of time and the job with the highest instantaneous priority in the queue is selected for execution at each scheduling epoch. In stead of algorithmic treatment of time-function scheduling, our decay function model facilitates the development of analytical fundamentals of the time function abstraction on Internet servers.

1.3 Contributions

We constructed a functional model for schedulers, and using this function as kernel for transforming Internet traffic to server utilization (or workload) processes. By studying this I/O relationship, we have following major discoveries.

On top of mean value analysis, we show that second-order statistics between traffic and server workload is a convolution by scheduling in time domain, and is multiplication in frequency domain. Further more, there are two major parts in the transformation:

First, the transformation decided by scheduling functions and statistics averages of request sizes and average of traffic process. This part contributes to workload evenly allocated in each frequency bandwidth, and is minimized when scheduling function is homogeneous, and when scheduling function is minimizing

$$Min\{\overline{|H_k(f)|^2} - |\overline{H_k(f)}|^2\}. \tag{1.4}$$

This constraint optimization can be solved under homogeneous scheduling assumption. However, condition 1.4 is especially useful in measuring the impacts of queueing delay impact to second-order statistics of the server workload process. It shows that delaying smaller tasks tends to increase condition 1.4 and in the mean time cause decrease to 1.5. which explains the degrading performance of size oriented scheduler when sizing factor becomes less dominate as request size is light-tail distributed [29]. Because in heavy-tail distributed traffic, the relevance of decreasing in $|\overline{H_k(f)}|^2$ for sizing factor is getting smaller as variance in traffic increases (note that $\overline{w^2} - \overline{w}^2$ is variance of the traffic).

Second, a scheduling policy that minimize the product with square of power spectrum of the traffic.

$$Min\{\overline{w}^2 PSD_r(f)|H(f)|^2\}. \tag{1.5}$$

For this we proof that when power spectrum of incoming traffic is monotonically decreasing, a convex structured traffic will minimize the variance of the traffic process.

Of course, afore mentioned optimality is under the constraint

$$\sum h_k(t)_t^{t_d} = 1, \tag{1.6}$$

where t_d is the predefined service deadline. Of course with different t_d. Given the trade off in server capacity and service deadline, the exact optimal solution for scheduler may differ. But the convex structure in scheduler for highly correlated traffic is invariant.

As complimentary to traditional mean value analysis, decay function model can be easily used for capacity configuration with guarantee in responsive time. Decay function builds response time in the model, and using capacity/workload relationship to do predication in the second order. The entire analysis is independent from underlying distribution, which is a big loosen in requirements in comparing to asymptotic study of tails for buffer size or waiting time in queueing models. a very general bounds can be derived for capacity planning by using the general Chebschev inequality . If distribution information is available, the capacity estimation can be much more accurate. In unimodal distributed systems, the bounds can be tightened with VysochanskiPetunin inequality [72]. The relationship between scheduling deadline t_d and first-order/second-order statistics can also be used for predicting server SLA. All these results can be extended from mono-policy application to server with multiple SLA and different patterned services.

The principle of scheduling optimality is demonstrated by applying conditions to general processor sharing policy. The simulation result shows great improvements. If statistics of incoming traffic is known when doing scheduling, algorithms are also analyzed to derive the optimal schedulers. On the other hand, using equations 1.5 and 1.4, the power spectrum from traffic can also be analyzed for its impact to power spectrum of the server. Such as LRD traffic shows shift of power to low frequency band, where the main lobe of the scheduling function presides. The overlapping between traffic power and scheduler's main lobe in low frequency indicates the negative impact of LRD to server performance.

1.4 Outline of the Dissertation

The major framework of this thesis is surrounding the decay function scheduling model. As seen in above reviews in Section 1.2, the schedulers were traditionally treated as an algorithm, or policy. The mathematical representation of scheduling is often built into queueing analysis, which separates scheduling from the modeling itself. every new scheduling policy, a new analytical method is needed to carry out study. More than often, the policy is too hard for the theoretical analysis in the queueing framework.

1.4.1 The Decay Function Model

Chapter 3 gives the formal definition of the decay function model. The model is composed of two parts, first, the traffic model, and second, the scheduling and system model. We will derive the properties of the system in our modeling frame work in both time-domain and frequency-domain. we will also illustrate the samples of traffics and scheduling that can be represented in the framework. Model is built for both compounded traffic and traffic modeled as random point processes.

1.4.2 Optimality of the scheduler

One of major goal of this research is to review how does correlation in the traffic impacting the system performance in engineering point of view. Without knowing the optimal format of scheduling against different correlation, the above question is very hard to formulate and to solve. In the thesis, the optimal structure of scheduler under the monotonically decreasing power spectrum is derived. The monotonically decreasing power is popular across the Internet domain. We shown that the optimal structure of scheduler for that type of traffic should be convex. The simulation proves the derivation, when we simply compare the performance of system under a convex scheduler and a random scheduler. The simulation also shown that convex scheduler can outperform PS or SRPT scheduler depending given different traffic setup. Based on the concept, a modified version of PS can easily out-perform

GPS scheduler, when the modified version of PS basically overcome the randomness of PS scheduler, by making scheduling function approaching uniform function. and uniform function is actually optimal for independent traffics. Algorithms are proposed in this chapter for theoretical solutions to each optimal conditions, with simulations demonstrate the optimality of the schedulers.

On the other track, the model from Chapter 3 indicates that influence of scheduler to performance is divided into two major components (equations 1.5 and 1.4). When distribution of sizing factor is also considered (by further breaking down incoming traffics to random modulated point process), the scheduler biased for small jobs (SRPT) can perform very well in certain conditions. Our thesis shows that SRPT actually tends to dominantly minimize the sizing factor condition 1.4. On the other hand, when overload happens, FCFS tends to deliver the worse in sizing condition 1.4. The queueing delay, or waiting for request, or very small initial allocation of resources to the request, has the impact on the server by increasing the server workload spectrum in sizing factor 1.4. To translate into another words, the delay will reduce the power from second term 1.5, but at the cost of increasing 1.4.

1.4.3 Application of the model

Finally, the thesis will bring the focus back to the application of model in answering real-life questions, such as what allocation is needed to guarantee service in a deadline, or vice versa, what kind of responsive time can a fixed capacity server provide. The boundaries based on chebshev's and tighter for unimodal is proposed with easy calculation and well defined measure inputs for calculation. The optimal theories can be used to derive new type of schedulers. The improvement to GPS scheduler is proposed in Chapter 6. Further, using the model to analyze LRD traffics, and using the model to measure practical schedulers are also characterized in the chapter.

1.5 Chapter summary

This chapter has summarized the motivation, problem statements research backgrounds and major results. The scheduling in the server (leave along Internet) has been well studied in the last several decades. The approach has mostly been starting from a queueing model, and built the scheduling in the solution process of the model. The approach is very elegant, and produced large amount of meaningful and useful results. However, the solution of queueing model under different scheduling policies can each become an entire chapter. The complexity becomes major hurdle in the usefulness of the results. As we have pointed out, influence to Internet server is major coming from two sources, the variation of the request size, and the correlated arrival process. Many researches carried out before mainly focusing on one of the two factors. Or dropped one by using over-simplified model. such as commonly assumed Poisson distribution eliminates the signature of correlation in the traffic. and constant service time partially hides any sizing factor in the performance evaluation. These researches are summarized in the next chapter, and subsequent chapters will be using decay function model to explicitly show the impact from sizes and impact from correlations.

Chapter 2

Related Work

The origin of problem starts from Internet traffic characterization. It has pointed out that Internet traffic possesses a salient feature of strong correlation — Long-range dependency (LRD) [34, 1, 18, 33]. Internet has created a complex environment where millions of various types of servers and computers inter-connected with each other. The complexity is high in both time and aggregation levels. In this environment, the traditional renewal random process, such as Poisson process, is incapable of modeling the highly complicated Internet traffic and server processes.

In the meantime, Internet applications evolved from best-effort service models to time-constraint and mission critical services, such as online payments, banking, trading and multimedia streaming. These applications pose new challenges as they demand guaranteed service level to end users with big financial incentives on the hook as well. Design of Internet servers is required to provide reliable, scalable and predictable Quality of Services (QoS) for the complex traffic processes pointed out in traffic characterization.

There are several approaches that are focusing on above challenges, including building new traffic models that can mimic complicated correlations, extending queueing models with heavy tails service time, classification of scheduling policies and adopting control theorem for server performance evaluation. Some mathematical backgrounds are also introduced in this chapter for Fourier transform and its properties.

2.1 Advancing in Internet Traffic Engineering

Workload characterization and modeling are fundamental activities in the design and evaluation of performance of web systems. The study of Internet traffic can be summarized in two sub-groups, the first group focusing on study the size distribution of Internet server. The earlier work started in 1996, where the focus is primary on the static web content. With the advancement of Internet applications, the latest studies are almost entirely focused on the dynamic content in the Internet. On the other hand, the Internet traffic has been known to have strong correlations. Especially, the Internet traffic has been proved to have long-range dependency (LRD).

The activities on Internet is growing rapidly. Odlyzko [53] observed that the growth of Internet traffic would double every two years (Moores Law). This observation may suggest that the underlying measurements in Internet traffic could have changed dramastically in the short history of Internet era. The 1996 study of Web server workloads [1] involved six Web sites with substantially different levels of server activity. Nevertheless, all of the Web sites exhibited similar workload characteristics. The result shows that the nature of Internet applications have more influences on traffic, rather than sheer volume itself. The Internet application has transited from static content based pages to dynamic, and interactive application serving both financial and time critical functionalities.

The sizes of HTTP requests have been increasing based on evolution of web traffic from 1995 to 2003 [32], while the sizes of HTTP responses have been decreasing. In the mean time, Web usage by both content providers and Web clients has significantly evolved with advances in technologies, such as such as persistent connections, server load balancing, and content distribution networks etc. 1996 invariants for Web traffic was revisited in 2001 [14] It shows that 2-4% of files account for 90% of server requests. This level of skew is more pronounced than claimed in 1996, when 10% of the files accounted for 90% of the activity. This may be attributed by the change in Internet service pattern, or in the difference between the class of server access logs, which make result inconclusive. Such as similiar study [7] compares measurements of workloads obtained from the same server at Boston University between 1995 and 1998, and found that document size distributions kept the same while the distribution of file popularity changed. A detailed survey of Web workload characterization for Web clients, servers, and proxies is provided in [58].

On the other hand, there was recent trend to study the service time in the dynamic content web sites. The conclusion from the study shows that there was no direct link between the statistics of content size (transfer size), rather, the service time itself is also showing a heavy-tailed properties. The observation for commecial website traffic shows that traffic from e-commerce is highly busty in nature, much more than previously analyzed web traffic. Very high self-similarity was found in the traffic. The response time is heavy-tailed. Further, it breaks the traditionally believed link between responsive time and transfer sizes, it points out that the the responsive time is more related to the processing time in the backend server, rather than from the transfer sizes.

Leland, Taqqu, Willinger, and Wilson [41] observed that, over long time-scales, Internet traffic is self-similar. Self-similar is a fractical structure discovered in the nature. A lot of physical activities are shown to following a pattern, where a smaller portion of the object reassembles the larger pieces and repeating. one perfect example is the crystal structure in the snow flakes and other crystals. For traffics, the mathematical notation: Let $X(0, t]$ be the total amount of traffic passing through a particular point in the network in the time interval $(0, t]$. Write X for the overall traffic process. Define the speed-up version X_L by $X_L(0, t] = X(0, Lt]$. the rescaled process $aHX.a$ has approximately the same distribution whatever the value of a (for large a). The parameter H is the Hurst parameter and is good indication of self-similarity; they measured H and found it typically lay in (0 , 1). A self-similar process has bursts over all time-scales; i.e. the traffic does not become any smoother by averaging over any longer period of time.

2.2 Queueing Theorem

Queueing network (QN) models are widely used to address similar issues in packet scheduling in networks and job scheduling in closed computer systems [64]. Queueing theories are based on an assumption of input renewal or Markovian processes. There were early studies that treated Internet requests as packets in routers or jobs in computer systems and simply applied the QN models for performance evaluation of Internet servers; see [51, 68] for examples. But the model applicability and accuracy were found very limited because recent Internet workload characterization studies [1, 18, 33] all pointed to long-range dependency and heavy tail as inherent properties of Internet traffic. as a result, a number of researches are carried out to extend classical queueing theorems to handle these complicated situations.

2.2.1 Hierarchical Queueing Models

One approach focuses on extending the simple renewal model by incorporating a modulate process. Examples of the models include Markov Modulated Poisson Process (MMPP) [42, 43, 44, 6] and Quasi Birth and Death (QBD) [39] process. For example, Li and Hwang [43, 44] pointed out that queueing servers should be only sensitive to low frequency components in the power spectral density function of the input traffic. These models can be solved with mathematical deductions or numerical methods. But their solvability is limited to low order Markovian processes, due to the computation complexity. Moreover, the low order Markovian process can only model traffic with short range dependency.

To overcome the limitations of Markovian processes, there are new models based on LRD input traffic, such as $M/G/\infty/G/1$ queue model, and $FGN/G/1$ model [20, 63, 25, 52, 79]. They can only be analyzed asymptotically with Large Deviation (LD) [21] techniques. For example, Norros [52] characterized negative impacts of self-similarity of input traffic on storage servers; Xia and Liu [79] modeled the server performance with different service time distributions under the LRD traffic generated by $M/G/\infty$ and FGN models. However, it is also shown that the effect of LRD on server performance is ambiguous [62]. LD techniques assume the steady state of a process can be bounded by "envelope process" [10, 30, 35, 36]. The envelop process in a simpler analytical format makes it easier to derive performance bounds. All these results are insightful, but only give information about system's steady state performance under various assumptions, such as traffic models, queueing disciplines, infinite buffer, and infinite time horizon.

2.2.2 Network Calculus

We also note that there are established frameworks for modeling of highly variable traffic on network routers [54, 65, 40]. To handle the high-variability of Internet traffic, the frameworks assume the traffic smoother model (or shaper), which essentially reduces the burst of Internet traffic and enables subsequent rate-based scheduling/traffic analysis. To further compensate the high variance of Internet traffic, traditional single-rate leaky-bucket scheduling is extended to multiple leaky-bucket scheduling. These techniques for handling the traffic variance are not applicable to end-server modeling for two main reasons. First, routers take packets from the same connection as input, which can be buffered for smoothing the downstream traffic. By contrast, Internet servers take client request as a scheduling unit and the processing cost for each individual request cannot be smoothed by any preprocessing. Second, from QoS point of view, drop or loss a packet in routers may cause an intolerable loss (multimedia traffic is an exception because it can tolerate a loss rate up to $10^{-5} \sim 10^{-7}$ [65]). However, scheduling on Internet servers can have wide choices of adaptations for different quality levels. For example, a multimedia server has choices of different compression ratio, different encoding to balance the resources needs and QoS.

2.3 Scheduling Policies and Advancements in Internet

The study of computer scheduling policies has long history with vast literature of analytic results. Though simple policies like FCFS and PS are traditionally the most common policies used to model scheduling in computer systems. In recent years, there is a resurgent study in the field, this resurgence was led by researches from [74, 75] on group studies of scheduling performance. These researches focus on design schedulers based on priority-based policies. There are many ways that priority-based policies can assign priorities. Commonly, users are willing to pay money in order to receive high priority service and, thus, spend less time queueing. Other times, the goal may be to assign priorities in a way that minimizes some cost function (such as mean response time) of the queue. In the second scenario, it is often beneficial to give priority to small job sizes. In this section we will mainly introduce the later, which is commonly referred as Shortest-remaining-processing-time policy.. Under SRPT, at every instance, the job with the smallest remaining service time is scheduled. In this way SRPT greedily tries to minimize the number in system by always working on the job that can be finished the quickest. In the preempt resume setting, this greedy approach is good enough to minimize the number in the system, and thus $E[response\_time]$, regardless of the arrival and service processes because any scheduling decision can be reversed without penalty if a more attractive (smaller) candidate arrives. Because of this optimality, SRPT has received a large amount of attention in the literature.

in [74], mean responsive time from SRPT under $M/G/1$ traffic is compared to the results from processor sharing. It shows that previous conclusion ”unfairness” for SRPT is not justified — SRPT doesn’t introduce great deal of penalty to large jobs are intuitively envisioned before [70]. Further, average responsive time from SRPT is much smaller than that from PS schedulers, especially when server is approaching overloaded situation.

In [75], authors define SMART schedulers, which includes Shortest remaining processing time (SRPT) and preemptive shortest job first (PSJF). The upper and lower bounds on mean response time were derived and schedulers in the group group of schedulers have very similar mean response times. The importance of SMART scheduler is that the mean responsive time from schedulers are insensitive to the variation of the request sizes. On the other hand, the complementary research [29] shows the sensitivities of SRPT and PS scheduling to selected characteristics of the arrival process and job size distribution.

Because these results for SRPT based scheduling policy, there are applications using SRPT in different areas, such as web servers [31, 48] and network routers [59]. It also triggers a wave of simulation for traffics more general than $M/G/1$ [29], these results further reveals that even though SRPT is indeed optimal in mean responsive time. Its performance in ”fairness” changes for traffics with light tail vs. heavy tail, and for Poisson traffic vs. LRD traffics. These results call for further advancements in theoretical analysis of scheduling policies. In this thesis, we show that two factors in the traffic, request sizes and traffic correlation, and how these two factors each interact with schedulers to influence the second-order statistics of server workload process. With the model in this thesis, the details of factors deciding optimality of schedulers are revealed.

2.4 Signal Processing Techniques in Server Models

Signal processing techniques have been used in popularity in characterizing traffic, such as time-scaling, LRD and self-similarity. By studying the PSD functions for Internet traffic traces, one observation is monotonically decreasing PSD function from low to high frequency in many traffic traces [28, 5, 8, 49]. Given above progresses in traffic anagnorisis, little work exists for server performance modeling and optimization in the same track. Li and Hwang [43, 44] studied frequency responses of FCFS queueing servers using simulations for MMPP traffic. Their results were mainly based on numerical methods and at lower order of MMPP traffic model. In communication area, such as [37], authors applied the filter concept in wireless communication that tries to reduce the power consumption of devices. Their results is limited to a special case of independent input traffic.

2.5 Feedback Control Approach

Given the challenge of Internet server providing stable processing capabilities to end user requests. Using feedback control to maintain system stability and optimality becomes another choice. Since Abdelzaher and Lu [4, 2, 46] introduced the concept to Internet server QoS management, there are several extensions to the framework, such as, using a simplified queueing model as feed-forward predictor to help maintain stability and compensate for errors in the control, extending operating system kernel to enable the control mechanism and real time measurements, introducing request admission control mechanism to handle unexpected situations, and adopting neural network to construct self-adaptive server system for variability of Internet traffics. The unique challenge in this approach is the stability of the system, and non-negligible delay between correction action and effect can be observed. In the bursty traffic of Internet, it might be simply too late for feedback to response and taking corresponding adjustment to the fast variation in the traffic.

Chapter 3

The Decay Function Model

The high variable nature of Internet traffic imposed great challenges for Internet service providers to design the system architects to meet the quality demand by the users. It has been pointed out that the response time is an important quality measurement from end-user point of view about the service they received. The long delays increase user frustration and the abortion of transactions. Past surveys [12] show that the experience of users is under great influence of service delay, the sensitivity is graded in range of 5 seconds; and tolerance for delay increases if the page can be presented and loaded incrementally as opposed to present all the page once its entire content has been loaded.

The performance of an Internet server system depends in large part on the scheduling policy, which decides how and when the user requested services will be processed. Scheduling policies have been studied for may years in server operating and networking systems. Especially in recent years, the reveal of heavy tailed distribution in server request requirement (size), and long-range dependent (LRD) arrival pattern has triggered renewed interest in the performance evaluation in scheduling policies.

In this research, we are designing a model that can characterize the Internet server from its basic activity, that is the scheduling of resources for each individual request. The newly emerged Internet applications often requires bounded service deadline, or response time for each requests under finance-critical situations. In this chapter, we show that when assuming deadline constraint for each request, the Internet server can be modeled as filter-alike convolution relationship with compound traffic as input and server utilization process (or workload process) as output. In Section 3.1, we describe the steps of constructing decay function model. In Section 3.2, we show the simplification path of the model under constraint deadline assumption.

Analytically, two things make decay function model attractive, first, it converts rather complicated queueing relationship into well studied filtering alike transformation, second, we can study the general scheduling with relaxed traffic assumption but in definition of formalism.

3.1 Decay Function

Consider an Internet server that takes requests as an input process, passes them through scheduling — the decay function kernel, and generates a system load output process. Each incoming request needs to consume a certain amount of resource. We define the required resource amount as the request "size". An Internet request often needs to consume more than one type of resource (e.g. CPU time, memory space, network-IO bandwidth, disk-IO bandwidth, etc.). Since the server scalability is often limited by a bottleneck resource class, this paper focuses on the management of this critical resource. The request size is a generic index and can be interpreted as different measures with respect to different types

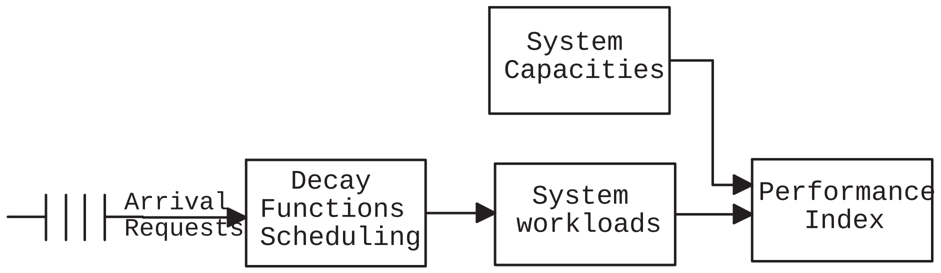


Figure 3.1: Illustration of the decay function model.

of resource. It is different from the size of its target file. In some cases, such as FTP and static content Web servers, the request size is proportional to the file size. For the system under consideration, we assume that the size distribution of requests is known. In a QoS-aware system, the size of a request can be derived from quality indices in different QoS dimensions [60].

We define the server capacity and server load accordingly. The output server load determines the level of QoS, in terms of the request completion time and rejection rate, for a given server capacity **c**. Associated with the server is a scalable region $[c_l, c_h]$ for each request, $0 \leq c_l < c_h < \mathbf{c}$, in which the service quality of the request increases with the amount of its allocated resource. If the maximum amount of resource c_h is allocated and the request can still not meet its quality requirements, the server performance will not be improved by simply increasing the resource quantity. c_l is the lower bound of allocated resource which is determined by the request minimum quality requirement. In a scalable system, it is always true that more allocated resource will lead to non-decreasing quality of service.

The load filtering model is a discrete time model, with t as sever scheduling epoch index, t_s as arrival time of a request, and t_d for the deadline of the request. As shown in Figure 3.1, the model consists of three major components: request incoming process, decay function scheduling and evaluation of system workload process. We model the request arrivals as a general fluid typed process:

$$\Big\{n(1), n(2), n(3), \ldots, n(t), \ldots\Big\}, \tag{3.1}$$

where $n(t)$ is the number of requests arrived at time $t, t = 1, 2, \ldots$. In the simplest case, $n(t)$ at different scheduling epoches can be i.i.d. random integers from a general probabilistic distribution. Taking the request size into the model, process (3.1) can be re-written as:

$$\Big\{\{w_i^t\}_{i=1,2,\ldots,n(t)}\Big\}_{t=0,1,\ldots}, \tag{3.2}$$

where w_i^t is the size of i^{th} request that arrived at time t.

Scheduling activities of a computer system are represented in an algorithmic manner. In this paper, we define a decay function to abstract the scheduling on the Internet server. It is defined as a relationship between time and resource allocation for each request. Formally, the decay function is a function of system time t, request arrival time t_s, request size w, and the current server workload $l(t)$. We denote it as $d\big(t, t_s, w, l(t)\big)$. Strictly speaking, the decay function should also depend on request deadline t_d. For tractability, we treat t_d as a model parameter, rather than a free variable.

In real Internet servers, the adaptation of scheduling to request size and server workload does not change with time. That is, $\partial d/\partial w$ and $\partial d/\partial l$ are functions independent of time t.

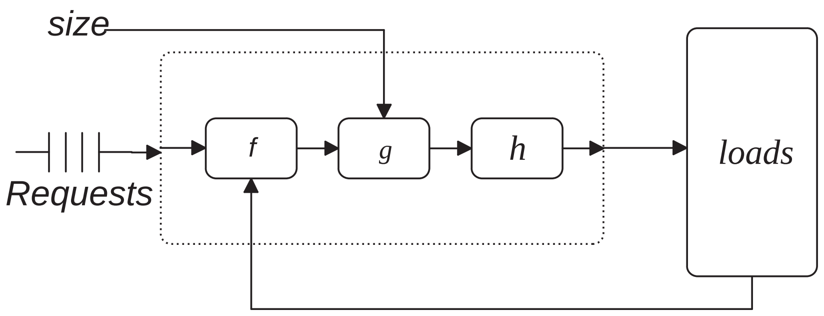


Figure 3.2: Decomposition of scheduling

Under this assumption, by the use of variable separation techniques the decay function can be rewritten as a three stepped process:

$$d\big(t, t_s, w, l(t)\big) = h(t, t_s)g(w)f\big(l(t)\big), \tag{3.3}$$

where $f(\cdot)$ measures the effects of workload on scheduling, $g(\cdot)$ represents the impact of request size w on scheduling, and $h(\cdot)$ is the function evolving with time to determine the resource allocation. This scheduling decomposition is illustrated in Figure 3.2. We assume that the scheduling is a causal activity. That is, for all $t < t_s$, $h(t, t_s) = 0$. It means that no resources will be reserved in advance for a request before it arrives.

From the definition of decay function, the amount of resource actually consumed by a request, must be equal to the request size w. In another words, the scheduler will always allocate enough amount of resources to a request by its specified deadline t_d. That is,

$$\begin{aligned} w &= \sum_{t=t_s}^{t_s+t_d} d\big(t, t_s, w, l(t)\big) \\ &= \sum_{t=t_s}^{t_s+t_d} h(t, t_s) g(w) f\big(l(t)\big) \end{aligned} \tag{3.4}$$

We refer to (3.4) as a *scheduling function*.

The server workload at time t is equal to the sum of all resources allocated to the requests that arrive before t. Thus,

$$\begin{aligned} l(t) &= \sum_{t_s=0}^{t} \sum_{k=1}^{n(t_s)} d\big(t, t_s, w_k, l(t)\big) \\ &= \sum_{t_s=0}^{t} \sum_{k=1}^{n(t_s)} h(t, t_s) g(w_k) f\big(l(t)\big) \end{aligned} \tag{3.5}$$

We refer to (3.5) as a *workload function*. The workload and scheduling functions together determine the dynamics of resource scheduling on the Internet server.

3.2 Filter Function Simplification

The decay function model defined above is applicable to any scheduling algorithms. To make the model tractable, we classify the scheduling algorithms by two dimensions: request size awareness and server load adaptation. Specifically,

1. Size-oblivious scheduling, if $g(w) = 1$. It means scheduling algorithms is unaware of resource demand of the requests.

2. Size-aware scheduling, if $g(w)$ is an explicit function of w. This is the class of scheduling that is able to allocate resources proportional to size of requests.

3. Non-adaptive scheduling, if $f\big(l(t)\big) = 1$. The resource allocation algorithm is independent of server load $l(t)$.

4. Adaptive scheduling, if $f\big(l(t)\big)$ is a non-degenerating function. That is, the scheduler will adapt its resource allocation policy to the change of server load.

Best-effort scheduling is an example of size oblivious scheduling. It is popular in today's desktop operating systems. Size aware scheduling like proportional scheduling, is discussed extensively in QoS researches, where the feature of "controllable" is emphasized. Adaptive scheduling is to allocate resource in response to the change of system utilization for improving the system throughput. Non-adaptive scheduling is often oriented to applications where the quality of service cannot be compromised. In this paper, we focus on size-aware and non-adaptive scheduling algorithms. In particular, we assume $g(w) = w$ for size-awareness. Recall that the request deadline t_d is considered as a predefined model parameter. It means the server can always finish a request in a fixed period of time. We hence refer to it as *fixed-time scheduling*. Fix-time scheduling decouples the deadline constraint from scheduling and simplifies the analysis of the decay function model.

We note that scheduling algorithms are normally time invariant. That is, $h(t, t_s) = h(t - t_s)$. Consequently, the workload function (3.5) in the case of fixed-time scheduling can be simplified as

$$\begin{aligned} l(t) &= \sum_{t_s=t-t_d}^{t} \sum_{k=1}^{n(t_s)} h(t-t_s) \cdot g(w_k) \\ &= \sum_{t_s=t-t_d}^{t} h(t-t_s) \cdot \sum_{k=1}^{n(t_s)} g(w_k) \\ &= \sum_{t_s=t-t_d}^{t} h(t-t_s) \cdot \tilde{w}(t_s), \end{aligned} \tag{3.6}$$

where

$$\tilde{w}(t_s) = \sum_{k=1}^{n(t_s)} g(w_k). \tag{3.7}$$

Given the arrival process in (3.2) and predefined impact of request size $g(w)$, the properties of the compounded process $\tilde{w}(t_s)$ are derivable. In the case that $n(t)$ is i.i.d. random number from a distribution, the size w is i.i.d. random number from another distribution, and $g(w) = w$, the compounded random process $\tilde{w}(t_s)$ is a simple random process following a distribution of *random sum* [26].

By introducing the convolution operator "$*$" on two vectors $a(n)$ and $b(n)$, we have

$$a(n) * b(n) = \sum_{m=-\infty}^{\infty} a(m)b(n-m).$$

It is known that $h(t-t_s) = 0$ when $t < t_s$ for causality consideration. Also, we note that no scheduling will be made before system starts running. That is, $h(t-t_s) = 0$ when $t_s < 0$.

As a result, the simplified workload function (3.6) can then be rewritten as a convolution:

$$l(t) = h(t) * \tilde{w}(t). \tag{3.8}$$

Equation (3.8) presents fixed-time scheduling on Internet servers as a perfect format of linear system model with a transfer function $h(\cdot)$. Meanwhile, for fixed-time scheduling, the scheduling function (3.4) can be simplified as:

$$\sum_{t=t_s}^{t_s+t_d} h(t-t_s)g(w) = w.$$

That is,

$$\sum_{t=0}^{t_d} h(t) = \frac{w}{g(w)}.$$

In the case of $g(w) = w$, the scheduling function becomes

$$\sum_{t=0}^{t_d} h(t) = 1. \tag{3.9}$$

According to theorems of random signal processing [50], when the input "signal" $\tilde{w}(t)$ is a *wide sense* (WSS) stationary—the mean of $\tilde{w}$ is constant and its autocorrelation function depends only on the time difference, the statistical relationships with respect to their mean, variance, autocorrelation and cross-correlation between input and output of (3.8) can be established. The assumption of WSS is quite general in the modeling of random signal. This assumption is also valid for Internet traffic. In [69], the authors proved that the Internet arrival process could be at least modeled as a phase-pieced WSS processes.

3.3 Filtering Analysis in Frequency Domain

In previous section, the server utilization process and input traffic process are coupled by a convolution filter. In signal processing theories, the filter satisfying condition (3.9) is referred to as Finite Impulse Response (FIR) filter. It is natural to study this type of filter in Frequency domain by using Fourier transform. This chapter will first convert convolution in (3.8) into frequency domain by using power spectral density (PSD) of input traffic and server utilization process, we then use this relationship to reveal some basic properties of scheduler as a filter. These properties include frequency selectiveness of the scheduler, adaptiveness of leakage of power to the service deadline, main-lobe and its lower bound under the service deadline constraint.

3.3.1 Fourier Transform of Scheduler

Because of the distinct environment of the Internet, we cannot follow the conventional path of filter design in DSP to construct the scheduling function $h(t)$. First, coefficients of the scheduler are exclusively non-negative. This restriction implies the scheduler is a non-high-pass filter. Second, in signal processing such as detection and modulation, the interesting region is often in narrow frequency band. In contrast, the power of Internet traffic usually spreads across entire frequency domain. This requires the server scheduler to be optimized globally. Third, Internet servers impose a capacity constraint on the scheduler, which DSP filters do not have.

We apply Fourier transform to the scheduling and workload functions. Let f be the angular frequency variable. The Fourier transform of the scheduler $h(t)$, denoted by $H(f)$, is

$$H(f) = \sum_{t} h(t)e^{-2jtf}. \tag{3.10}$$

Because of the symmetric property of the Fourier transform, we only need to consider the frequency variable f in $[0, \pi]$. According to the definition of Fourier transform, $H(f)$ has the following properties:

Theorem 3.1. *The server scheduler $H(f)$ has following properties:*

$$H(0) = 1 \tag{3.11}$$

$$0 \leq |H(1)| < 1 \tag{3.12}$$

Theorem 3.1 reveals that a scheduler is not a strictly low-pass filter as traditionally perceived. First, it does not guarantee $|H(\pi)| = 0$, as required by a low-pass filter. Second, some theoretically valid schedulers satisfy this requirement, but actually exhibit a "comb-alike" frequency shape. Figure 3.3 shows such an "edge scheduler" in both time and frequency domains, which allocates resource at the beginning and the end of the request lifespan. This property implies that the server performance are sensitive to low as well as high frequency traffic.

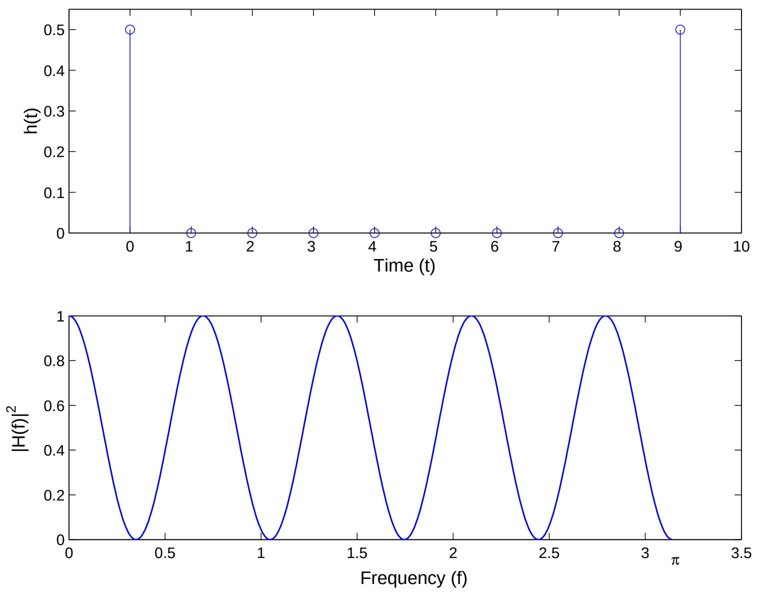


Figure 3.3: Edge scheduler and its Fourier transform.

3.4 PSD Functions and Adaptiveness of Scheduler

To study the impact of correction in the input traffic, we introduce Power Spectral Density (PSD) function, which corresponds to the autocorrelation function (ACF) in the time domain. For the input compounded process $\tilde{w}(t)$, the PSD function $P_{\tilde{w}}(f)$ is

$$P_{\tilde{w}}(f) = \sum_{\tau} R_{\tilde{w}}(\tau) e^{-2j\tau f}, \tag{3.13}$$

where $R_{\tilde{w}}(\tau) = E(\tilde{w}(t), \tilde{w}(t+\tau))$ is the ACF of $\tilde{w}(t)$. PSD function $P_l(f)$ for the workload process $l(t)$ can be defined in a similar way, which characterizes the distribution of average power in entire frequencies.

For a stationary scheduler and an WSS input process, the convolution relationship in the workload function (3.6) can be rewritten in the frequency domain as

$$P_l(f) = P_{\tilde{w}}(f) \cdot |H(f)|^2. \tag{3.14}$$

$|H(f)|^2$ is called energy density function (EDF) in this paper. EDF characterizes the filtering properties of the scheduler in frequency domain. In DSP terminologies, EDF determines the "power leakage" from PSD of input traffic process to PSD of system utilization process.

Because the optimality of the scheduler is defined across entire frequencies, we define a cumulative power function (CPF) for scheduler $h(t)$ as:

$$A_h(f) \quad = \quad \int_0^f |H(\omega)|^2 d\omega. \tag{3.15}$$

Similarly, the CPF for a random process $w(t)$ is defined as:

$$A_w(f) \quad = \quad \int_0^f P_w(\omega) d\omega. \tag{3.16}$$

Because EDF and PSD functions are non-negative, both $A_h(f)$ and $A_w(f)$ take the maximum value at $f = \pi$.

From Figure 3.3 we can see the EDF of the edge scheduler has a number of lobes. The first lobe in lower frequencies is called "main lobe" and the rest are "side-lobes." According to Theorem 3.1, the main lobe leaks lower-frequency power in the input traffic to server utilization process, and the side lobes determines the power leakage in higher-frequencies.

The main lobe often dominates the scheduler performance, because Internet traffic often has a majority of power in low frequencies. We establish a trade-off relationship between the main lobe size and service deadline in Theorem 3.2. It can be proved by the definition of Fourier transform.

Theorem 3.2. *Assume a scheduler h with service deadline t_d. We "expand" the scheduler in the time domain by scale a to form a new scheduler $h'(n) = h(n/a)/a$, the Fourier transform $H(f)$ of the new scheduler is "compacted" by scale a and the CPF $A(f)$ is reduced by scale a. That is,*

$$H'(f) = H(af), \tag{3.17}$$

$$A'(f) = \frac{1}{a}A(f). \tag{3.18}$$

Figure 3.4 illustrates the trade off relationship. The top line in Figure 3.4(a) shows a uniform scheduler with a service deadline of 10, in which $h(t) = 1/10$, for $t \in [0, 9]$. Another line shows an extension of the deadline to 20. Accordingly, the main lobe width of the scheduler with the extended deadline drops from $\pi/5$ to $\pi/10$, as shown in Figure 3.4(b). Figure 3.4(c) shows that its CFP value is reduced by approximately half in high frequencies.

Comparing the edge scheduler in Figure 3.3 and uniform scheduler in Figure 3.4, we can observe the difference between their main lobe widths ($\pi/9$ versus $\pi/5$ when $t_d = 10$). In fact, the main lobe width of a scheduler is determined by the scheduler structure and bounded by the main lobe width of edge scheduler.

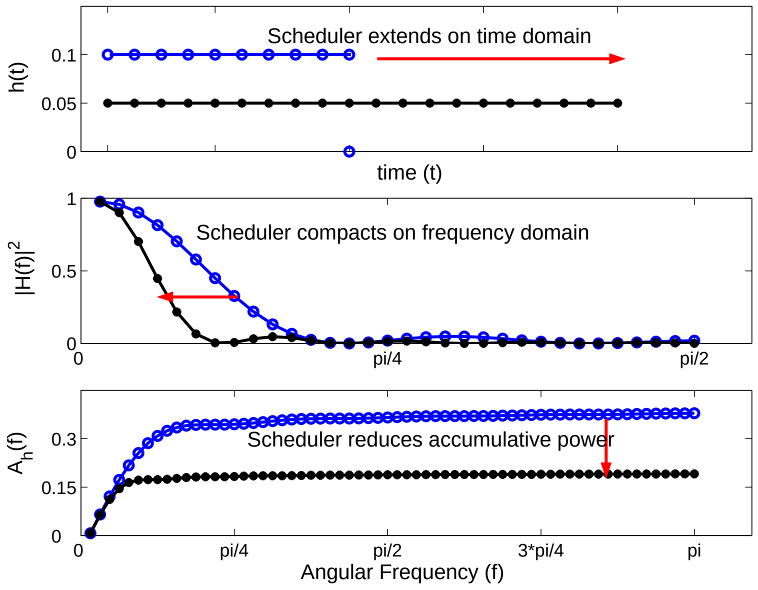


Figure 3.4: Trade-off between length of scheduler $h(t)$, width of EDS function $|H(f)|^2$ and value of CPF function $A_h(f)$

Theorem 3.3. *For any scheduler with a service deadline of t_d, edge scheduler has the narrowest main lobe and its width equals to $\frac{\pi}{t_d-1}$.*

Proof: From the definition of $H(f)$, we have the EDF

$$\begin{aligned} |H(f)|^2 &= \sum_{t=0}^{t_d-1} h(t)e^{jft} \sum_{t=0}^{t_d-1} h(t)e^{-jft} \\ &= \sum_{t_1=0}^{t_d-1} \sum_{t_2=0}^{t_d-1} h(t_1)h(t_2)e^{jf(t_1-t_2)} \\ &= \sum_{t_1=0}^{t_d-1} \sum_{t_2=0}^{t_d-1} h(t_1)h(t_2)cos\big((t_1-t_2)f\big). \end{aligned} \tag{3.19}$$

In (3.19), the maximum multiplier for frequency variable f in the $cos()$ function is t_d-1.

We know that EDF of edge scheduler is:

$$|H_e(f)|^2 = 2 + 2cos((t_d - 1)f). \tag{3.20}$$

Clearly, $|H_e(\frac{\pi}{t_d-1})|^2 = 0$ and the width of its mainlobe is $\frac{\pi}{t_d-1}$.

When $f \in [0, \frac{\pi}{t_d-1})$, $cos(nf)$ in (3.19) is monotonically decreasing for $n \in \{0, \ldots, t_d - 1\}$, so is $|H(f)^2|$ for general traffic.

When $f = \frac{\pi}{t_d-1}$, and $t_d > 2$,

$$\begin{aligned} |H(\frac{\pi}{t_d-1})|^2 &= \sum_{t_1=0}^{t_d-1}\sum_{t_2=0}^{t_d-1} h(t_1)h(t_2)cos\big((t_1-t_2)\frac{\pi}{t_d-1}\big) \\ &= \sum_{t_1=t_2} h(t_1)h(t_2) \\ &\quad + \sum_{t_1\neq t_2} h(t_1)h(t_2)cos\big((t_1-t_2)\frac{\pi}{t_d-1}\big). \end{aligned} \tag{3.21}$$

Because $cos()$ function is symmetric around $\pi/2$ when $f \in [0, \pi]$, and the number $\frac{t_1-t_2}{t_d-1}$ is anti-symmetric when $t_1, t_2 \in [1, t_d - 1]$, and $t_1 \neq t_2$, we have,

$$\sum_{t_1\neq t_2} h(t_1)h(t_2)cos\big((t_1-t_2)\frac{\pi}{t_d-1}\big) = 0. \tag{3.22}$$

This proves,

$$|H(\frac{\pi}{t_d-1})|^2 > |H_e(\frac{\pi}{t_d-1})|^2 = 0. \tag{3.23}$$

Equation (3.23) states that when an edge scheduler reaches its mainlobe, all other schedulers are still having value greater than zero, while decreasing from zero frequency. This proves edge scheduler has the narrowest mainlobe.QED

From Figures 3.3 and 3.4, we can also see a big difference between the side lobes of edge and uniform schedulers. It means that, other than main lobe, the scheduler structure also determines power-leakage in high frequencies. This will be discussed in the following sections.

3.5 Random Point Process and Decay Function Model

Because of the importance of responsive time, the majority of existing studies are focusing on the evaluation of scheduler by *mean* responsive time or sojourn time in the server. The well known results have been established for both size-oblivious, such as First-Come First-Serve (FCFS), Processor Sharing (PS) and size-based scheduling policies like Shortest-Remaining-Processing-Time (SRPT), and preemptive Shortest Job First (SJF). These researches have pointed out that size-based scheduling has great advantage over size-oblivious scheduling policies. In fact the SRPT scheduler is the optimal scheduler in minimizing the *mean* responsive time, the Least Attained Service (LAS) is the approximation for SRPT when size of jobs is unknown (size-oblivious). The researches by Harchol-Balter et al [74, 75] further point out that the starvation faced by size-based scheduling such as SRPT is not severe in the Internet server environments. On another hand, there are recent studies on the configurable scheduling policies in the server that can be tuned continuously to behave from FCFS to LAS (approximating SRPT).

The less attention has been given to the statistical guarantees of the response time. That is, other than the mean value of the responsive time, what is the probability for responsive time to fall within (or fail to meet) the predefined time range, for the class of the Internet services and level of quality of services. The solution to the responsive time with Queueing model is analytically complicated, and often involved multiple integration. Even the second-order statistics of responsive time is very hard to solve and few results can be drawn from the complex mathematical formulas. The result from previous sections provides a meaningful solution in second-order statistics, and will be greatly useful for optimality study in scheduling. It hides the dual factors of the traffic into a compounded process.

The dynamics of Internet traffic are coming from two factors, the service requirement (size) by the individual request (which is determined by the service provider) and the service arrival process, which is determined by the end-user activities. The sizing impact to server performance has been relatedly well studied in recent years [29, 74, 75]. The correlation impact is largely less known. The major results are still relying on simulation for particular given web traffic [29] — not to mention the combined analysis of dual factors with a model capture the abstraction of scheduling policies.

This section presents a *Decay function* model extension. with decay function character scheduling policies under one generic mathematical format. We apply it to analytically study the impacts from request size distribution and the correlation in arrival process. The optimality of scheduling policy considers the size distribution and autocorrelation in the traffic.

With more and more statistical evidence of long-range dependence observed in communication networks [63] and in Internet server [69], it has become increasingly important to understand the performance impact by traffic exhibiting long-range dependence structure. Previous work in this area has been mostly focused on the dependence structure while assuming deterministic service times. Two traffic input models, the $M/G/\infty$ model and Fractional Gaussian Noise (FGN) model, have been used extensively to model the long-range dependence structure. Studies using $M/G/\infty$ include [79], while studies using FGN input model include [20, 79, 63, 25] A number of work has been carried out by extending deterministic service time to more realistic assumption, such as subexponential service time in [79]. Due to mathematical difficulties, the results from above models assume the constant service time, which means the results are hardly applicable to general Internet server environments where service requirements are commonly described as heavy-tail distributed. Also, in these researches, it is unknown how scheduling affects asymptotic behavior of the queueing model.

3.5.1 Random Point Process

We assume that the job arrives on the server by following a random point process. The random point process is defined as t_k, where t_k means the arrival time of each individual request. To simplify the model, we assume the process follows the condition that no multiple requests at one single time. This is a reasonable assumption in reality, since all devices in Internet are operating on a discrete time clock, and critical devices, such as routers, and dispatchers are all working in sequential mode. Also, tightly associated with the time of the

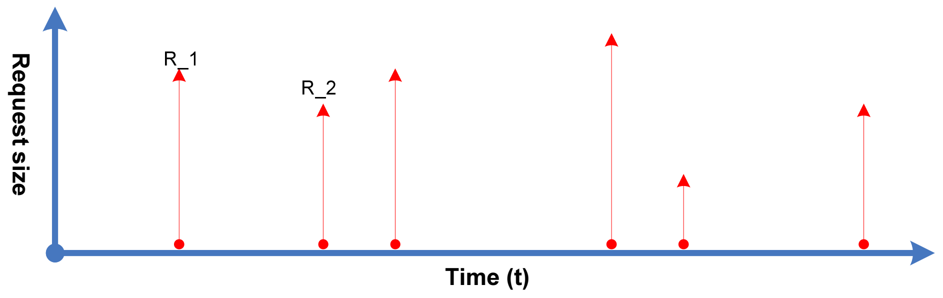


Figure 3.5: An illustration of marked random point process.

arrival is the size of individual request. with it, the overall traffic is modeled as "marked random point process". Figure 3.5 shows the sample of how this process looks like.

After each request comes onto the server, the scheduler will fire up and allocate resource to the request. The allocation is modeled as time function in our model. Assume for request arrives at t_k, the allocation for it is $h_k(t-t_k)$. Because the function h is describing scheduling of the server , it has to meet certain conditions. Here is the brief of the conditions: causality — the server will not allocate resource to any unknown request, or non-arrived requests. limited duration — server will not allocation resource for any request that has left the server, and each request has limited lifetime in the server.

Assume that the request for the Internet server comes at a random time t_k, and $\{t_k\}_k$ forms a random point process. Writing in mathematical format, it is: $r(t) = \sum_k \delta(t - t_k)$, where δ is Dirac Delta function.

At each time t_k, the request comes with a size w_k, so the overall traffic can be written as:

$$s(t) = \sum_k w_k \delta(t - t_k)$$

This process is named as marked random point process as shown in Figure 3.5.

3.5.2 Second Order Properties of Random Point Process

Given the process in Eq 3.19, assuming that the size for each individual request is independent from each other. It is interesting to see how does the major factors: size and arrival affect the overall traffic. To see this, we derive the second order statistics of the final process.

$$\begin{aligned} E[s(t)] &= E_w E_t[\sum_k w_k \delta(t - t_k)] \\ &= \overline{w}\lambda(t) \end{aligned} \tag{3.24}$$

where $\overline{w}$ is the expected value of the size of each request, and $\lambda(t)$ is the average of the point process $r(t)$. If we assume the wide-sense stationary condition in $r(t)$, the Eq. can b further simplified as: $\overline{w}\lambda$.

Correspondingly, the autocorrelation of the process $s(t)$ is

$$\begin{aligned}
R_s(t,\tau) &= E[s(t)s(t+\tau)] \\
&= E_w E_t[\sum_k w_k \delta(t-t_k) \sum_n w_n \delta(t-t_n+\tau)] \\
&= E_w E_t[\sum_k w_k^2 \delta(t-t_k)\delta(t-t_k+\tau)] + \\
&\quad E_w E_t[\sum_k w_k \delta(t-t_k) \sum_{n\neq k} w_n \delta(t-t_n+\tau)] \\
&= \overline{w^2}\lambda\delta(\tau) + \\
&\quad \overline{w}^2 E_t[\sum_k \delta(t-t_k) \sum_n \delta(t-t_n+\tau)] - \\
&\quad \overline{w}^2 E_t[\sum_k \delta(t-t_k)\delta(t-t_k+\tau)] \\
&= (\overline{w^2} - \overline{w}^2)\lambda\delta(\tau) \qquad (3.25) \\
&\quad + \overline{w}^2 R_r(t,\tau), \qquad (3.26)
\end{aligned}$$

where $\overline{w} = E[w]$ and $\overline{w^2} = E[w^2]$, E_w is expectation taken on the size and E_t is the expectation taken on the time t.

The correlation of the traffic is composed of two parts: First, the correlation caused by the random size of each request, under independent situation, this correlation is an impulse function with the only nonzero value at zero displacement, with weight equal to the variance of the size. Second part is caused by the correlation of the arrival point process, weighted by the square of mean of size.

Correspondingly, the power spectrum of the incoming traffic, which is the Fourier transform of the traffic can be written as:

$$PSD_s(\theta) = \lambda(\overline{w^2} - \overline{w}^2) + \overline{w}^2 PSD_r(\theta) \tag{3.27}$$

Assume that as each request arrives at the server, it is subject to a scheduler for resource allocation. The scheduler in our model is described as a random function $h_k(t)$, for the request arrived at time t_k. Several conditions are imposed, for the fact that it is a scheduler in the server.

1.
$$h_k(t) \geq 0, \forall \text{t}. \tag{3.28}$$

2.
$$h_k(t) = 0, \text{for } \ t \leq 0. \tag{3.29}$$

3.
$$h_k(t) = 0, \text{for } \ t \geq t' \tag{3.30}$$

First condition means that request can only consume the resources, not increase resources to the server. The second condition ensure the causality of the scheduler, i.e. no resource should be assigned before this request comes to the server. The third condition ensures the resource can only stay in the server for limited duration.

Naturally, the workload in the server is the accumulation of resources that have assigned

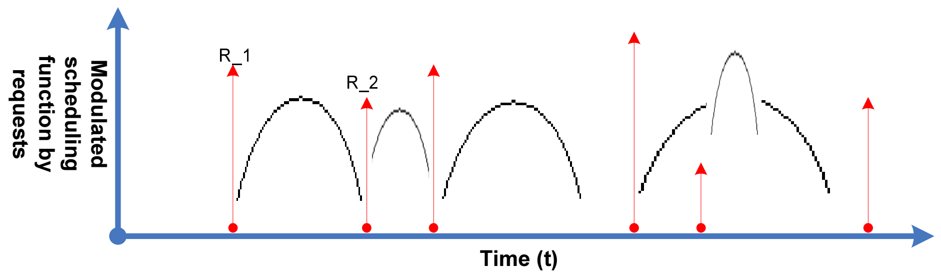


Figure 3.6: Scheduling function modulated marked point process and aggregation of these modulated functions leads to server workload process.

to all active requests in the server, that is

$$l(t) = \sum_k w_k\{\delta(t - t_k) * h_k(t)\}, \tag{3.31}$$

where $*$ is the linear convolution operator. Or simply,

$$l(t) = \sum_k w_k h_k(t - t_k) \tag{3.32}$$

Based on above definition, it can be seen that server can be effectively modeled as the filtering relationship with the kernel of the system corresponds to the scheduling of the system. Figure 3.6 shows that when server allocate resource to each point, with a (set of) scheduling function, the aggregation of these allocation translates into the workload (utilization) process in the server.

To simplify the derivation, let,

$$w_k h_k(t - t_k) \equiv d_k(t - t_k) \tag{3.33}$$

The second order statistics of the workload process can be derived as:

$$\begin{aligned} E[l(t)] &= E_t E_d[\sum_k d_k(t-t_k)] \\ &= E_t[\sum_k d(t-t_k)] \\ &= E_t[\sum_k \delta(t-t_k)] * d(t) \\ &= \lambda(t) * d(t), \end{aligned} \tag{3.34}$$

where, $d(t)$ is the average of $d_k(t)$, $E_d[d_k(t)]$.

If the random point process is stationary, the expression can be further simplified as:

$$E[l(t)] = \lambda \int_{-\infty}^{\infty} d(t)dt \tag{3.35}$$

The autocorrelation function of the workload process is then:

$$
\begin{aligned}
R_l(t,\tau) &= E_t E_d[\sum_k d_k(t-t_k)\sum_n d_n(t-t_n+\tau)] \\
&= E_t E_d[\sum_k d_k(t-t_k)d_k(t-t_k+\tau)] + \\
&\quad E_t E_d[\sum_k d_k(t-t_k)\sum_{n\neq k} d_n(t-t_n+\tau)] \\
&= E_t[\sum_k R_d(t-t_k,\tau)] + \\
&\quad E_t[\sum_k d(t-t_k)\sum_n d(t-t_n+\tau)] - \\
&\quad E_t[\sum_k d(t-t_k)d(t-t_k+\tau)] \\
&= E_t[\sum_k \delta(t-t_k) * R_d(t,\tau)] + E_t[(\sum_k \delta(t-t_k) * d(t)) \\
&\quad \cdot(\sum_n \delta(t-t_n+\tau) * d(t))] - E_t[\sum_k \delta(t-t_k) * d(t)d(t+\tau)] \\
&= \lambda(t) * R_d(t,\tau) + \int\int R_d(t-x,\tau+x-y)d(x)d(y)dxdy - \qquad (3.36) \\
&\quad \lambda(t) * [d(t)d(t+\tau)] \qquad (3.37)
\end{aligned}
$$

When the random point process follows stationary condition, the autocorrelation simplifies as:

$$
R_l(\tau) = \lambda \int_{-\infty}^{\infty} R_{d_k}(t,\tau)dt + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R_d(\tau+x-y)dxdy - \lambda \int_{-\infty}^{\infty} d(t)d(t+\tau)dt. \qquad (3.38)
$$

And apply Fourier transform to above expression, we can derive the power spectrum of the workload process:

$$PSD_l(\theta) = \lambda[\overline{|D_k(\theta)|^2} - \overline{|D(\theta)|}^2] + PSD_r(\theta)|D(\theta)|^2 \tag{3.39}$$

Note that the first group of expression is the difference between “mean of square” and “square of mean” of the Fourier transform of the random function $d_k(t)$.

Corresponding to the equation for the pure traffic, the power spectrum of the workload process is also formed by two parts. The first term corresponding to the power that is caused by the random sizes, while the second term refers to the power in the correlation of the request arrival random process.

3.5.3 Decay Function Modulated Random Point Process

In this section, we assume that the service agreement in the server requires that all requests to be completed within a fixed time window, so that the users of the services can have meaningful and bounded service delay (in making above assumption, we temporary ignore the delay factor caused by the network during the computation). Denoting the fixed service deadline as t_d, the random scheduling function can now be written as:

$$d_k(t) \equiv w_k h_k(t), \tag{3.40}$$

where $h_k(t)$ is another random function following an additional condition:

$$\int_t h_k(t)dt \equiv 1. \tag{3.41}$$

Under this assumption, the power spectrum of the workload process can be simplified as:

$$PSD_l(\theta) = \{\lambda[\overline{w^2|H_k(\theta)|^2} - \overline{w}^2\overline{|H(\theta)|}^2]\} + \overline{w}^2 PSD_r(\theta)|H(\theta)|^2 \tag{3.42}$$

Above expression 3.42 can be separated into two halves, the first half $\{\lambda[\overline{w^2|H_k(\theta)|^2} - \overline{w}^2\overline{|H(\theta)|}^2]\}$ is independent of correlation of the incoming traffic. Thus why we call it "sizing factor". The second half, $\overline{w}^2 PSD_r(\theta)|H(\theta)|^2$, is exactly what we derived for compounded traffic with correlation of traffic — power spectral reflected in the formula, and thus we call it "correlation factor".

Recall that previous researches have pointed out the two dynamics for the traffic [78]. Correspondingly, we show that power spectrum of the server workload is also heavily influenced by the two factors from traffic, and with expressions in Eq. 3.42. This relationship is illustrated in Figure 3.7.

In Chapter 4, the effect of scheduling and its relationship with above two factors in Eq. 3.42 are analyzed in details.

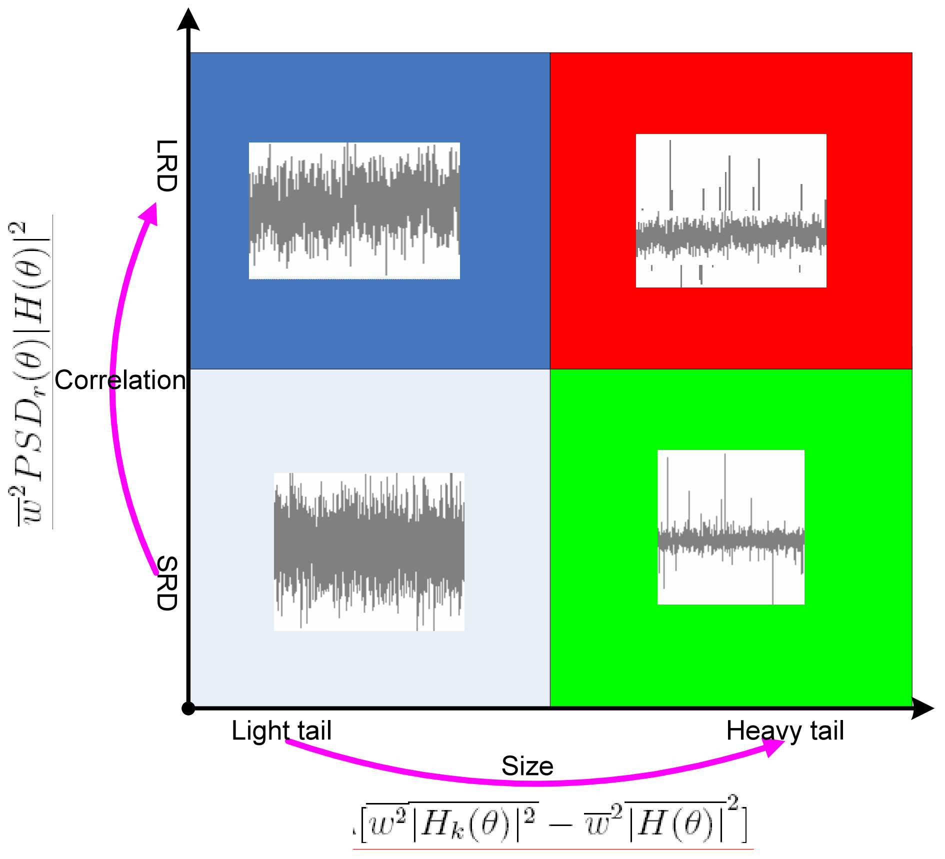

LRD
SRD
Correlation
$\overline{w}^2 PSD_r(\theta)|H(\theta)|^2$
Light tail
Heavy tail
Size
$\sqrt{\overline{w^2|H_k(\theta)|^2} - \overline{w}^2\overline{|H(\theta)|}^2}$

Figure 3.7: Map of four regions [78] in traffic dynamics to server workload dynamics with workload equation 3.42 for decay function modulated point process .

Chapter 4

Optimality Analysis of the Scheduler

Traditionally, people emphasis more on the impact of scheduler to individual requests rather on the performance of the server, such as fairness analysis of lottery scheduler and its variants, response-time analysis of Shortest-Job-First scheduling. Above approach is valid under the situation that incoming requests are well bounded, and that QoS of server is predictable when capacity is overly-provisioned. This assumption is no longer the truth in Internet, where traffic is much more dynamic and traditionally overly-provisioning capacity can no longer provide guarantee for the server. In this circumstance, it is important to evaluate the impact of each individual scheduling on the overall server performance. In queueing theories, this has to be done for each type of scheduling individually. Our model provides a more convenient approach as we show that there exists an optimal scheduler for each given traffic and service deadline constraint. All the encountered server performance analysis can be carried around the optimal scheduler, which not only provides response time guarantee to individual requests, but also optimize the server performance in terms of minimizing variance of server workload process.

This chapter will prove the existence of such an optimal scheduler, and will conduct simulations to validate the optimality of such scheduler.

4.1 Optimal Fixed Time Scheduling

It is known that the requirement for server capacity can be relaxed by reducing server load variance so as to maintain a uniform server load timewise. The goal of scheduling is to minimize the workload variance.

Define $\mathbf{h}(t) = [h(t), h(t+1), \cdots, h(t+t_d-1)]'$ as a vector form of the decay function, and $\mathbf{w}(t) = [\tilde{w}(t), \tilde{w}(t-1), \cdots, \tilde{w}(t-t_d+1)]'$ as a vector form of request sizes. Let $\Omega(t) = E[\mathbf{w}(t)\mathbf{w}'(t)]$, being the correlation matrix of input process $\tilde{w}$ in the order of t_d. We formulate the optimization problem in the following theorem. We also prove there exists one and only one solution to this problem.

Theorem 4.1. *If the compounded input process $\tilde{w}(t)$ is wide sense stationary, the optimal fixed-time scheduling is to find a $\mathbf{h}$ that*

$$\textit{Minimize} \quad \mathbf{h}'\Omega\mathbf{h} \tag{4.1}$$

$$\textit{Subject to} \quad \sum_{i=0}^{t_d} \mathbf{h}_i = 1, \ \textit{and}\, \mathbf{h}_i \geq 0. \tag{4.2}$$

Moreover, the optimization problem (4.1) has a unique solution.

Proof: Write (3.6) in vector format as

$$l(t) = \mathbf{h}'(t)\mathbf{w}(t). \tag{4.3}$$

Recall that $\tilde{w}$ is a WSS (wide sense stationary) process and $\sum_{t=0,\ldots,t_d-1} h(t) = 1$. It follows that the mean of system load at time t

$$E[l(t)] = E[\mathbf{h}'(t)\mathbf{w}(t)] = E[w].$$

The explanation is that the server must finish all the work requirements for the requests. As we assume no degradation in the services (sum of h must be one), it has nothing to do with scheduling policy.

The variance of system load at time is

$$\begin{aligned} Var(l(t)) &= E[l^2(t)] - (E[l(t)])^2 \\ &= E[\mathbf{h}'(t)\mathbf{w}(t)\mathbf{h}'(t)\mathbf{w}(t)] - (E[w])^2 \\ &= E[\mathbf{h}'(t)\mathbf{w}(t)\mathbf{w}'(t)\mathbf{h}(t)] - (E[w])^2 \qquad (4.4) \\ &= \mathbf{h}'(t)\Omega\mathbf{h}(t) - (E[w])^2. \qquad (4.5) \end{aligned}$$

where $\Omega = E[\mathbf{w}(t)\mathbf{w}'(t)]$.

We point out that Ω is semi-positive definite matrix. That is, for any non-zero vector $x \in \mathbf{R}^{t_d}$, $x'\Omega x \geq 0$. In real Internet environment, the covariance of input traffic random process should be non-degenerating — it is impossible to determine one component of $\mathbf{w}(t)$ from other components of $\mathbf{w}(t)$ with probability one [26]. This means the correlation matrix is strictly positive definite.

Since Ω is a symmetric positive definite matrix, there exists an orthogonal matrix $\mathbf{U}$, such that $\mathbf{U}^{-1}\Omega\mathbf{U} = \Lambda$ and

$$\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_{t_d})$$

where λ_i are the eigenvalues of the matrix Ω and $\lambda_i > 0$.

Let $y = \mathbf{h}'\Omega\mathbf{h}$. It follows that

$$y = \mathbf{h}'\mathbf{U}\Lambda\mathbf{U}^{-1}\mathbf{h}$$

Define $\mathbf{g} = \mathbf{U}^{-1}\mathbf{h}$. It follows $y = \mathbf{g}'\Lambda\mathbf{g}$. This is a standard form. It is minimized when $\lambda_i g_i^2 = \lambda_j g_j^2$ for any $1 \leq i, j \leq t_d$. Since $\mathbf{h}'\mathbf{1} = 1$ and $\mathbf{U}\mathbf{g} = \mathbf{h}$, there exits one and only one solution of $\mathbf{h}$ for the optimization problem (4.1). QED

This theorem reveals the impact of correlation structure of Internet traffic on the scheduling policy.

In the following, we give two optimal results in the case that the compound input process is homogeneous. The first result can be derived from Theorem 4.1 and its proof is omitted. We demonstrate the calculus of the results by examining input traffic with different autocorrelation functions.

Corollary 4.1. *If the compounded input process $\tilde{w}(t)$ is independent for different time t, the optimal fixed-time scheduling is to allocate equal amount of resource to a request at each time unit before its deadline.*

The optimal scheduler in Theorem 4.1 can be solved by non-linear programming techniques. We demonstrate the shape of optimal scheduler in the following example.

Example 4.1. *In the first example, we illustrate the above theorem by considering several traffic with distinct autocorrelation functions(ACFs)* $p(\tau)$*, where* τ *is the lag of time unit. The server is assumed to finish each unit of task by 10 time units; that is,* $t_d = 10$ *for each request. The first two sample traffic models are multimedia traffic patterns introduced in [38]. One with* shifted exponential scene-length distribution *whose ACF is* $p(\tau) = e^{-\beta|\tau|}$ *and* $\beta = 1/49$ *and the other is* subgeometric scene-length distribution *with ACF in recursive form* $p(\tau) = p(\tau-1) - \alpha^{\sqrt{\tau}}/d$*, where* $\alpha = 0.8$ *and* $d = 40.67$*. The third traffic model under consideration is the* Fractional Gaussian Noise *process with a Hurst parameter* $H = 0.89$*, as described in [56]. We applied an exponential transformation to this traffic to eliminate negative values.*

The last traffic model was generated from a real scene-based MPEG I video trace from the popular "Simpsons" cartoon clip [66]. It consists of around 20,000 frames which lasts for about 10 mins at 30 frames per second. The video clip has a Hurst parameter 0.89 and thus posses high degrees of long-range dependence and burstiness. The autocorrelation structure was produced at a coarse-grained Group-of-Pictures level under the assumption that the expected scene length was 50.

Figure 4.1 shows the ACFs of the traffic models, where ACFEXPON, ACFSUBGEO, ACFFGN, and SCFMPEG represent the four distinct traffic models respectively. Their radical impacts on the optimality of scheduling functions (Theorem 4.1) are shown on Figure 4.2. A uniform decay function for independent traffic (Corollary 4.1) is included in Figure 4.2 for reference.

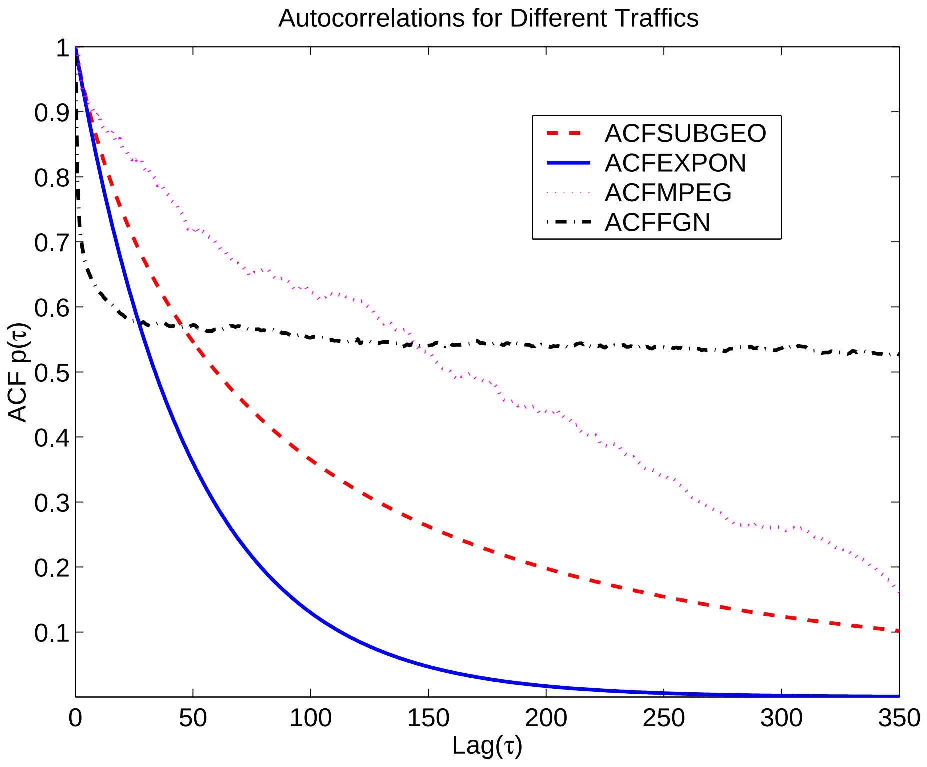


Figure 4.1: Four distinct patterns of auto-Correlation function.

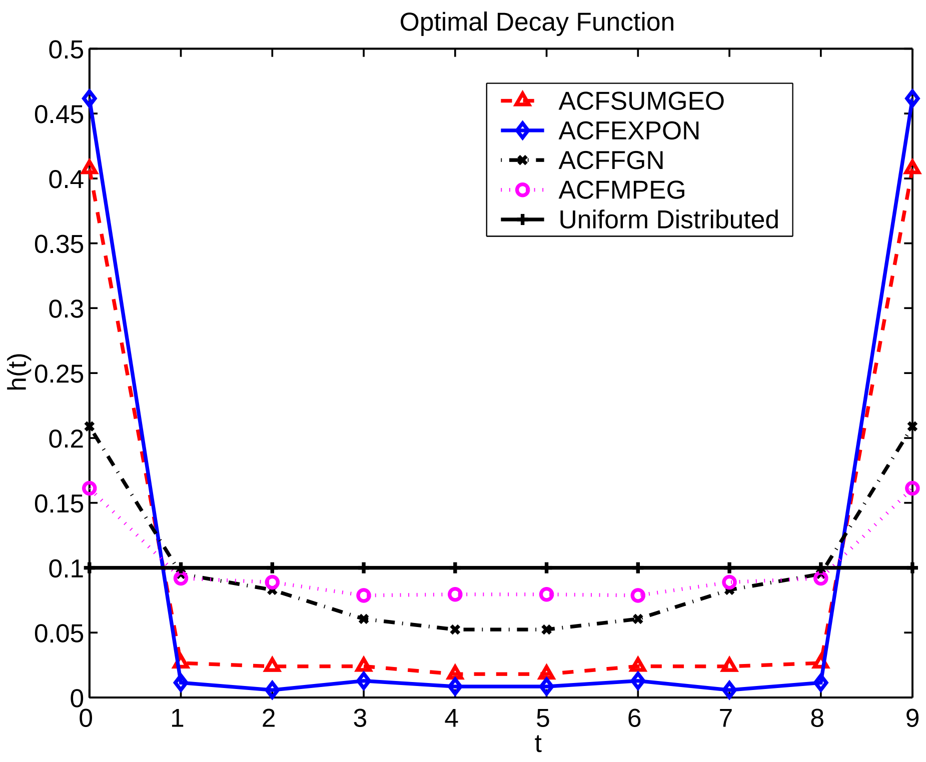


Figure 4.2: Optimal decay functions for traffic with different ACFs.

4.2 Performance Evaluation

To verify the above analytical results, we conducted simulations of the Internet server based on the decay function model. The simulation was conducted in two aspects: (1) Compare the decay function scheduling with another popular GPS scheduling [54]; (2) Analyze the sensitivity of the decay function to request traffic.

4.2.1 Experiment Setup

We generated normal random numbers for request size and lognormal random numbers for the number of arrival requests at each scheduling epoch (unit time). All these random numbers are generated inter-independently. The server was simulated with an infinite queue so that we can find exact distribution for the completion time of each request. This distribution serves as a common base for comparison of different scheduling algorithms. We did not consider admission control in the simulated server because the relationship between completion time and rejection rate is beyond the scope of this paper. The simulated server treats all requests with the same priority. When a request arrives at the server, it first be buffered in the queue. If the server has enough resource, the request will be scheduled for processing immediately without any queueing delay. Otherwise, the request will be blocked in the queue. We implemented two different schedulers on the server module: one for fixed-time scheduling and the other for GPS scheduling. GPS scheduling ensures fair-sharing of resource between requests in processing at any scheduling epoch. Two performance metrics were used: completion time and server load. Completion time of a request includes waiting

Table 4.1: Notations of Performance Metrics

μ_t	the mean of completion time
σ_t^2	the variance of completion time
$\%_t$	the percentage of requests meeting the deadline t_d
μ_l	the mean of server workload
σ_l^2	the variance of workload
$\%_{\mathbf{c}}$	the percentage of server workload less than capacity **c**

time in the back logged queue, plus its processing time.

In summary, the simulation proceeds in the way as: (1) generate the number of arrival requests $n(t)$ at time t from a lognormal distribution; (2) for each request, it is assigned a size $w(t)$ from a normal distribution; (3) each request then enters the server for scheduling; (4) measure the completion time of each request and the server workload at each scheduling epoch as output. Because the number of arrival requests at each time and their sizes are i.i.d. random numbers, the compounded workload $\tilde{w}(t)$ is not only a WSS process, but also an ergodic process [50]. In this section, we use notations as listed in Table 4.1.

4.2.2 Comparing Different Schedulers

We compared the optimal fixed-time scheduling with the GPS scheduling. In addition, we implemented two more heuristic fixed-time scheduling algorithms. One is *randomized allocation* that takes t_d to random numbers from a uniform distribution and then scales them to standard decay functions $h(t)$ so that $\sum_{i=1}^{t_d} h(t) = 1$. The other is *incremental scheduling*, in which the server increases resource allocation at a constant rate b to a request as its deadline approaches. That is, $h(t+1) = h(t) + b$. We assume the initial allocation in the first scheduling epoch $h(1) = b$, as well. Table 4.2 presents their comparative results.

Table 4.2: Statistics of server with various scheduling algorithms. ($\mathbf{c} = 171000$, $n(t) \sim ln(1, 1.5)$, $w \sim n(5000, 1000)$, $t_d = 10$)

Scheduling	GPS	Incremental	Random	Optimal
Load Mean μ_l	37534	37534	37534	37534
Load Variance σ_l^2 (10^9)	1.68	1.22	1.00	0.894
Capacity Satisf. $\%_c$	99.8%	96.7%	99.2%	99.3%
Time Mean μ_t	4.25	10.08	10.13	10.08
Time Variance σ_t^2	5.38	0.70	0.51	0.33
Deadline Satisf. $\%_t$	96.7%	90.6%	97.0%	97.3%

From the table, it can be observed that the optimal fixed-time scheduling outperformed the others in two aspects. First, it dramatically lowers the variance of server workload by up to 50% in comparison with the GPS scheduling and more than 10% in comparison with the other fixed time scheduling. Second, it provides better guaranteed completion time. GPS scheduling leads to a lower mean of completion time and a high variance. As a result, the percentage of requests that miss their deadlines is much higher than fixed-time scheduling. All the fixed-time scheduling algorithms guarantee completion-time, the optimal scheduling algorithm yields a much smaller time variance, in comparison with incremental and randomized allocations.

Figure 4.3 shows the empirical CDF of the server load due to different scheduling algorithms. From this figure, we can see the tail of the optimal fixed-time scheduling is lighter than GPS and other heuristic fixed-time scheduling algorithms. This again indicates the superiority of the optimal fix-time scheduling.

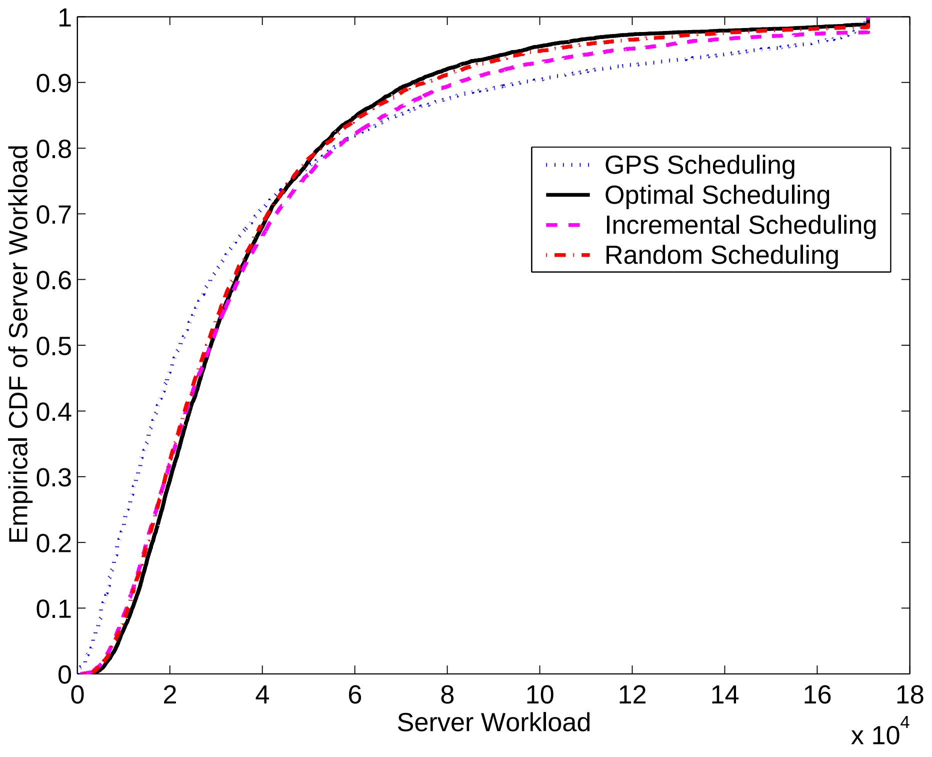


Figure 4.3: The empirical CDF of server load with various scheduling algorithms.

Table 4.3: Statistics server with various scheduling algorithms. ($\mathbf{c} = 171000$, $n(t) \sim ln(1, 1.7)$, $w \sim n(5000, 1000)$, $t_d = 10$)

Scheduling	GPS	Incremental	Random	Optimal
Load Mean μ_l	52497	52497	52497	52497
Load Variance σ_l^2 (10^9)	2.92	2.31	2.04	0.189
Capacity Satisf. $\%_c$	99%	96.2%	96.8%	97%
Time Mean μ_t	9.2	11.1	11.1	10.9
Time Variance σ_t^2	65.7	14.8	14.5	13.2
Deadline Satisf. $\%_t$	84%	77%	81%	86%

4.2.3 Sensitivity to the Change of Traffic Intensity

The Internet features high variability traffic patterns. It is not uncommon for an Internet server to experience a heavier-than-expected incoming request traffic without any warning. In this simulation, we show that the optimality of fixed-time scheduling stands when the number of arrival requests $n(t)$ increases for each unit time. We increased the lognormal distribution parameter σ from 1.5 to 1.7, while keeping all the other parameters unchanged. This small increase in the parameter σ leads to a jump of server workload by almost 140%.

Table 4.3 shows that the variances of server load and request completion time due to the optimal fixed-time scheduling are significant lower than those in GPS and other heuristic fixed-time scheduling. The 97% of capacity satisfactory rate $\%_\mathbf{c}$ reveals that the optimal fixed-time scheduling likely leads the server to fully loaded conditions. In other words, the optimal fixed-time scheduling is able to fully utilize existing resource on the server. This can be seen from the completion time index. It shows that more requests can be finished within deadline in the optimal fixed-time scheduling than the other scheduling algorithms . With a lower variance, the tail of completion time due to the optimal fixed-time scheduling

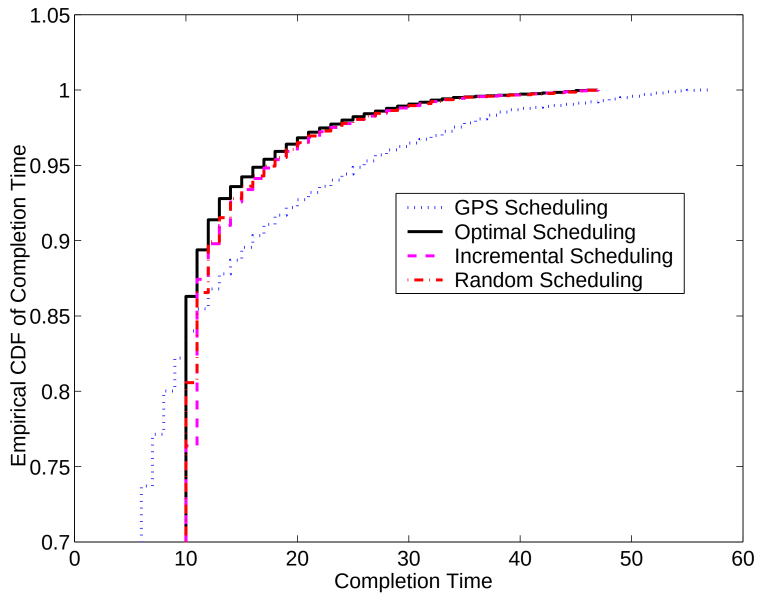


Figure 4.4: The CDF plot of completion time when $n(t) \sim ln(1, 1.7)$

is significantly lighter than GPS scheduling in Figure 4.4 and Figure 4.5. With a light-tail and small variance, the optimal fixed-time scheduling tends to provide better deadline guarantees in terms of both maximum and average delays.

Notice that in the above simulation, the average utilization of the server was maintained at a range of $[40\%, 60\%]$. From this point of view, the price for providing guaranteed service time on the Internet server under high-variability traffic is very high.

4.3 Optimality of Scheduling for Power Spectral

Theorem 4.1 in Chapter 4 points out the existence of optimal scheduler. This optimality, as shown in the proof, is actually achieved by minimizing variance in server workload process. In Chapter 3, the input and output relationship for scheduler is extended into frequency

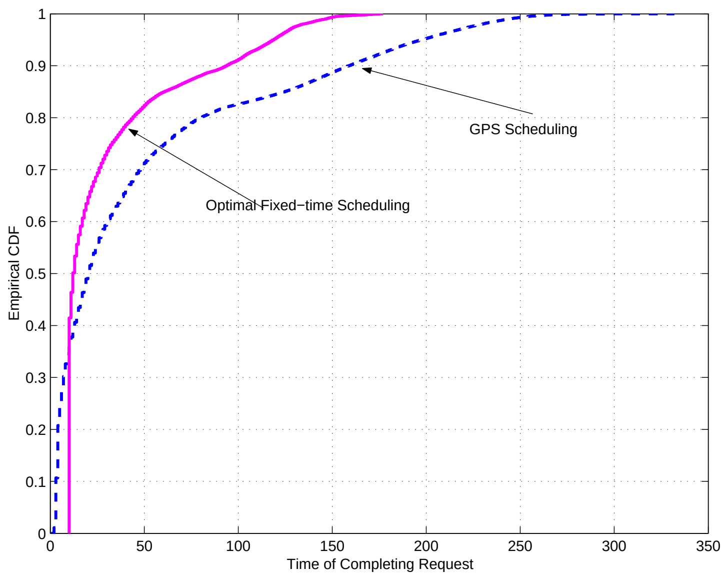


Figure 4.5: The CDF plot of completion time when $n(t) \sim ln(1, 2)$

domain. In this chapter, we not only extend optimal condition of scheduler into frequency domain, but also prove the *Convex* scheduling policy for traffic with monotonically decreasing PSD functions. The convex rule is applied into GPS scheduler, simulation shows the performance improvement through convex-enhancement.

4.4 Minimum Power Scheduling

Server scheduler leaks power of input traffic process into server utilization process. It is known that the cumulative power function (CPF) of a random process reflects its variance in the time domain. Reducing the CPF of system utilization process leads to a more stable and predictable server operation condition. It can also help reduce the demand for server capacity in QoS provisioning. Therefore, the optimality of a scheduler in second-order statistics is to minimize the power leakage for input traffic.

The existence of such an optimal scheduler has been proved in the time domain in previous chapter in the sense of least-mean-square. From the definition of PSD function, we know that CPF $A_l(\pi)$ is equivalent to the variance of process $l(t)$ in the time domain. We present the existence of the optimal scheduler in the frequency domain in the following theorem.

Theorem 4.2. *For input traffic with a wide-sense stationary compounded input process $\tilde{w}(t)$ in (3.7), there exists one and only one scheduler that minimizes the CPF $A_l(\pi)$ of server utilization process $l(t)$.*

Generally, the optimal scheduler could be any structure as long as it meets the requirements for scheduling function in (3.4). When input traffic is i.i.d., its PSD function corresponds to a horizontal line in the frequency domain. The flat PSD function means the power of input traffic is distributed evenly in all frequencies. In the following theorem, we prove that uniform scheduler $h_u(t)$ is optimal for i.i.d. traffic.

Theorem 4.3. *For an i.i.d. traffic and a given service deadline t_d, uniform scheduler $h_u(t)$ minimizes the CPF $A_l(\pi)$ of server utilization process.*

Proof: By Parseval theorem, we have

$$\begin{aligned} A_h(\pi) &= \int_0^{\pi} |H(f)|^2 df \\ &= \pi \sum_t h^2(t). \end{aligned} \tag{4.6}$$

According to Cauchy Inequality, the time domain constraint for $h(t)$ in (3.4) becomes,

$$1 = \big(\sum_{t}^{t_d-1} h(t)\big)^2 \le t_d \cdot \sum_{t}^{t_d-1} h^2(t), \tag{4.7}$$

Consequently, (4.6) becomes

$$A_h(\pi) \ge \frac{\pi}{t_d}, \tag{4.8}$$

and it takes equality only when $h(t) = \frac{1}{t_d}$. The lower bound of the $A_h(\pi)$ for schedulers with deadline t_d is inversely proportional to t_d.

When the input traffic is i.i.d, the PSD function takes a constant value in all frequencies f. This means that minimizing $A_l(\pi)$ is equivalent to minimizing $A_h(\pi)$. QED

4.5 Evaluation of Optimality in Correlated Traffics

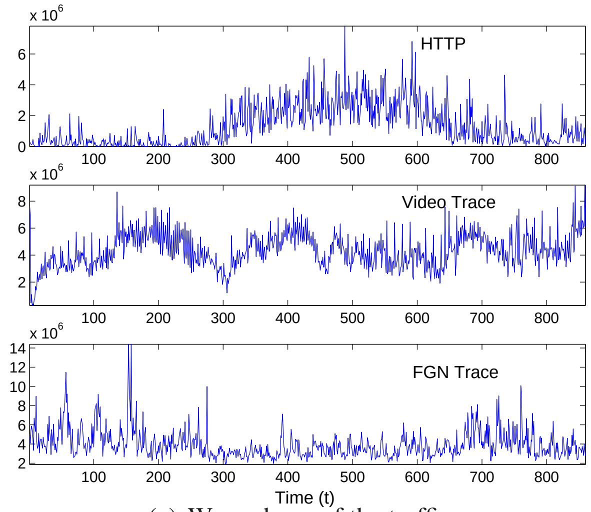
In this chapter, we used traffic models that go beyond the i.i.d. case used in previous chapter to validate the optimality. We assume an Internet server with defined capacity C. The server uses the scheduling function h to schedule resources for the requests. Requests arrive at the server according to three different patterns: (1) A video trace taken from Simpson Cartoon Video [66], (2) A trace generated from FGN model using Paxson method [56], (3) A measured Internet HTTP trace from Internet Traffic Archive [71]. For FGN and Video traffic, each request is matched independently with size from Gaussian distribution $N(20637, 95992)$. The mean and variance for the distribution were taken to match with the mean and variance of response size in HTTP trace. The HTTP traffic is matched with real HTTP response size from the trace. When the server is overloaded (utilization is close to 1), the additional requests are kept in a waiting queue. Four performance indexes were considered in the simulation: mean value of response time, variance of response time, percentage of requests that exceed predefined service deadline of 10 time units (deadline missing rate), and the maximum response time among all requests. Response time is defined as queueing delay plus actual service time.

Three scheduling policies were considered in the experiment, (1) random scheduling function, (2) optimal scheduling function, (3) uniform scheduler.

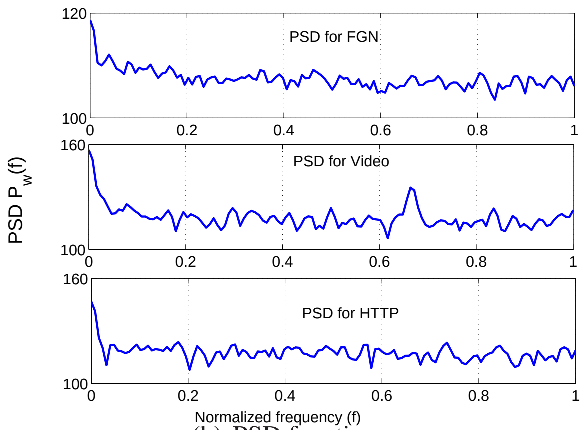
Figure 4.6(a) shows the wave shapes of three traffic. Their PSD functions for each traffic were estimated using periodiagram method, as shown in Figure 4.6(b). FGN traffic has more power in low frequencies. HTTP trace appears to be more independent than other two, with PSD in high frequencies approximately of being flat. The video traffic has less power in the lower frequency range than the FGN trace. It shows a spike in frequency range $[0.6\pi, 0.7\pi]$. The spike in its PSD function indicates a periodic scene in video traces, which is common for VBR video streams.

The martingale distributions for three input processes are shown in Figure 4.6(c). From the figure, HTTP trace is showing an Zipf-alike distribution, which is in agreement with past findings in Internet traffic characterization. In summary, three experimental traces have distinct time-domain wave shapes, distributions and PSD functions. The choice of these traces is to demonstrate the applicability of our model in typical Internet traffic.

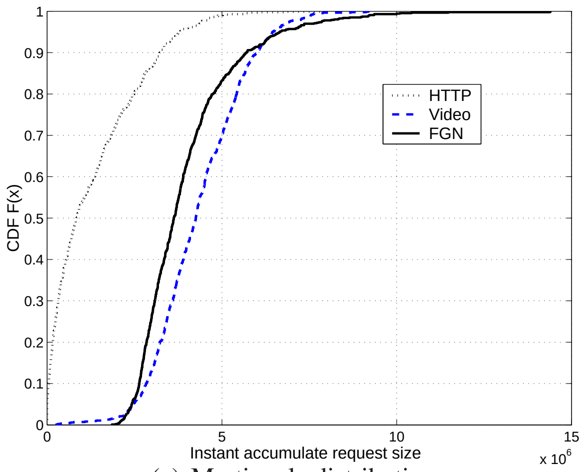
We calculated the optimal scheduler for each traffic trace with with nonlinear program-



(a) Wave shape of the traffic



(b) PSD functions



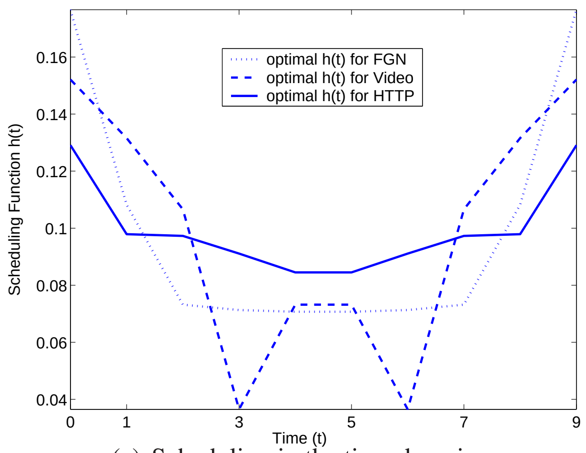
(c) Martingale distribution

Figure 4.6: Distributions of three traffic models in simulation.

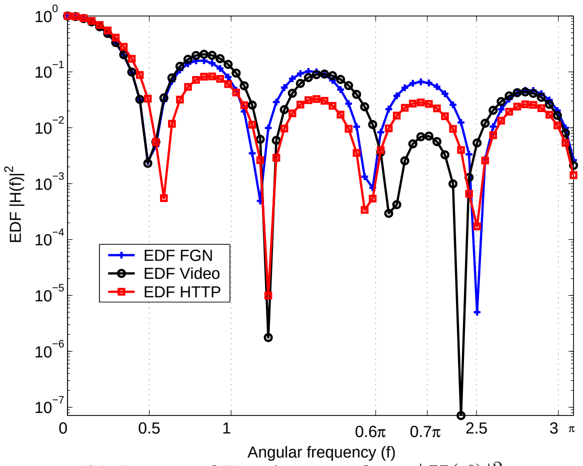
Table 4.4: Server performance with different scheduling policies for different traffic traces. Each trace is simulated using three different schedulers: uniform scheduler, optimal mini-power scheduler and a random scheduler.

FGN	optimal	random	uniform
missing rate	12.20	14.31	12.54
mean	10.2930	10.3684	10.3014
variance	0.8480	1.3657	0.8672
max	15	24	16
Video	optimal	random	uniform
missing rate	6.32	14.35	6.30
mean	10.0640	10.2237	10.0641
variance	0.062	0.5257	0.063
max	11	22	12
HTTP	optimal	random	uniform
missing rate	11.90	15.33	11.94
mean	10.1481	10.2550	10.1526
variance	0.1866	0.5190	0.1982
max	13	20	13

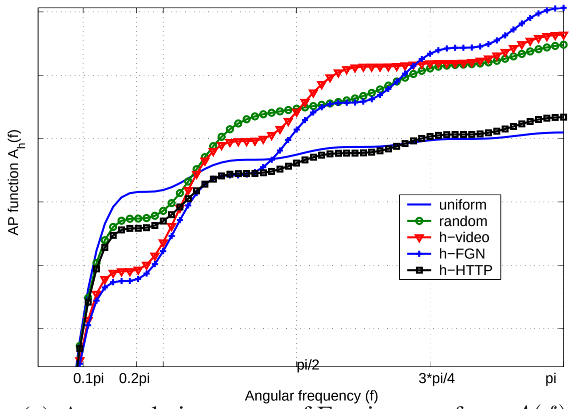
ming under the conditions in Theorem 4.2. They were plotted in Figure 4.7(a). From the figure, we can observe that the schedulers for FGN and HTTP are both convex-structured, with FGN's has a "deeper bowel". scheduler for video streams shows a local concaveness within the valley of convex function. By looking at their EDF plots in Figure 4.7(c), FGN scheduler has the narrowest main lobe, which explains the deeper bowel in convex shape. HTTP scheduler is closer to uniform allocation, this is because HTTP trace is more independent than the other two traces, and the uniform scheduler is optimal for i.i.d. traffic. The video scheduler has a much lower value in frequency range $[0.6\pi, 0.7\pi]$, which illustrated that optimal scheduler is adapted to the spike in video trace's PSD function. By looking at plot of CPF functions in Figure 4.7(c), FGN and video schedulers share the similar properties: lower value in low frequencies at the cost of higher value in high frequencies.



(a) Scheduling in the time domain



(b) Power of Fourier transform $|H(f)|^2$



(c) Accumulative power of Fourier transform $A(f)$

Figure 4.7: Optimal decay functions corresponding to the three traffic models.

Table 4.4 shows the simulated server with different schedulers. Optimal schedulers can outperform other two schedulers in all three traffic in each sense of measurement. Comparing to random scheduler, the optimal schedulers reduced the deadline missing rate by 50% for all input traffic. Same is true for the maximum response time.

The optimal schedulers outperform uniform scheduler marginally There are two reasons for this. even though three traffic traces have different PSD functions, the functions have the same structure — they all decrease fast in lower frequencies band and at a much slower rate in higher frequencies. This corresponds to the signature of i.i.d. traffic. Second, the main lobe width of the schedulers almost covers the fast decreasing region in PSD functions. We define this the cut-off frequency of this fast decreasing region as "cut frequency", and define the corresponding time unit in time domain as "main time-scale". Since service deadline determines the width of main lobe, we conclude that if service deadline is approximately the same as main time-scale of input traffic, the uniform scheduler can be used as a good approximation to the optimal scheduler. The big performance difference between random scheduler and optimal scheduler raised an important question. What determines the optimality of scheduler from structure point of view? This is the problem we answered in following section.

4.5.1 Convex Scheduling

The optimality of scheduling in Theorem 4.2 is in the sense of balancing power leakage across the entire frequency domain. The uniform structure is optimal for i.i.d. input traffic.

In reality, Internet traffic can rarely be modeled as independent.

In previous simulation, the three input traffic traces: video stream, FGN and HTTP trace, share common characteristics in PSD function (Figure 4.6(b)). Their power is mainly concentrated in lower frequencies, and their PSD functions are asymptotically "monotonically decreasing." These characteristics are common in real Internet traffic [28, 5, 8, 49].

In this section, we prove that for traffic with such decreasing PSD functions, the optimal scheduler structure is convex. Before presenting the main result, we introduce the concept of scheduler permutation. For a given scheduler h with service deadline k, we represent its structure as a sequence of instant allocation

$$\langle h(0), h(1), \ldots, h(k-1) \rangle.$$

The scheduler can be in different structures: random, convex, concave, etc.

Let σ denote a permutation of list $(0, 1, \ldots, k-1)$. The permutation h_σ is

$$\langle h(\sigma(0)), h(\sigma(1)), \ldots, h(\sigma(k-1)) \rangle.$$

In the time domain, the permutation for a request changes the sequential order of its resource allocation. In the frequency domain, this structure change causes the redistribution of power in the EDS function in the system utilization process.

According to Parseval theorem, we have

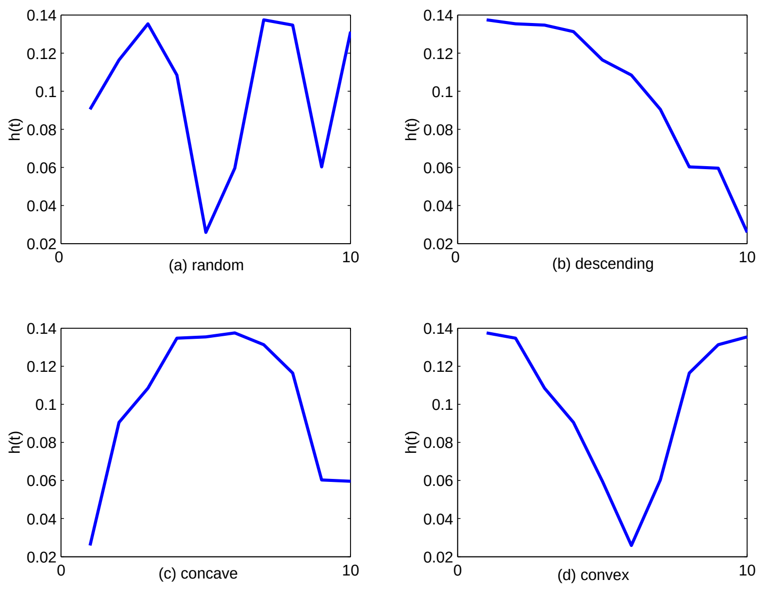


Figure 4.8: Sample permutation of a random scheduling to decreasing, concave and convex shaped scheduling function

Lemma 4.1. *Let h_σ be a permutation of scheduler h. Its CPF remains unchanged and*

$$A_{h_\sigma}(\pi) = A_h(\pi). \tag{4.9}$$

The permutation operator provides us a tool to study the impact of scheduler structure on the server performance. A special structure is "exact-convex", defined as a permutation h_σ, such that

$$\{h(\sigma(0)), h(\sigma(k-1))\} \geq \{h(\sigma(1)), h(\sigma(k-2))\}$$
$$\geq \{h(\sigma(2)), h(\sigma(k-3))\}, \ldots,$$

where $\{a, b\} \geq \{c, d\}$ means $a \geq c, b \geq c, a \geq d,$ and $b \geq d.$

Figure 4.8 shows the sample of permutation of a random scheduler to different structured shapes: monolithically decreasing, concave, and convex.

Theorem 4.4. *For input traffic with monotonically decreasing PSD functions, the optimal scheduler that minimizes the CPF of server utilization process is in an exact convex structure.*

Proof By Theorem 4.2, there exists one and only one optimal scheduler that can minimize the power of workload process. Assume the optimal scheduler is h^*. We prove that if h^* is not convex, there exists a convex permutation h_σ of h^*, that will generate smaller power of the workload process.

It is know that

$$\begin{aligned} A_l(\pi) &= \int_0^{\pi} |H^*(f)|^2 P_{\tilde{w}} df \\ &= \int_0^{\omega} |H^*(f)|^2 P_{\tilde{w}} df \\ &\quad + \int_{\pi-\omega}^{\pi} |H^*(f)|^2 P_{\tilde{w}} df. \end{aligned} \tag{4.10}$$

Suppose that $|H_\sigma(f)|^2 > |H^*(f)|^2$. According to Lemma 4.1, it follows that,

$$\begin{aligned} &|H^*(f)|^2 > |H_\sigma(f)|^2, f \in [0, \omega] \\ &|H^*(f)|^2 < |H_\sigma(f)|^2, f \in (\omega, \pi] \end{aligned} \tag{4.11}$$

Because $P_{\tilde{w}}(f)$ is monotonically decreasing with f, by using Rearrangement Inequality, we can prove,

$$\begin{aligned} A_{l_\sigma}(\pi) &= \int_0^\pi |H_\sigma(f)|^2 P_{\tilde{w}} df \\ &< \int_0^\pi |H^*(f)|^2 P_{\tilde{w}} df = A_l(\pi). \end{aligned} \tag{4.12}$$

It means that in order to prove h_σ is a better scheduler than h^*, we only need to prove inequalities (4.11)

Following the proof of Theorem 3.3, we know

$$|H(f)|^2 = \sum_{t_1=0}^{t_d-1} \sum_{t_2=0}^{t_d-1} h(t_1)h(t_2)cos\big((t_1-t_2)f\big) \tag{4.13}$$

We define two matrices to simplify the equation,

$$\overline{C} = \big[h(t_1)h(t_2)\big]_{t_1,t_2 \in N_{[0,t_d-1]}}, \tag{4.14}$$

and,

$$\overline{S}(f) = \big[cos\big((t_1-t_2)f\big)\big]_{t_1,t_2 \in N_{[0,t_d-1]}}, \tag{4.15}$$

where $N_{[0,t_d-1]}$ is set of integers between 0 and $t_d - 1$.

It follows that,

$$|H(f)|^2 = \overline{C} \otimes \overline{S}(f), \tag{4.16}$$

where $\otimes$ is the "inner product" of two matrics that sums up element-wise products of the

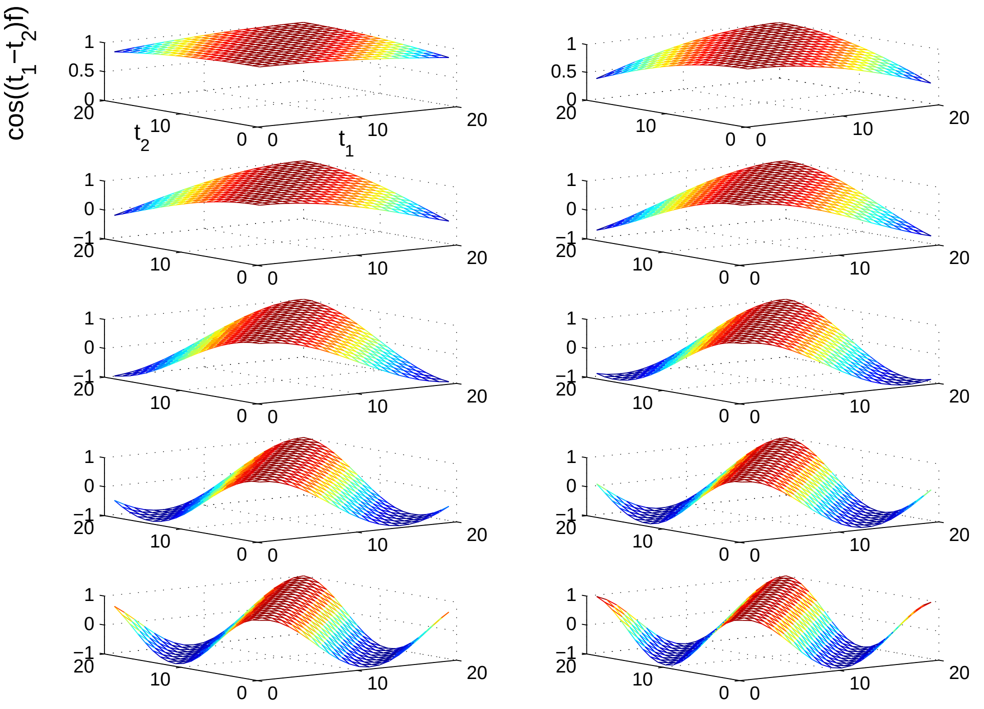


Figure 4.9: The Cosine surfface $\overline{S}(f)$ changing with different frequency f.

matrics.

We call matrix $\overline{S}(f)$ as cosine surface (see Figure 4.9) Cosine surface has the following properties when $f \in [0, \pi/(t_d - 1)]$:

(1) $\overline{S}(f)$ takes the maximum value at the center. That is $\overline{S}(f)_{[\frac{t_d}{2}],[\frac{t_d}{2}]} \geq \overline{S}_{i,j}$, for all $i, j \in [0, t_d - 1]$;

(2) $\overline{S}(f)$ is monotonically decreasing from this maximum point. Proof follows: First draw a straight line from maximum point, $([\frac{t_d}{2}], [\frac{t_d}{2}])$, to any point on the edge of $(t_1 - t_2)$ plane; Taking two points in the straight line, say (x_1, y_1), (x_2, y_2) and their corresponding projections in the Cosine surface $\overline{S}(f)$ is $\overline{S}(f)_{x_1,y_1}$ and $\overline{S}(f)_{x_2,y_2}$. Without loss of generality, assuming (x_1, y_1) is closer to the center $([\frac{t_d}{2}], [\frac{t_d}{2}])$. It is evident that $|x_1 - y_1| \leq |x_2 - y_2|$,

which means $\overline{S}(f)_{x_1,y_1} \geq \overline{S}(f)_{x_2,y_2}$. The two inequalities only take the equal sign when the straight line happens to be the main-diagonal line.

Similarly, we can prove that, when a scheduler is in the shape of exact-convex, the scheduling surface defined by $\overline{C}$ takes minimum value at its center $([\frac{t_d}{2}], [\frac{t_d}{2}])$. It is monotonically increasing in the same fashion as $\overline{S}(f)$ according to the definition of exact convex.

Based on Rearrange Inequality, an exact convex permutation will have a smaller value in $|H_\sigma(f)|^2$, when $f \in [0, \pi/(t_d - 1)]$. This exactly proves the inequalities (4.11). Consequently, the optimal scheduler must have an exact convex shape.

The above proof make an simplification at the center of $(t_1 - t_2)$ plane. Strictly to say, the points in the plane are discrete. When t_d is an even number, there would be four points qualifying for the plane center. We can make further discussions by treating one of them as the center. The conclusion remains the same.

This completes our proof.QED

Intuitively, for traffic with a decreasing PSD function, we would expect the optimal scheduler can minimize the power leakage in lower frequencies covered under its main lobe. This is exactly what a convex scheduler achieves.

To validate the theorem, we conducted another simulation under the same setup as in the last experiment. We built a new convex scheduler by permuting the random scheduler used in that experiment. The sequence for the new scheduler, which is exact-convex, is

$$h_\sigma = \langle h(\sigma(0)), h(\sigma(1)), \ldots, h(\sigma(k-1)) \rangle$$

Table 4.5: Server performance with a convex scheduler. The scheduler is a permuation of the random scheduler used in the last experiment.

traffic	FGN	Video	HTTP
missing rate	12.93	6.64	12.29
mean	10.3012	10.0672	10.1517
variance	0.8625	0.0651	0.1871
max	16	11	13

such that

$$h(\sigma(0)) \geq h(\sigma(k-1)) \geq h(\sigma(1)) \geq h(\sigma(k-2)), \ldots.$$

Table 4.5 listed the results. Comparing them with the performance of the random scheduler in Table 4.4, we can see that the convex scheduler performs almost as good as optimal schedulers. It is a surprise that reshuffling same resource allocation would produce more than 50% of improvement in response-time variance and deadline-missing rate for HTTP traffic, 100% for video streams and 20% for FGN traces. It is also interesting to see that different traffic can lead to different performance gains in this situation. The small improvement for FGN traffic is because FGN generates traffic with LRD property, and that LRD has some negative impact in server performance.

4.5.2 Improved GPS Scheduler

GPS is a popular scheduler for Internet traffic without assuming any advance knowledge of request sizes and their correlation. It allocates server resource equally to all requests waiting for service at any time.

An Internet server can be run into one of the four states with respect to system utilization:

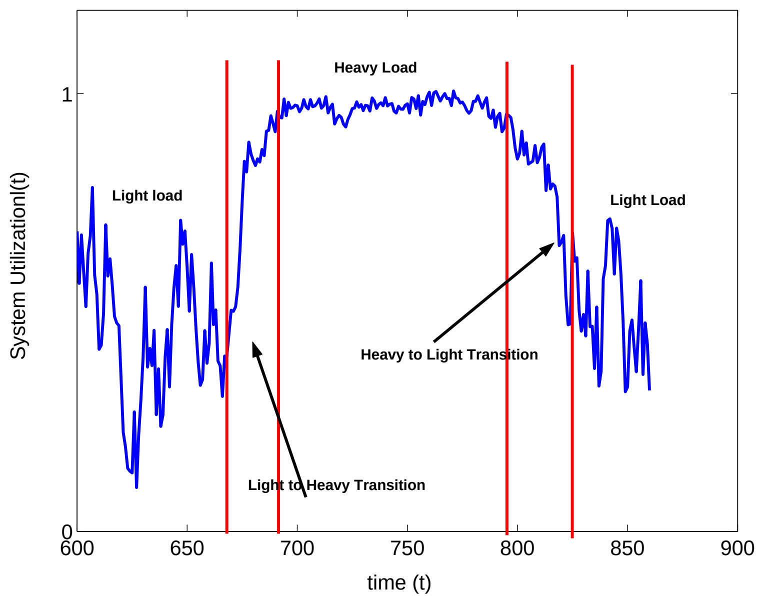


Figure 4.10: System utilization process under an input of video traffic.

(1) under-load, (2) transition from under to heavy load, (3) heavy load, and (4) transition from heavy to under load. Figure 4.10 shows an example of server utilization processes under the input of video traffic benchmark in our simulation.

In states (1) and (4), the server has more than enough resource to allocate. As a conservative scheduler, GPS can process requests at their fastest paces. There is little space for improvement of resource allocation in these two situations. In state (2), as the server becomes more heavily loaded, the waiting requests tend to receive less and less resource until their completion. In state (3), the amount of resource allocated to a request tend to be a random number, depending on the actual number of requests in the server. According to Theorem 4.4, for traffic with monotonically decreasing PSD functions, GPS is not optimal in either of these two cases.

Table 4.6: Statistics of server under two PS schedulers: original PS scheduler and an Convex-enhanced PS scheduler.

Original GPS	FGN	Video	HTTP
missing rate	5.47	7.57	7.89
mean	3.2417	3.941	3.2806
variance	12.4885	15.8016	52.8519
max	33	36	284
Modified GPS	**FGN**	**Video**	**HTTP**
missing rate	1.67	0.48	5.96
mean	2.2031	2.2237	2.7198
variance	5.0409	3.2818	28.0067
max	25	19	220

An improvement is to integrate the exact-convex scheduling principle into the GPS scheduler. For each request, the scheduler calculates an resource amount using the standard GPS algorithm. It compares this number with the past average of allocation for the request and then assigns the larger one. This modification incurs little overhead to the original GPS implementation.

The motivation for the modification is to keep the allocated amount in each scheduling epoch from randomly fluctuating. (Because GPS assumes no knowledge about request sizes.) Although there is no way to devise a strictly convex-structured scheduler, the resulting scheduler becomes close to uniform allocation, which is an approximate of the optimal scheduler for the input traffic with monotonically decreasing PSD functions.

We compare the convex-structured GPS scheduler with the original GPS scheduler in simulation. Table 4.6 listed the quality of service under the GPS schedulers for three traces used in previous experiments. The enhanced PS scheduler clearly outperforms the original PS scheduler. For all of the traces, the improvements are more than 50%, except the 30%

gain in maximum response time in HTTP traffic. The high variance in request response time indicates GPS scheduler would be a bad choice if stringent QoS is required by all requests. It also implies that filter function scheduler fits HTTP-typed traffic much better.

As seen from above optimality analysis, the PSD functions of input traffic have great impact on scheduler optimality and the server performance. For example, FGN traffic possesses LRD property that the other two traces do not have, or not as strong. Reflected in its PSD function, FGN trace has more power concentrated in lower frequencies. In above simulations, the impacts by different schedulers on FGN trace are among the least. It means that is harder to get server performance improved by adaptive scheduling.

4.6 Optimal Analysis of Sizing Factor

From Section 3.5, it shows that power spectrum of server workload is under influence of both request sizes and correlation in the traffic. In previous sections of this chapter, we have put major focus on the scheduling influenced by autocorrelation in the traffic. In this section, the relationship between sizing factor and scheduling policy will be analyzed. Recall the basic formula for sizing factor is:

$$\lambda\{E[|H_k(f)|^2] - |H(f)|^2\}, \tag{4.17}$$

where $H_k(f)$ is Fourier transform ($\mathbb{F}(\cdot)$) $h_k(t)$, the scheduling function for request k, arrived at t_k. $H(f) = \mathbb{F}\{h(t)\}$, with $h(t) = E_h[h_k(t)]$. Given that Fourier transform is linear operation, we have,

$$\begin{aligned} H(f) &= \mathbb{F}\{h(t)\} \\ &= \mathbb{F}\{E_h[h_k(t)]\} \\ &= E_h[\mathbb{F}(h_k(t))] \\ &= E_h[H_k(f)]. \end{aligned}$$

The sizing factor can be rewritten as based on above derivation:

$$E[|H_k(f)|^2] - |E[H_k(f)]|^2. \tag{4.18}$$

To minimize the overall variation, the sizing condition is expressed as

$$Min\{\int_0^{\pi} \{E[|H_k(f)|^2] - |E[H_k(f)]|^2\}df\}. \tag{4.19}$$

In this section, we will focus on boundary conditions that simplify problem 4.19. The first case is when scheduling functions $h_k(t)_k$ are the same for all the k's. Under this condition, $E[|H_k(f)|^2] \equiv |E[H_k(f)]|^2$.

According to theorem 4.3, a uniform scheduling function will minimize the sizing factor. This leads to following conclusion:

Corollary 4.2. *When scheduling function is time-invariant, and unified for all requests, the optimal scheduling function is the uniform scheduling function that allocation resources* *equally during the life cycle of the requests.*

Proof: Assuming that for a incoming traffic $\{w_k\delta(t-t_k)\}$, the scheduling functions are $h_k(t)$, where $h(t) \geq 0$ when $0 \leq t \leq t_d$, and $h(t) = 0$ otherwise.

Noticing that condition 4.19 becomes:

$$Min\{\int_0^\pi \{\overline{w_k^2}E[|H(f)|^2] - \overline{w_k}^2|E[H(f)]|^2\}df\}. \tag{4.20}$$

It is known that $E[|H(f)|^2] = |E[H(f)]|^2$, with which $\overline{w_k^2}|H(f)|^2 - \overline{w_k}^2|H(f)|^2$ can be rewriten as $(\overline{w_k^2} - \overline{w_k}^2)|H(f)|^2$.

From Theorem 4.3, when $h(\cdot)$ is a uniform function, it minimizes the condition. QED

Above conclusion, even in its simple format, directly leads to following theorem that demonstrates the theoretical relevance of assumption made for time-invariant and unified scheduling function.

Theorem 4.5. *For a system serving particular traffic $\{w_k\delta(t-t_k)\}$, for any scheduling policies expressed with a set of random scheduling function $h_k(t)$, there exists a time-invariant and unified scheduling function that provide lower bound in the variance for server workload process.*

Proof: let $h(t) = E[h_k(t)]$, and we let all requests be scheduled by $h'_k(t) = h(t)$. It can be proved that for two policies, the sizing factor in 4.18,

$$|E[H_k(f)]|^2 \equiv |E[H'_k(f)]|^2, \tag{4.21}$$

and,

$$E[|H_k(f)|^2] \geq E[|H'_k(f)|^2].$$

QED

This theorem shows that for any scheduling policies, we can find a time function scheduler that performs better in minimizing the variances of the server. Since variance is indicator of the tail of request responsive time, or sojourn time, it means that we can almost always find a decay function scheduler to approximate the tail and provide a tighter bound of any scheduling policies. The simulations (Figure 4.4 and 4.5) in previous section 4.2.3 demonstrate validness of this Theorem.

Under lossless service requirement, with the assumption in Corollary 4.2 for scheduling function, when system is overloaded, the requests will inevitably be delayed with queueing waiting. Assuming a special situation when one request $\tilde{k}$ is queued with waiting time Δ, and the rest of request are still scheduled by their original scheduling function $h_k(t)$. Using time-shift property of Fourier transform, we have

$$h_{\tilde{k}}(t) \equiv h(t-\Delta) \underset{\longleftrightarrow}{\mathbb{F}} e^{-jw\Delta} H(e^{jw}) \tag{4.22}$$

From property 4.22, queueing delay will not affect value of $E[|H_k(f)|^2]$, as $|e^{-jw\Delta}| \equiv 1$; queueing delay will reduce the value of $|E[H_k(f)]|^2$ given well known inequality:

$$|x + e^{jy}x| = |1 + e^{jy}| \cdot |x| \leq 2|x| = |x + x|. \tag{4.23}$$

This leads to an intrinsic view of queueing delay:

Theorem 4.6. *Queueing delay will increase the power of server workload process from the sizing factor under time-invariant and unified scheduling.*

Queueing delay will increase the variance from sizing factor. In the mean time, it is reducing the variance from correlation factor, as its impact is translated into reduction of $|E[H_k(f)]|^2$.

The queueing delay in reality happens in more complication situation than the condition specified in Theorem 4.6. Such as for Equation 4.23,

$$|x_1 + e^{jy}x_2| \nleq |x_1 + x_2|,$$

when $x_1 \neq x_2$. However, the model proposed here still gives a good measurement tool for performance indicator for the schedulers. In following example, we will show an example for the effect of queueing delay — when scheduling function and requests are different, how different queueing delay choices impact the sizing factor.

In traditional scheduling analysis, specifically, we use size-aware scheduling policy SRPT and commonly discussed first-in-first-out — FIFO as policies in the example.

Example 4.2. *Assuming a stream of requests, showing as Figure 4.11. The traffics are coming at three sizes —* 2 *at* $t(0)$, 10 *at* $t(1)$, *and* 1 *for the rest of 10 requests. The requests are served by a single server, with capacity serving size-*1 *during each time unit. We further assume that scheduling is none-preemptive.*

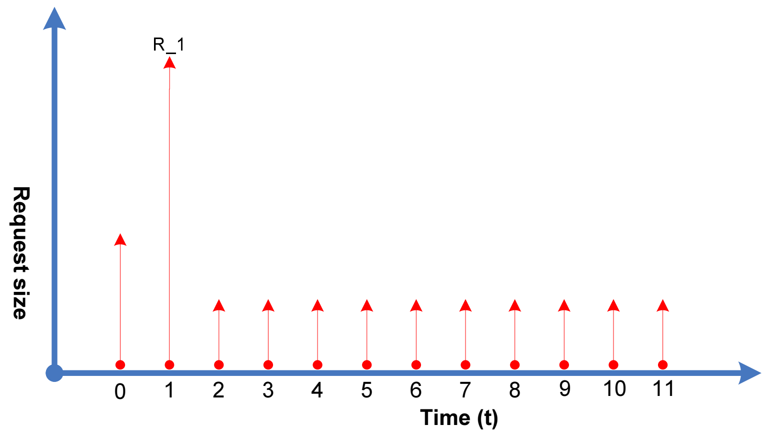


Figure 4.11: A sample piece of incoming traffic in the stream.

In this setup, the queueing delay will not have impact over factor $E[|H_k(f)|^2]$. *This can be easily verified with time-shift property of Fourier transform. We will focus on the* $|E[H_k^{SRPT}(f)]|^2$ *vs.* $|E[H_k^{FCFS}(f)]|^2$.

With this simple example, the scheduling function $E[h^{SRPT}(t)]$ *and* $E[h^{FCFS}(t)]$ *is illustrated in Figure 4.12. Derivation is simple — SRPT will delay the largest job* $t(1)$ *in this sequence by* 10 *time units, and then finish job* $t(1)$ *in next* 10 *time units. whereas FCFS will finish jobs in order of arrival, and thus delay all "size* 1*" jobs coming after* $t(1)$ *for* $10, 9, ...$ *to* 1 *time unit respectively. Corresponding* $|H(f)|$ *for SRPT and FCFS is calculated using FFT, and is shown in Figure 4.13. The accumulative power from scheduler* $A(f)$ *for SRPT and FCFS is shown in Figure 4.14.*

From this example, it shows that SRPT is superior in reducing the variances of server workload process from correlation factor, with the cost (increase) in the sizing factor.

Alternative, continuing on this example, when the request at time $t(1)$ *is size* 3 *— request size is less heavy-tail than previous set up. The* $H(f)$ *and* $A(f)$ *for SRPT and FCFS are*

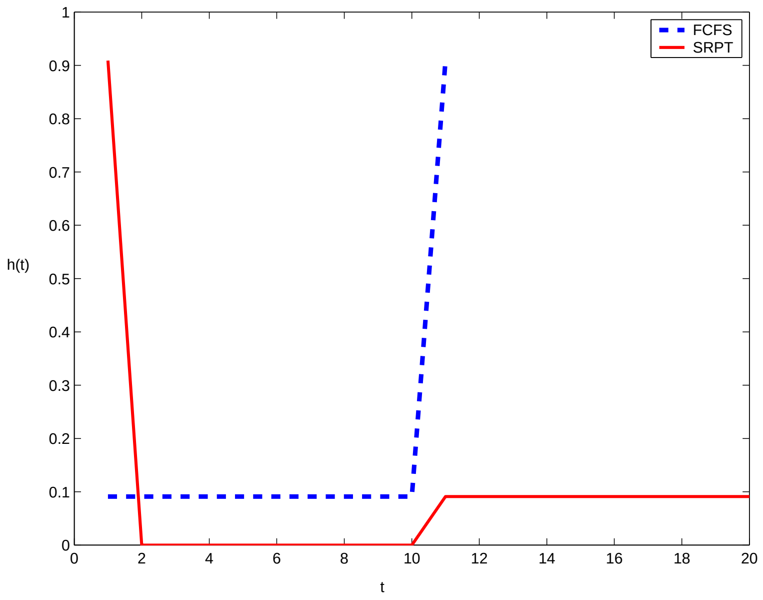


Figure 4.12: Average of decay function for SRPT and FCFS scheduling

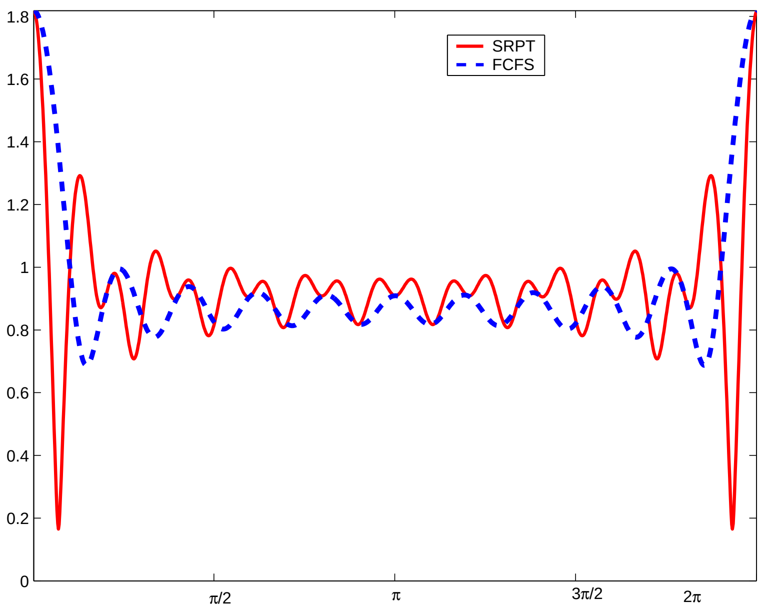


Figure 4.13: $|H^{SRPT}(f)|$ and $|H^{FCFS}(f)|$

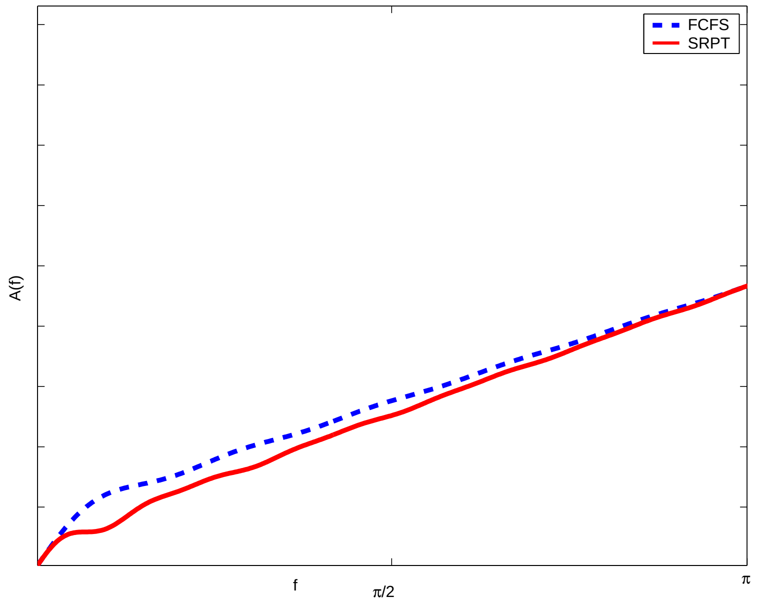


Figure 4.14: $|A^{SRPT}(f)|$ and $|A^{FCFS}(f)|$

shown correspondingly in Figure 4.15 and Figure 4.16.

It can be observed that in this situation, SRPT's correlation benefit is not as significant as heavy tailed situation, and verse vice for sizing factors. This can be seen from the shift of cut off of main-lobe from lower frequency to higher frequency. And in both examples, it shows that SRPT responds better to traffics with heavy power concentration in lower frequency bandwidth. In Chapter 6, we show that this corresponds to traffic with LRD features.

Above examples demonstrate the powerfulness of the decay function model in analyzing the real life scheduling policies. We also recognize that some theoretical analysis can be done in this approach to generalize the conclusions based on examples, which leads to further expansion of study and application decay function model.

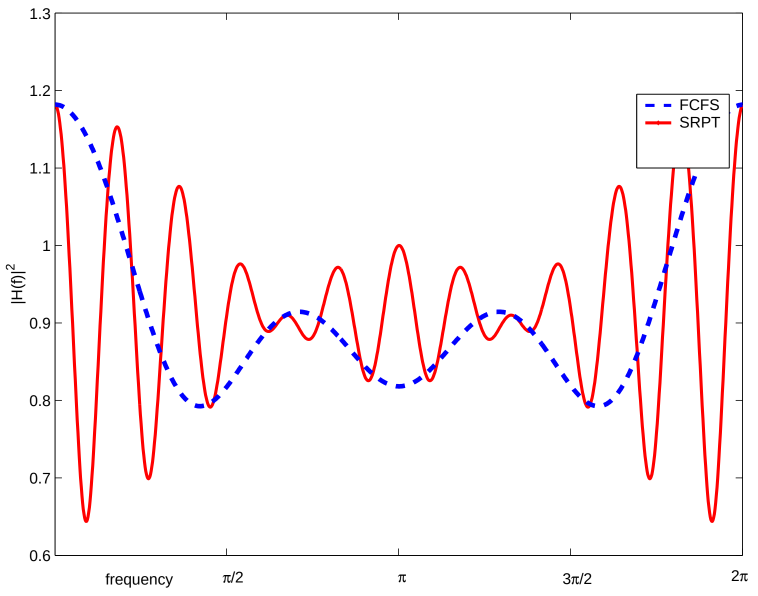


Figure 4.15: $|H^{SRPT}(f)|$ and $|H^{FCFS}(f)|$for less heavey tailed sizes

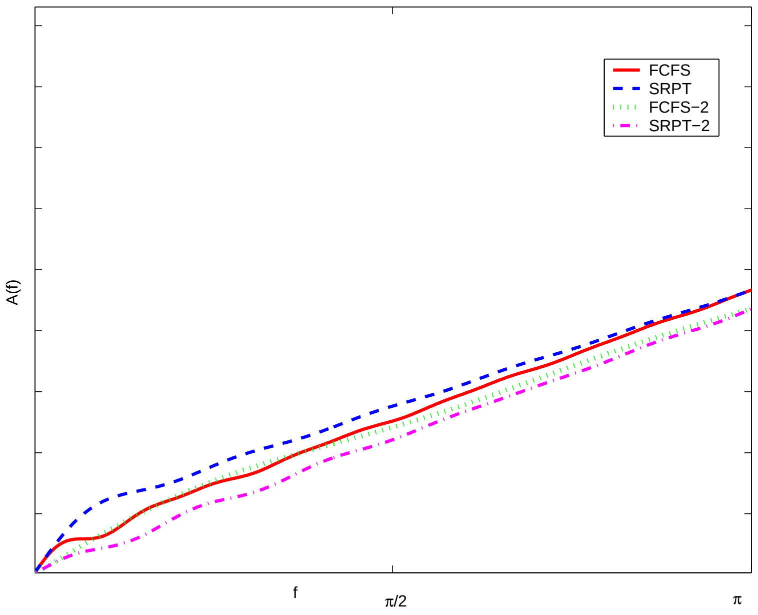


Figure 4.16: $|A^{SRPT}(f)|$ and $|A^{FCFS}(f)|$ for less heavy tailed sizes

Chapter 5

Service Differentiation and Adaptive Scheduling

In this chapter, we extend the decay function model in three aspects to further match against real world situations. First, with popularity of virtual machine technologies and $VMWare^{TM}$, it is common practice that service providers can now host multiple different types, or same application with different SLA on the same set of hots. We show that decay function model can be extended to support requests with different delay constraints and close expression for the optimal solution is derived as well.

Second, in order to simply the theoretical analysis, most of discussion made in previous sections are under time-invariant scheduling assumption. This chapter shows that by allowing the scheduler to be adaptive according to input traffic dynamics, we can further reduce the server load variance and hence the requirements for server resource. An algorithm is introduced to accomplish the calculation of such time-variant decay function scheduling. Simulation results show that while time-variant indeed provided more optimal resource utilization. On the other hand, it also shows that time-invariant scheduling is a close approximate to time variant version in probability of overloading and the variance of workload process.

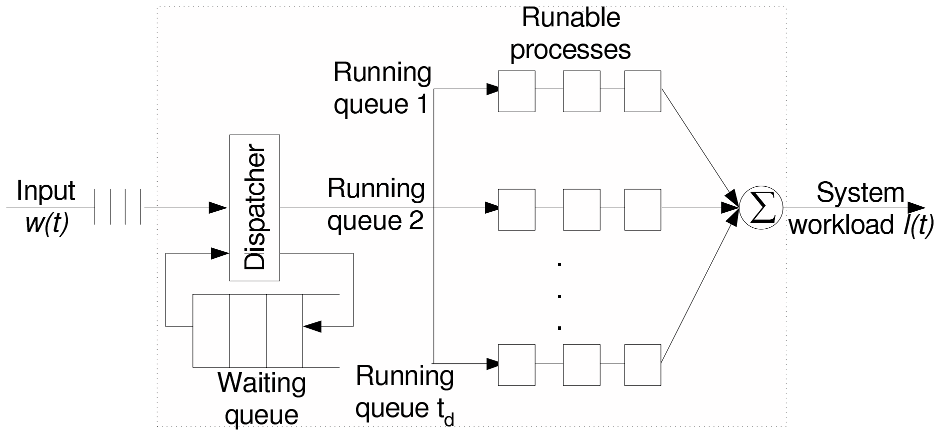


Figure 5.1: Decay function model with multiple queues to support service differentiation

5.1 System With Multiple Service Level Agreements

First, we extend the decay function model into a system with multiple running queue. This is very typical server farm provision technologies in industry with support of virtual machines and $VMWare^{TM}$. The extended system model can be viewed as the Figure 5.1.

Let t_d denote the supreme of the constraints of all the requests. We make t_d separate queues which contain requests with remaining delays 1, 2, ..., t_d. All requests with delay constraints j are put into queue j. Scheduling is conducted in all the t_d running queues. Requests in each queue are given a fractional-capacity on a full-time basis, which is similar to the processor-sharing concept in CPU scheduling. After all the scheduled requests have been served at the current scheduling slot, all requests with remaining delay *j* now have delay *j*-1. Thus queue 1 becomes empty and queues 2, 3,..., t_d become queue 1, 2, ..., t_d-1. Newly arrived inputs are dispatched into the queues according to their delay constraints. A request entering queue j at time t enters queue 1 at time $t + j - 1$ and out of the system at the beginning of $t + j$.

If an input request could result in system overload, this request should not be dispatched. We can either develop an admission control policy to decline the request or put it into a waiting queue to wait for the next scheduling slot. The choice can be made depending the type of the server. For example, if it is a media-server providing a stream of video frames at a certain rate. A frame that misses its deadline can be simply dropped because the application is very delay-sensitive. One can still tolerate a few missed deadlines without a significant degradation in video quality. For a Web contents server providing web contents, since customers may tolerate a little bit longer delays, a request that would miss its deadline can be put into the waiting queue temporarily.

Applying the decay function model to queue j, the load $l_j(t)$ becomes the output of aggregated inputs $\tilde{w}_j(t)$ passing through a system with resource allocation function $h_j(t)$. Define $\mathbf{h}'_j = [h_{j0}, h_{j1}, ..., h_{j(j-1)}]$ and $\mathbf{w}_j(t) = [\tilde{w}_j(t), \tilde{w}_j(t-1), ..., \tilde{w}_j(t-j+1)]$. The load at queue j can be expressed as

$$l_j(t) = \tilde{w}_j(t) * h_j(t) = \sum_{i=0}^{j-1} h_{ji}\tilde{w}_j(t-i) = \mathbf{h}'_j\mathbf{w}_j(t)$$
$$\text{subject to} \sum_{i=0}^{j-1} h_{ji} = 1 \; and \; h_{ji} \geq 0, \tag{5.1}$$

where h_{ji} means the proportion of resource allocated in queue j to the request $\tilde{w}_j(t)$ at time slot $t+i$.

The system load at time t, $l(t)$, is simply the sum of load over all the queues. Assume the load in each queue is independent, the variance of system load at time t is:

$$
\begin{aligned}
Var[l(t)] &= Var[\sum_{j=1}^{t_d} l_j(t)] = \sum_{j=1}^{t_d} Var[l_j(t)] \\
&= \sum_{j=1}^{t_d} E[l_j^2(t)] - \sum_{j=1}^{t_d}(E^2[l_j(t)]).
\end{aligned}
$$

Assume the input process to queue j is WSS. Then $E[l_j(t)] = E[\tilde{w}_j]$ is a constant. According to equation 5.1, we have

$$
E[l_j^2(t)] = E[\mathbf{h}_j'\mathbf{w}_j(t)\mathbf{h}_j'\mathbf{w}_j(t)] = \mathbf{h}_j'\Omega_j\mathbf{h}_j. \tag{5.2}
$$

We hence have the following result.

Theorem 5.1. *The optimization of server load variance due to a time-invariant decay function for input with different delay constraints is to*

$$
\begin{aligned}
&\textit{Minimize} \sum_{j=1}^{t_d} \boldsymbol{h}_j'\Omega_j\boldsymbol{h}_j \\
&\textit{subject to } \boldsymbol{c}_j\boldsymbol{h}_j = 1 \; and \; h_{ji} \geq 0,
\end{aligned} \tag{5.3}
$$

where $\boldsymbol{c}_j$ is a row vector with j components all equal to 1.

This is a typical minimization problem subject to a linear constraint and can be solved using Lagrange multipliers. We first ignore the non-negativity constraint and consider the

jth item in the Lagrangian,

$$L(h_j, \lambda_j) = \mathbf{h}_j' \Omega_j \mathbf{h}_j + \lambda_j(\mathbf{c}_j \mathbf{h}_j - 1).$$

Its gradient is $2\Omega_j \mathbf{h}_j + \lambda_j \mathbf{c}_j'$ with a solution $\mathbf{h}_j^* = -(\frac{\lambda_j^* \Omega_j^{-1} \mathbf{c}_j'}{2})$. To find the value of the Lagrange multiplier, the solution must satisfy the constraint. Imposing the constraint, the solution can be expressed as

$$\mathbf{h}_j^* = \frac{\Omega_j^{-1} \mathbf{c}_j'}{\mathbf{c}_j \Omega_j^{-1} \mathbf{c}_j'}. \tag{5.4}$$

A special case is when the input arrivals to the jth queue is independent, the covariance matrix becomes diagonal matrix and the resource allocation function is uniform with values $1/j$.

Note that the solution must be positive to satisfy the non-negativity constraint of equation 5.3. As Ω_j is a symmetric positive definite matrix, the denominator of the solution is always positive. However, the numerator is not necessarily positive. Since $\mathbf{c}_j$ is a constant for a given queue, the covariance matrix determines the sign of the numerator. There exists some covariance matrix that leads to negative solution. We show by simulation in Section 5.3.1 that the solution gives negative values only in extreme cases for synthetic input. The equation 5.4 provides an extremely simple solution for the resource allocation function for real world traffic. In case it produces a negative value, the solution can still be retrieved by solving the optimization problem in equation equation 5.3 directly.

5.2 Time Variant Solution to Decay Function Model

The resource allocation function discussed in the preceding section is optimal only in the set of feasible functions that do not change during the scheduling process. The load variance can be further reduced if $h(t)$ can be adapted according to input traffic in each scheduling epoch. In this section, we propose a time-variant scheduler and prove it to be optimal in the minimization of server load variance. Define q_j as the backlog of queue j. We formulate and solve the optimization problem in the following theorem.

Theorem 5.2. *The optimal delay bounded scheduling can be achieved by finding a scheduler that*

$$\textit{Minimize } L = \sum_{i=0}^{t_d-1}\Big(\sum_{j=i+1}^{t_d} l_j(i)\Big)^2$$

$$\textit{subject to } l_j(i) > 0,\ 0 \le i \le t_d - 1, 1 \le j \le t_d$$

$$\sum_{i=0}^{j-1} l_j(i) = q_j,\ 1 \le j \le t_d, \tag{5.5}$$

The optimal solution can be achieved by the following process for each queue j, $1 \le j \le t_d$,

$$l_j(i) = (\mu_j - \sum_{k=i+1}^{j-1} l_k(i))^+,\ 0 \le i \le (j-1), \tag{5.6}$$

where the notion $(x)^+ = max(x, 0)$ *and* μ_j *is determined by evaluating the backlog constraint in equation 5.5.*

Proof: We first show that the optimization solution can be achieved by the optimization of L. Recall that for a WSS input process, we have $E[l(t)] = E[\tilde{w}]$ and it is a constant. The variance of system load is

$$Var[l(t)] = E[l^2(t)] - (E[l(t)])^2 = E[l^2(t)] - (E[\tilde{w}])^2.$$

Therefore, the optimization problem is to find a scheduler that minimizes the second moment $E[l^2(t)]$.

At any of the scheduling slot, the scheduler needs to consider the current queue backlog including new arrivals in the last time slot and history arrivals not yet finished. To complete all the requests in the t_d queues, the second moment it generates can be estimated as

$$E[l^2(t)] = \frac{1}{t_d}\sum_{i=0}^{t_d-1} l^2(i) = \frac{1}{t_d}\sum_{i=0}^{t_d-1}(\sum_{j=i+1}^{t_d} l_j(i))^2. \tag{5.7}$$

It is an average of the square of load in the next t_d time-slots summed over all queues. It means the minimization of the second moment of load can be achieved by minimizing the L from equation 5.5. The solution to the problem gives optimal load for all queues from the current time to the next $(t_d - 1)$ time slots. The sum of load from all queues at the current time, $\sum_{j=1}^{t_d} l_j(0)$, forms the optimal solution for current load. However, for all other time, the solutions are not necessarily optimal if there are new input arrivals after the current time. It is because the arrival may change the queue backlog. In this case, the optimization process needs to be performed again to get the optimal load.

The minimization objective in equation 5.5 is a standard constrained optimization problem. It can be solved by using Lagrange Multipliers. For all i and j, we form the Lagrangian

$$L(l_j(i), \lambda_j) = \sum_{i=0}^{t_d-1}(\sum_{j=i+1}^{t_d} l_j(i))^2 - \lambda_j(\sum_{i=0}^{j-1} l_j(i) - q_j)$$

and differentiate it with respect to $l_j(i)$. Applying the Kuhn-Tucker conditions [57], we have

$$\begin{aligned}
&\sum_{j=i+1}^{t_d} l_j(i) - \frac{1}{2}\lambda_j = 0 \\
&\sum_{k=i+1}^{j-1} l_k(i) + l_j(i) + \sum_{k=j+1}^{t_d} l_k(i) - \frac{1}{2}\lambda_j = 0 \\
&l_j(i) = \frac{1}{2}\lambda_j - \sum_{k=j+1}^{t_d} l_k(i) - \sum_{k=i+1}^{j-1} l_k(i). \qquad (5.8)
\end{aligned}$$

The condition in equation 5.8 gives the solution for queue j at time i, depending on values of all other queues. This dependence makes the condition hard to solve.

Define the variable $\mu_j = \frac{1}{2}\lambda_j - \sum_{k=j+1}^{t_d} l_k(i)$ for each queue j. The condition equation 5.8 becomes

$$l_j(i) = \mu_j - \sum_{k=i+1}^{j-1} l_k(i).$$

Consider the fact that the load $l_j(i)$ is nonnegative. This leads to the condition in equation 5.6 with an unknown value μ_j. The value can be determined by the constraint in equation 5.5 for queue backlog q_j. Therefore, we can get the values of μ_j and $l_j(i)$ that satisfy the condition in equation 5.8. Note that equation 5.8 only gives necessary conditions for the existence and solution of a global minimum. Since the optimization problem we address is a convex

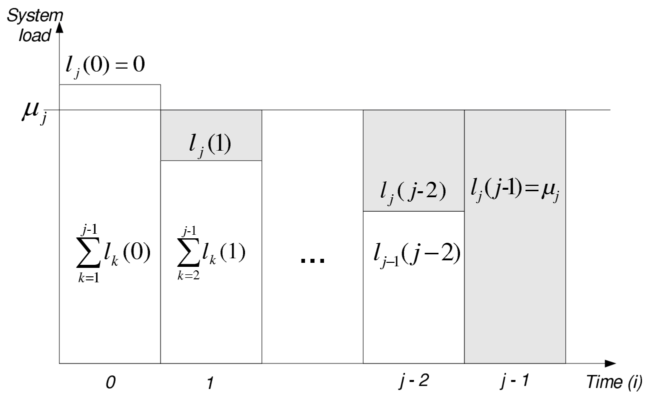


Figure 5.2: Illustration of the optimal time-variant resource allocation for queue *j*.

function, the conditions are also necessary. □

The optimal solution of equation 5.5 can be illustrated graphically in Figure 5.2. For queue j at time i, starting from $i = 0$ for simplicity, the vertical white area indicates the current accumulated load from queue *i+1* to *j-1*. As input size increases from zero, the requests are first distributed to the time slot with the lowest accumulated load, denoted by the shaded areas. The first time slot to be distributed is always $j - 1$. As the input increases further, parts of the requests will be put into busier slots. Once the load in a queue reaches the level denoted by μ_j, no more requests will be distributed to that queue in the current scheduling slot. This process in which requests are distributed among different time slots is similar to the way in which water is distributed in a vessel. It was initially referred to as a “water-filling” process when distributing power among a set of parallel Gaussian channels for maximum information capacity [16]. The water-filling process has been applied to various fields. A recent application is the minimization of average transmit power over Gaussian channels where the water-filling process is interpreted as a time-variant low-pass filter [37].

The optimal solution can be fit into the decay function model as time-variant coefficients in equation 5.1. The coefficients from current to the next (j-1) time slots for queue j can be written as:

$$h_j(t+i) = \frac{(\mu_j - \sum_{k=i+1}^{j-1} l_k(t+i))^+}{\tilde{w}_j(t)}. \tag{5.9}$$

When all requests have the same delay constraints t_d, the process only needs to be performed at the queue t_d. Load in other queues remains the same. Since the solution depends on accumulated values of all queues with smaller delays, the complexity of the solution is $O(t_d)$.

When requests have different delay constraints, the requests may enter different queues instead of only queue t_d. By aggregating input requests with identical delay constraints into to the same queue, the input process becomes less bursty and more controllable. All queues with new arrivals during the last time slot need to follow the solution process to get the new scheduled load. The worst case is when requests arrived in the last time slot have delays ranges from 1 to t_d. Then the complexity of the process is $O(t_d^2)$.

5.3 Performance Evaluation

In this section, we verify the previous analysis and evaluate the proposed scheduler through simulation. The first objective was to verify the effectiveness of the proposed scheduler in minimizing server load variance. When using the time-invariant scheduler, we assume the input correlation structure is known. No assumption about the input is made when using the time-variant scheduler. We also investigated the bound tightness due to both general and special cases.

5.3.1 Minimization of Server Load Variance

The first experiment was conducted based on traces from the 1998 World Soccer Cup server logs (WorldCupTrace) [3]. To show performance of the schedulers under traffic with different burstiness, we chose two days logs: day 17 and day 27. Their Hurst parameters are 0.9 and 0.72 respectively calculated by using the variance-time plot method [17]. The Hurst parameter H is used to denote the traffic burstiness. The higher is the Hurst parameter, the greater is the burstiness of the traffic. It means the day 17 trace is more bursty. To determine the request size of each request, we first retrieve the file size of each HTTP request and assume a proportional relationship between the file size and actual request size. Each request with a file size of 1K bytes is assumed to require one unit of resource under consideration.

Figure 5.3 shows the workload variance with different time constraints for day 27 ($H = 0.72$). We assume each request has the same delay constraint for simplicity. We observe that both schedulers are effective in reducing the impact of input burstiness to $1/t_d$ of the original variance. The variance due to the optimal time-variant scheduler is consistently smaller. For example, when t_d is 10 time units, the time-variant scheduler reduces the variance 20% more than the time-invariant schduler. We can also observe that as the delay constraints increase, the variances given by both methods decrease. This is expected because the schedulers have more time slots to *smooth* the input requests so the load variance becomes smaller. With the increase of delay constraint, the time-variant process has more time slots to adapt resource allocation according to the inputs so as to achieve smaller load variance. As a result, it can benefit slightly more from the increase of delay constraints.

Figure 5.3 also shows the server load variance for the trace of day 17 ($H = 0.9$). We observe that as the traffic becomes more bursty, the benefit by time-variant decay function become less obvious. Intuitively, the time-variant decay function achieves the optimal bound by adapting resource allocation according to the input requests to minimize load variance. When the requests get more bursty, the requests may cause sudden resource demands on the server. In this case an adaptation of resource allocation becomes less effective in keeping the load in small variance. We also observe from the figure that for the trace of day 17, the system load variance decreases slower with the increase of delay constraint than that of day 27. This can be verified from the fact that the presence of self-similarity makes it less effective to smooth the input with longer delay.

We also examined the system load based on a synthetic inputs from FGN processes with different Hurst parameters settings. The load variance and relative performance of the two schedulers show the same trend. Please refer to [86, 85] for more details.

It is interesting to compare the resource allocation functions of the two types schedulers. In the time-variant scheduler, the coefficients of the resource allocation function are adaptive according to the input. For comparison, we record the coefficients by all input requests and take the average to see the overall trend of resource allocation. The solution for the time-

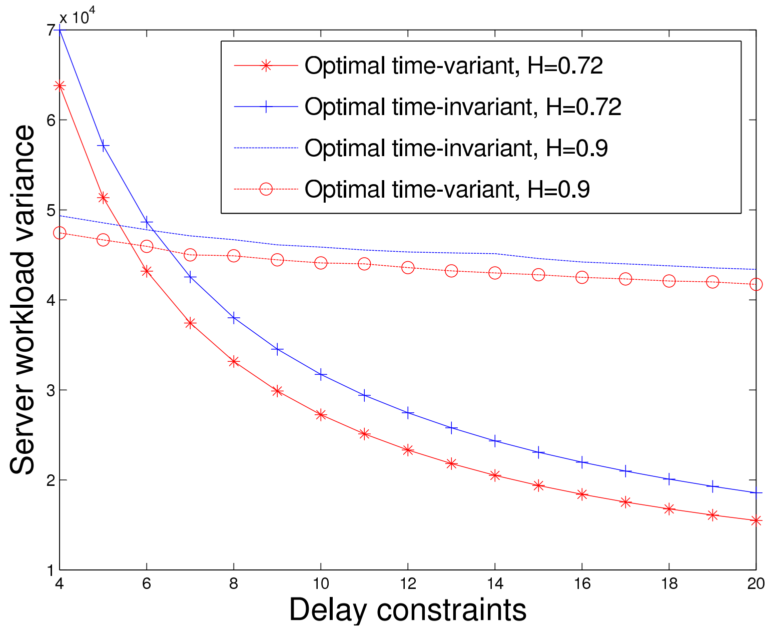


Figure 5.3: Impact of Delay Constraints with WorldCupTrace.

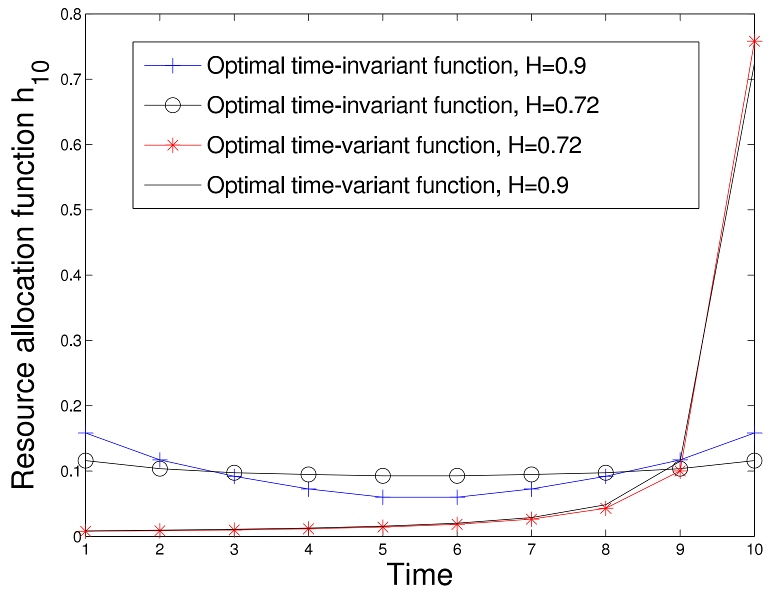


Figure 5.4: Comparison of the resource allocation functions.

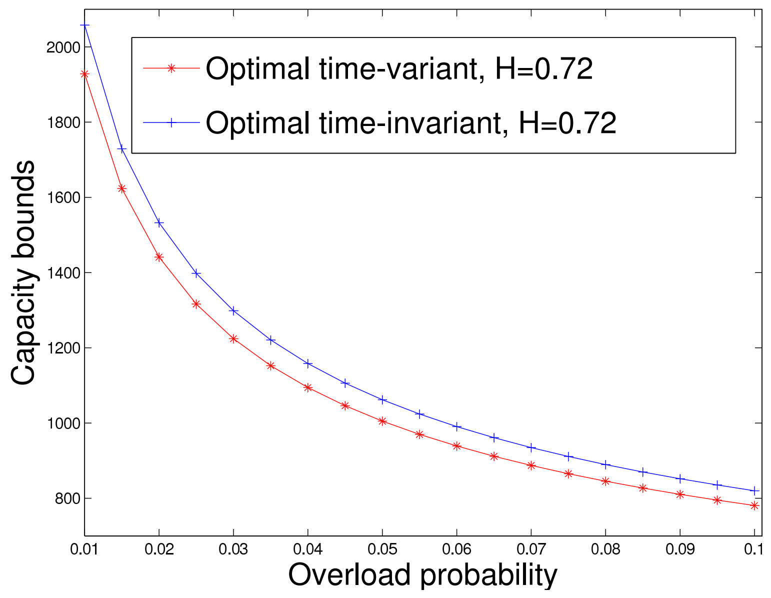


Figure 5.5: Impact of server overload probability with WorldCupTrace.

invariant function is attained by (5.4). We tried the same two traces as previous experiment and fixed the delay constraints to be 10. Figure 5.4 shows the results. The resource allocation denotes the percentage of resource in each time slot. It can be observed that the two functions are very different. In the time-invariant function, $h(t)$ is relatively stable with each value being around $1/10$. While in the time-variant scheduling, the $h(t)$ increases slowly in the beginning and abruptly in the last time slot. This shows the scheduler attempts to postpone the resource allocation as long as possible subject to the delay constraint. An intuitive explanation for this is as the deadline approaches, a request should have a higher priority to be processed so as to improve its chance of meeting deadlines. This can also be verified by (5.9) which shows that with a certain input, the coefficients are proportional to the predicted load at queue j in the next i time slots. With the increase of i, system load from less number of queues is accumulated, from queue $i + 1$ to $j - 1$. Therefore, the average coefficients show a trend of increase with time slot i. As time slot i becomes $j-1$, system load becomes the maximum μ_j and the coefficient also reaches its maximum $\mu_j/\tilde{w}(t)$.

We also observe that as the traffic burstiness varies, there is only subtle change in the optimal time-variant function. In the time-invariant case, the minimum value of the function becomes smaller. We tried traces from other days in the WorldCupTrace and another real scene-based MPEG-I video trace from the popular Simpson cartoon clip [66]. We found for all real traces the solution was always positive. We only got negative values for FGN process with Hurst parameter larger than 0.999, when the input is extremely correlated.

5.3.2 Comparison with Different Scheduling Policies

Using the same set of parameters as of last experiment, we compared both the optimal time-invariant and time-variant decay function scheduling with other schedulers. We set overload probability as 1% and delay constraint as 10. To compare the scheduling by a tight capacity configuration, we set the capacity to 171000 as shown from Figure 6.4.

The server was simulated with an infinite queue so that we can find exact distribution for the completion time of each request. The simulated server treats all requests with the same priority. When a request arrives at the server, it is first buffered in the queue. If the server has enough resource, the request will be scheduled for processing immediately without any queueing delay. Otherwise, the request will be blocked in the queue. We implemented two other different schedulers on the server module. One is GPS scheduling which ensures fair-sharing of resource between requests in processing at any scheduling epoch. In addition,

Table 5.1: Statistics of server with various scheduling algorithms

Scheduling	GPS	Random	Time-invar.	Time-var.
Load Mean	37534	37534	37534	37534
Load Var. (10^9)	1.68	1.00	0.894	0.71
Capacity Satisf.	99.8%	99.2%	99.3%	99.6%
Time Mean	4.25	10.13	10.08	10.04
Time Var.	5.38	0.51	0.33	0.19

we also implemented a heuristic time-variant *randomized allocation* algorithm. It takes t_d random numbers from a uniform distribution and then scales them to standard decay functions $h(t)$ so that $\sum_{i=1}^{t_d} h(i) = 1$. Table 5.1 shows the comparative results in terms of load mean and variance, completion time, and server capacity satisfactory rate. Completion time of a request includes waiting time in the backlogged queue, plus its processing time. Capacity satisfactory rate is the percentage of server workload less than the pre-defined capacity.

From the table, it can be observed that the optimal time-variant scheduling outperforms the others in two aspects. First, it dramatically reduces the variance of server load by up to 42% and 30% in comparison with GPS and random scheduling, respectively. Second, it provides better guaranteed completion time. GPS scheduling leads to a lower mean completion time and a high variance. Both the optimal schedulers guarantee completion time and yield much smaller time variance comparison with the other schedulers. In agreement with the findings in previous evaluation, the table also shows that the optimal time-variant scheduler consistently outperforms the time-invariant one.

Chapter 6

Applications of Model

In this chapter, we apply the analytical results derived from preceding chapters to solve the problems we posed in the Introduction. The first question is that given an arrival process and fixed-time scheduling, what server capacity should be configured so as to satisfy a predefined quality requirement.

6.1 Basic Formulas for Performance Boundaries

With *a priori* known mean and variance of the server load $l(t)$, we can estimate the probability distribution tail by Chebyshev's inequality [26].

Lemma 6.1. *The upper bound of the probability of workload* l *exceeding capacity* **c**, $prob(l > \mathbf{c})$, *is*

$$\frac{\sigma_l^2}{\sigma_l^2 + (\mathbf{c} - \mu_l)^2}. \tag{6.1}$$

Proof: From Chebyshev's inequality, it is known that

$$F\{I\} \leq a^{-1}E(u(y)),$$

where y is random variable, I is an interval, $F\{I\}$ is distribution function, $u(y) \geq 0$, $u(y) > a > 0$ for all y in I.

Substitute y with workload l, and define $u(l) = (l + x)^2$ with $x > 0$. It can be verified that $u(l) \geq 0$ and $u(l) > (\mathbf{c} + x)^2$ for $l > \mathbf{c} > 0$. Therefore,

$$prob(l > \mathbf{c}) \leq \frac{1}{(\mathbf{c} + x)^2} E((l(t) + x)^2).$$

Since

$$E((l(t) + x)^2) = \sigma_l^2 + \mu_l^2 + 2x\mu_l + x^2,$$

we have

$$prob(l > \mathbf{c}) \leq \frac{1}{(\mathbf{c} + x)^2}(x^2 + 2x\mu_l + \sigma_l^2 + \mu_l^2).$$

It can be proved that the right side of the inequality takes the minimum value at $x = -\mu_l + \sigma_l^2/\mathbf{c}$. This proves the lemma. QED

From this lemma, we have an estimate of server capacity as follows.

Theorem 6.1. *Let v be a pre-defined bound of the probability that workload exceeds the server capacity. The estimate of capacity is*

$$\mathbf{c} = \sqrt{\frac{1 - v}{v}} \cdot \sigma_l + \mu_l. \tag{6.2}$$

This theorem tells that the capacity is determined by the mean and variance of server load. Because of the high variability of the Internet traffic, the workload variance is often

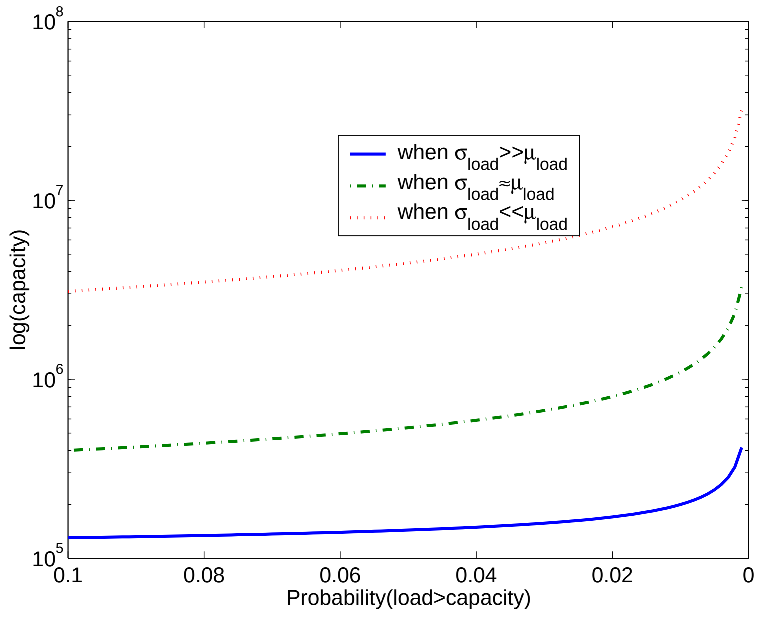


Figure 6.1: The capacity and chance of overload

a dominating factor in the calculation formula of the resource capacity. The relationship between server capacity, the chance of overload (workload exceeding capacity), and workload mean and variance is illustrated in Figure 6.1. From the figure, it can be seen that the capacity requirement increases sharply with setting of high criteria v, and this increase is amplified in the situations of high variance of workload. Note that Chebyshev's inequality provides a loose upper bound of $prob(l > \mathbf{c})$. In reality, depending on different underlying distributions, there could exist more accurate estimates of the bound. A tighter bound would be able to provide more precise estimate capacity or deadline. In this paper, we focus on general situations, where Chebyshev's inequality provides a uniform solution that applies to all distributions with finite first and second order moments.

Corollary 6.1. *If the compounded input process $\tilde{w}(t)$ is independent for different time t,* *then the relationship of length of deadline, capacity, chance of failure and input statistics can be characterized by equation*

$$t_d = \frac{(1-v)\sigma_{\tilde{w}}^2}{v(\mathbf{c}-\mu_{\tilde{w}})^2}. \tag{6.3}$$

Proof: According to Corollary 4.1, the function $h(t)$ should be set to a constant value $1/t_d$ for period $0 < t \leq t_d$ for minimizing the server load variance. Thus,

$$\sigma_l^2 = \sigma_{\tilde{w}}^2 \sum_{t=-\infty}^{t=\infty} h^2(t) = \frac{\sigma_{\tilde{w}}^2}{t_d}.$$

From Theorem 6.1, we know that

$$\mathbf{c} = \sqrt{\frac{1-v}{v}} \cdot \sigma_l + \mu_l.$$

Combining the two equations completes the proof. QED

This corollary shows that QoS (in terms of completion time) can be adapted in response to the change of traffic intensity. To keep the same level of QoS for traffic with different means and variances, say (μ_1, σ_1^2) and (μ_2, σ_2^2), the ratio of their completion time is

$$\frac{t_{d_2}}{t_{d_1}} = \frac{\sigma_2^2 \cdot (\mathbf{c}-\mu_1)^2}{\sigma_1^2 \cdot (\mathbf{c}-\mu_2)^2}. \tag{6.4}$$

When $\mathbf{c} \gg \mu_1$ and $\mathbf{c} \gg \mu_2$, their ratio approximates σ_2^2/σ_1^2.

Example 6.1. *The second example assumes the number of request arrivals at each time* *conforms to a lognormal distribution* $ln(1, 1.5)$*, the size of request fits a normal distribution* $n(5000, 1000)$*. Also, the sever uses fixed-time scheduling.*

1. *What is the resource capacity necessary to be configured so as to keep the deadline of each request within 10 time units and the chance of failure below* 1%*?*
2. *Given a server capacity of 400,000, what kind of deadline can the server guarantees if it wants to keep the chance of failure below* 1%*?*

Answer to Question 1: From properties of a random sum distribution, we know that

$$\begin{aligned}
E(\tilde{w}) &= E(ln(1, 1.5))E(n(5000, 1000)), \\
Var(\tilde{w}) &= E(ln(1, 1.5))Var(n(5000, 1000)) \\
&\quad +(E(n(5000, 1000))^2 Var(ln(1, 1.5))),
\end{aligned}$$

which leads to

$$\begin{aligned}
\mu_{\tilde{w}} &\approx 41,864, \\
\sigma^2_{\tilde{w}} &\approx 1.4884e + 10.
\end{aligned}$$

According to Corollary 4.1 , the workload variance

$$\sigma_l^2 = \sigma_{\tilde{w}}/10 \approx 1.4884e + 9.$$

Thus the server capacity can be calculated according to Theorem 6.1 as

$$\mathbf{c} \approx \sqrt{99} \cdot \sqrt{1.4884e+9} + 41,864 \approx 425,730.$$

From this calculation, we see that for a high variable traffic, the server capacity requirement is mainly determined by its variance, Purely mean-value analysis is insufficient for the resource configuration and allocation problem.

Answer to Question 2: According to Corollary 6.1, the target deadline can be calculated as

$$\begin{aligned} t_d &= \frac{(1-v)\sigma_{\tilde{w}}^2}{v(\mathbf{c}-\mu_l)^2} \\ &\approx \frac{0.99 \cdot (1.4884e+10)}{0.01 \cdot (400,000-41864)^2} \\ &\approx 12. \end{aligned} \tag{6.5}$$

The above calculations show the applicability of our model in practical situations. In the next session, we will also show through more examples that when workload distribution tail is a lognormal tail, the estimate of $t_{d_2}/t_{d_1} \approx \sigma_2^2/\sigma_1^2$ in 6.4 is very precise.

6.2 Special Cases for Tight Capacity Bounds

In this section, we consider two special cases to tight the bound in (6.2) when more information about the input distribution is assumed.

We first show that a better bound can be derived if the distribution of the load is unimodal distributed. In (5.1), the load is modeled as the convolution between input and the resource allocation function. Assume the input distribution is continuous and its density g is logconcave (i.e., $log(g)$ is concave). The logconcave property can be found in several widely used standard distributions such as Poisson distributions, Gaussian distributions. For example, Gaussian distributions are commonly used as the marginal distributions in modeling self-similar process with FGN and F-ARIMA [45]. The set of logconcave distributions are strongly unimodal [22] in the sense that they are closed under convolutions and hence the resulted load is unimodal distributed.

Let X be a random variable with a unimodal distribution, meaning that for some mode m, its CDF is concave on $[m, \infty)$ and convex on $(-\infty, m]$. From the Vysochanskiî and Petunin inequality [72], we can derive the probability that X exceeds k as:

$$P(X \geq k) \leq \begin{cases} \frac{4E[X^2]}{9(E[X^2]+k^2)} & \text{if } k^2 \geq \frac{8}{3}E[X^2], \\ \frac{4E[X^2]}{3(E[X^2]+k^2)} - \frac{1}{3} & \text{if } k^2 \leq \frac{8}{3}E[X^2]. \end{cases} \tag{6.6}$$

Applying the inequality (6.6), the probability that system load l exceeds capacity c becomes

$$\begin{aligned} P(l \geq c) &= P(l - \mu_l \geq c - \mu_l) \\ &\leq \begin{cases} \frac{4\sigma_l^2}{9(\sigma_l^2+(c-\mu_l)^2)} & \text{if c} \geq \sqrt{\frac{5}{3}}\sigma_l + \mu_l, \\ \frac{4\sigma_l^2}{3(\sigma_l^2+(c-\mu_l)^2)} - \frac{1}{3} & \text{if c} \leq \sqrt{\frac{5}{3}}\sigma_l + \mu_l. \end{cases} \end{aligned}$$

Recall that the probability is denoted as v. The upper capacity bound can be expressed as

$$c \leq \begin{cases} \frac{2}{3}\sqrt{\frac{1-v}{v}}\sigma_l + \mu_l & \text{if } 0 < v \leq \frac{1}{6} \\ \sqrt{\frac{3-3v}{3v+1}}\sigma_l + \mu_l & \text{if } \frac{1}{6} < v < 1. \end{cases} \tag{6.7}$$

Usually the probability v should be well below $1/6$ for an Internet server to provide acceptable performance. The first bound is thus more useful. In both cases, the capacities are much tightened in comparison with the bound from 6.2.

We next consider the case when the number of arrival requests at each time and their sizes are i.i.d. random variables. For independent input, the load distributions can be determined according to the input distributions and delay constraints. The implication of deriving the distribution is a sharp capacity bound can be calculated directly by considering the system load tail distribution.

Let the size of each request $w_1(t)$, $w_2(t)$, ... at time t be mutually independent random variables with a common distribution $f(w)$. Recall that $n(t)$ is the number of requests arrived from time $t-1$ to t. The aggregated input $\tilde{w}(t) = \sum_{i=1}^{n(t)} w_i(t)$ has the density $f(w)^{n(t)*}$, namely the $n(t)$-fold convolution of $f(w)$ with itself. Denote $L(t)$, $N(t)$, $W(t)$, and $\tilde{W}(t)$ as the random variables of $l(t)$, $n(t)$, $w(t)$, and $\tilde{w}(t)$. Let $P(N(t) = n(t)) = p_n$.

The PDF of the resulting random sums is the density of the aggregated input $\tilde{W}(t)$ [26]

$$f(\tilde{w}(t)) = \sum_{n=1}^{\infty} p_n f^{n(t)*}(w(t)).$$

By using Equation 3.19, the distribution of the load at time t is

$$f(l(t)) = \frac{1}{t_d} f^{t_d *}(\tilde{w}(t)) = \frac{1}{t_d}(\sum_{n=1}^{\infty} p_n f^{n(t)*}(\tilde{w}(t)))^{t_d *}. \tag{6.8}$$

A simpler approach to get the output distribution is to use characteristic functions. Let $\Phi_{W(t)}(\omega)$ be the characteristic function corresponding to the random variable $W(t)$ and $G_{N(t)}(z)$ be the generating function of $N(t)$. For distributions that the functions exist, the characteristic function of the aggregated input $\tilde{W}(t)$ can be retrieved as [26]

$$\Phi_{\tilde{W}(t)}(\omega) = E[z^{N(t)}(t)]|_{z=\Phi_{W(t)}(\omega)} = G_{N(t)}(\Phi_{W(t)}(\omega)).$$

That is, the characteristic function of $\tilde{W}(t)$ can be found by evaluating the generating function of $N(t)$ at $z = \Phi_{W(t)}(\omega)$.

The characteristic function of load $L(t)$ at time t can be derived as

$$\Phi_{L(t)}(\omega) = (\Phi_{\tilde{W}(t)}(\frac{\omega}{t_d}))^{t_d} = (G_{N(t)}(\Phi_{W(t)}(\frac{\omega}{t_d})))^{t_d}.$$

The PDF of the output load at time t is found by taking the inverse transform of the above expression.

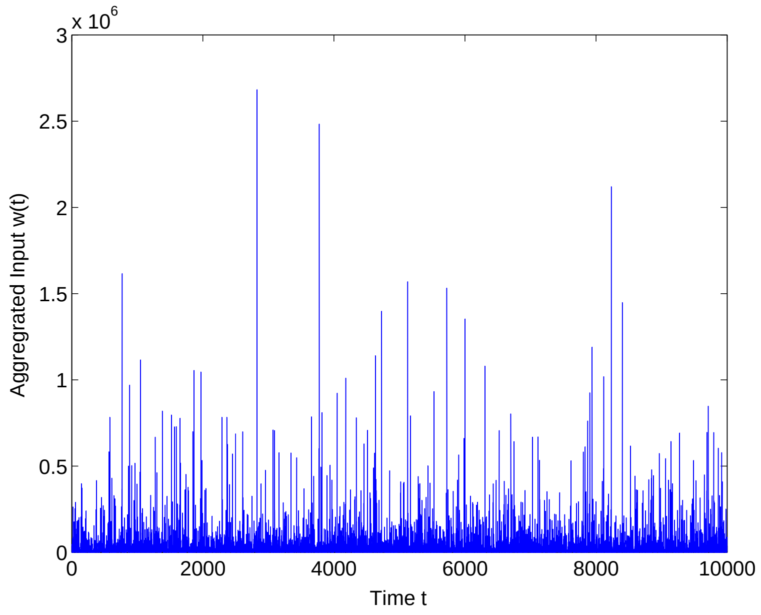


Figure 6.2: The compounded input $\tilde{w}(t)$

6.3 Simulation

We carried on the experiment with same setup as the one in Chapter 4, Section 4.2. In this experiment, we are intended to uncover the inherent properties of fixed-time scheduling. The simulated server deployed the optimal fixed-time scheduling algorithm with a request completion deadline of 10 time units ($t_d = 10$). We used the lognormal $ln(1, 1.5)$ and normal $n(5000, 1000)$ distributions to generate input traffic. We wish to keep the percentage of server load being less than its capacity $\%_{\mathbf{c}} \leq 99\%$, as the setting in Example 6.1. The compounded input pattern is shown in Figure 6.2.

Table 6.1 gives the statistics of the simulated server according to the Chebyshev estimation in Lemma 6.1. The 100% satisfactory rates in the table with respect to both of the server load and completion time show the conservativeness of this estimation because of the

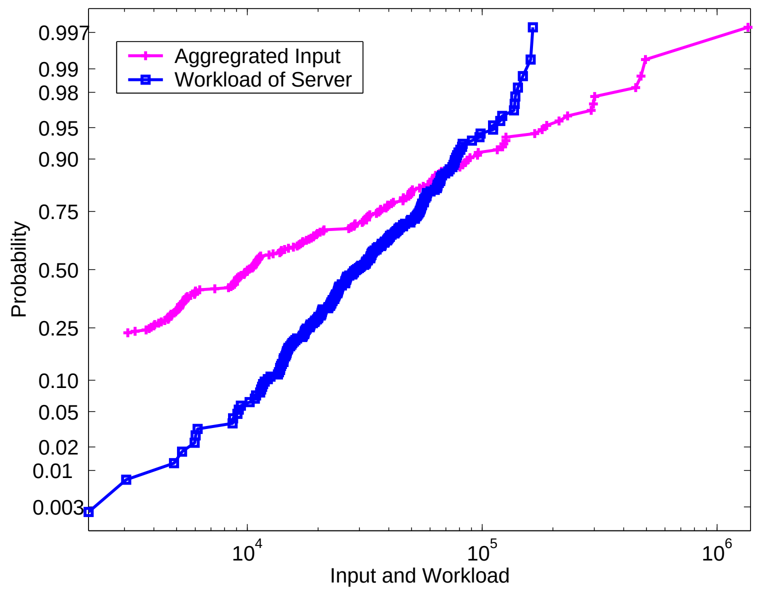


Figure 6.3: Normal plots of l and $\tilde{w}$

Table 6.1: Statistics of server with optimal fixed-time scheduling. ($\mathbf{c} = 425730$, $n(t) \sim ln(1, 1.5)$, $w \sim n(5000, 1000)$, $t_d = 10$)

Performance	mean	variance	percentage
Server Load	37534	1.0402×10^9	100%
Completion Time	10	0	100%

target 99% server capacity satisfactory rate.

As we mentioned in the preceding section, the Chebshev estimation can be tightened by taking into consideration of the empirical distribution tail. Figure 6.3 shows the normal plots of compounded input $\tilde{w}(t)$ and server workload $l(t)$. Note that the normal plot for samples from a normal distribution is a straight line. The figure shows both $log(\tilde{w})$ and $log(l)$ are close to normal distributions in the tail. However, the slope of the workload plot is steeper than that of the compounded request. This indicates that a scheduling algorithm

Table 6.2: Statistics of server with various scheduling algorithms. ($\mathbf{c} = 171000$, $n(t) \sim ln(1, 1.5)$, $w \sim n(5000, 1000)$, $t_d = 10$)

Scheduling	GPS	Incremental	Random	Optimal
Load Mean μ_l	37534	37534	37534	37534
Load Variance σ_l^2 (10^9)	1.68	1.22	1.00	0.894
Capacity Satisf. $\%_c$	99.8%	96.7%	99.2%	99.3%
Time Mean μ_t	4.25	10.08	10.13	10.08
Time Variance σ_t^2	5.38	0.70	0.51	0.33
Deadline Satisf. $\%_t$	96.7%	90.6%	97.0%	97.3%

is essentially a "smoother" of fluctuating traffic that reduces server load variance and leads to better schedulability and QoS assurance.

We used the estimated lognormal tail with distribution $ln(10.3, 0.74)$ to approximate the server workload. From this distribution, the estimated capacity is 171,000. The simulation results are shown in the last column of Table 4.2. From the table, it can be seen that the 99.3% capacity satisfactory rate is in agreement with the initial 99% rate in simulation setting. Because of occasional overloads in the simulation, the queueing delay leads to only 2.7% of requests that missed their deadlines.

In this experiment, we estimate the capacity required by the two types of scheduling in order to guarantee a specified delay constraint according to (6.2). Figure 5.5 shows the capacity bounds with different server overload probability. We fixed the delay constraint as 10 time units and used the trace at day 27. It is expected that the capacity requirement becomes higher with lower deadline miss rate, characterized by a lower overload probability. A close look at the data also tells that with a lower overload probability, the time-variant decay function provides a little bit tighter capacity bound. This can be explained by the fact that the time-variant scheduling is best at minimizing load variance. With the decrease of v, more resource is requirement. According to (6.2) and the fact the load mean is a constant, the variance plays a more important role in determining the capacity bound. As a result, the time-variant decay function provides a better bound. However, the difference is only about 3% as v varies from 0.1 to 0.01 and it is hard to be observed from the figure.

As predicted that for unimodal distributed traffics, a tigher bounds exists. We compare the next experiment to show the effectiveness of the bounds for general input (6.2), tight bounds for unimodal input (6.7) in Figure 6.4. We assumed the number of request arrivals at each time conforms lognormal distribution $ln(1, 1.5)$, the size of request fits normal distribution $n(5000, 1000)$. We get consistent results using the time-variant and the time-invariant decay functions with previous experiments. We therefore only present the results of time-invariant decay function. The normal distribution is strongly unimodal and the marginal distribution of the aggregated input process is strongly unimodal, as well. The input is slightly correlated due to the default random number generator in use and the correlation structure is used to derive the coefficients in the decay function which are not exactly uniformly distributed. The system load is therefore a convolution between the aggregated inputs; the load distribution is guaranteed to be unimodal distributed. We can then apply the bound in (6.7). Figure 6.4 also includes bounds retrieved by approximating the tail distribution of the system load through simulation. From the figure, it is expected that the general bounds are loose. By use of the unimodal density, we reduce the bounds to 72% of the general bounds.

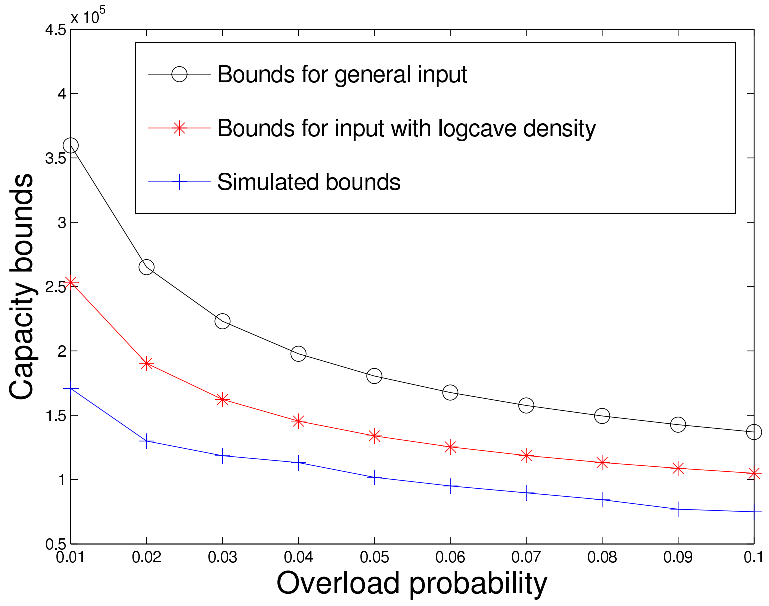


Figure 6.4: Capacity bounds for inputs with unimodal distribution.

6.4 Performance Prediction

In this experiment, we show that the server performance will degrade with more arrival requests. Interesting questions are how much degradation there will be and how much more resource the server needs to keep the same level of QoS. Answers to the first question will provide a higher customer satisfaction because people is likely to have more tolerance in performance degradation if they are informed of extra delay. Answers to the second question can enable Internet service providers to predict the needs for increasing capacity on their servers.

Although Chebyshev estimations of absolute server capacity and request completion time are conservative, their relative predictions are precise. Following the previous experiment with the number of requests coming in lognormal distribution $ln(1, 1.7)$, we wish to predict the server capability to assume service completion time.

From the remarks of Corollary 6.1, we can calculate the new request completion time t_{old} for $ln(1, 1.7)$ distribution based on the old completion time t_{old} for $ln(1, 1.5)$ as:

$$t_{new} = \frac{\sigma^2_{new}}{\sigma^2_{old}} \cdot t_1 \approx \frac{5.65e10}{1.4884e10} \cdot 10 \approx 38.$$

We ran simulations with a new deadline of 38 time units under the optimal fixed-time scheduling algorithm. The results are shown in Table 6.3. The simulation data shows that in response to the change of incoming traffic, more than 98% of the requests can meet their adjusted deadline of 38 and that there were less than 1% of chances for the server getting overload.

Table 6.3: Statistics of server with optimal fixed-time scheduling. ($\mathbf{c} = 171000$, $n(t) \sim ln(1, 1.7)$, $w(t) \sim n(5000, 1000)$, $t_d = 38$)

Performance	mean	variance	percentage
Server Load	52497	7.0448×10^8	99.3%
Completion Time	38.04	0.2	98.1%

If the server wants to keep the original deadline of 10 time units, it needs to be configured with more resource. According to 6.1, we have

$$1 = \frac{\sigma^2_{new} \cdot (\mathbf{c}_{old} - \mu_{old})^2}{\sigma^2_{old} \cdot (\mathbf{c}_{new} - \mu_{new})^2}.$$

As a result, the new server capacity $\mathbf{c}_{new} = 313000$.

We ran simulations based on this new predicted capacity. The experimental results in Table 6.4 show that with an adjusted server capacity 313000, 96% of the requests would meet their deadlines and the server would rarely become overloaded. Again, this demonstrates the prediction accuracy of Corollary 6.1.

Table 6.4: Statistics of server with optimal fixed-time scheduling. ($\mathbf{c} = 313000$, $n(t) \sim ln(1, 1.7)$, $w(t) \sim n(5000, 1000)$, $t_d = 10$)

Performance	mean	variance	percentage
Server Load	52497	2.733×10^9	99.4%
Completion Time	10.17	1.25	96.0%

6.5 Impact of LRD on Server Performance

Internet traffic has an important property of LRD. In [52], the author proved that LRD has negative impacts on the server performance by using the Hurst parameter. Using queueing models to study its impact has limitations, LRD is defined by ACF or PSD of the input traffic. Queueing model is unable to provide direct input and output relationship based on ACF and/or PSD functions themselves. One must build a random process model, and fit LRD into the process. This raises two immediate questions, (1) how to measure the LRD in the original traffic? (2) which model is more proper for fitting LRD traffic? The first question has been proved to be difficult, because LRD indicates lower rate of convergence in statistical measurement, and it even causes the divergence in classical confident interval analysis[15]. Numerous models, such as FGN, FARIMA, M/G/∞, Wavelet, and even chaotic map, were proposed to address second question[9, 73]. However, no common guidelines are available for when to use them. Theoretical results from one model cannot be borrowed directly by researches on different traffic models. Consequently, there are often contradictory results about effects of LRD on the server.

In this section, we use our filter model to study the impact of LRD on server performance from the perspective of power distribution in the frequency domain. It is more fundamental than the Hurst parameter approach, independent of parameter estimation. The result is also applicable to all LRD traffic models. Estimating PSD function from random process has been a throughly studied and matured area in digital signal processing, and is proved to be an easier and more reliable task than estimating Hurst parameter.

In the frequency domain, LRD is characterized by a highly condensed power in low frequencies. Asymptotically, a LRD process has the PSD function in the following format:

$$P_w(f) \sim C_p|f|^{-\beta}, \tag{6.9}$$

as $f \to 0$ for some constant C_p and a real number $\beta \in (0,1)$. β is related to Hurst parameter η and $\beta = 2\eta - 1$.

By (3.14), we know the power of server utilization process is,

$$P_l(f) = P_w(f)|H(f)|^2. \tag{6.10}$$

Its CPF function at $f = \pi$ is,

$$A_l(\pi) = \int_0^{\pi} P_w(f)|H(f)|^2 df. \tag{6.11}$$

Because $P_w(f)$ in (6.9) has a singular point at $f = 0$ for LRD traffic, for a given ϵ, $\epsilon \in (0, \pi)$ and $\epsilon \to 0$, (6.11) can be rewritten as:

$$\begin{aligned} A_l(\pi) &= \int_0^{\epsilon} P_w(f)|H(f)|^2 df + \int_{\epsilon}^{\pi} P_w(f)|H(f)|^2 df \\ &= C(\epsilon) + \int_{\epsilon}^{\pi} P_w(f)|H(f)|^2 df. \end{aligned} \tag{6.12}$$

Because $P_w(f)$ is monotonically decreasing, based on our convex scheduling policy, there exists an optimal scheduling function $h^*(t)$ that leads to a smaller $A_l(\pi)$ than all other scheduling functions, including uniform scheduling $h_u(t)$. In light of these, (6.12) becomes,

$$\begin{aligned} & A_l(\pi) = C(\epsilon) + \int_{\epsilon}^{\pi} C_p f^{-\beta} |H^*(f)|^2 df \\ & \leq C(\epsilon) + \int_{\epsilon}^{\pi} C_p f^{-\beta} |H_u(f)|^2 df \\ & = C(\epsilon) + \frac{\Theta}{t_d^2} \int_{\epsilon}^{\pi} \frac{sin^2(\frac{ft_d}{2})}{f^{2+\beta}} df, \end{aligned} \tag{6.13}$$

where t_d is service deadline. Model and traffic constant parameters are summarized into Θ. Because $f^{-2-\beta}$ is a non-increasing function of f, and $f^{-2-\beta} > 0$ for $f > 0$, there exists an $X \in (\epsilon, \pi)$, such that,

$$\begin{aligned} & \int_{\epsilon}^{\pi} \frac{sin^2(\frac{ft_d}{2})}{f^{2+\beta}} df = \epsilon^{-2-\beta} \int_{\epsilon}^{X} \frac{sin^2(\frac{ft_d}{2})}{t_d^2} df \\ & = \epsilon^{-2-\beta} \frac{B(t_d, X, \epsilon)}{t_d^2}, \end{aligned} \tag{6.14}$$

assuming $B(t_d, X, \epsilon) \equiv \int_{\epsilon}^{X} sin^2(ft_d/2) df$. The value of B is a definite integration of function $sin^2()$. Using the theorem of mean value in definite integration, $B \sim (X - \epsilon) sin^2(\lambda)$, where $\lambda \in (\epsilon, X)$. When ϵ is considerably smaller than X, the value of B can be treated as a constant.

When a server is optimized for LRD input traffic, we assume there exists a uniform scheduler h'_u with service deadline $t'_d \in (t_d, 2t_d)$, and the CPF of h'_u being smaller than the CPF of optimal scheduler h^* in the mainlobe. That is:

$$A_{h^*}(f) \geq A_{h'_u}(f). \tag{6.15}$$

From the property of edge scheduler in Theorem 3.3, we know that the width of main lobe for schedulers with service deadline t_d is lower bounded by $\pi/(t_d - 1)$, which corresponds to the mainlobe width of the uniform scheduler with a service deadline $2t_d - 2$. From Theorem 3.2, we can then conclude the existance of the h'_u. Figure 6.5 gives an example of the upper bound and the lower bound of uniform schedulers for the optimal scheduler.

Similar to (6.13), we have the following inequality as a lower bound for $A_l(\pi)$:

$$\begin{aligned}
A_l(\pi) &= C(\epsilon) + \int_\epsilon^\pi C_p f^{-\beta} |H^*(f)|^2 df \\
&\geq C(\epsilon) + \int_\epsilon^\pi C_p f^{-\beta} |H'_u(f)|^2 df \\
&= C(\epsilon) + \frac{\Theta'}{t'^2_d} \int_\epsilon^\pi \frac{sin^2(\frac{f t'_d}{2})}{f^{2+\beta}} df. \\
&= C(\epsilon) + \frac{\Theta'}{t'^2_d} \epsilon^{-2-\beta} B(t'_d, X', \epsilon).
\end{aligned} \tag{6.16}$$

Putting the upper and lower bounds together, we have:

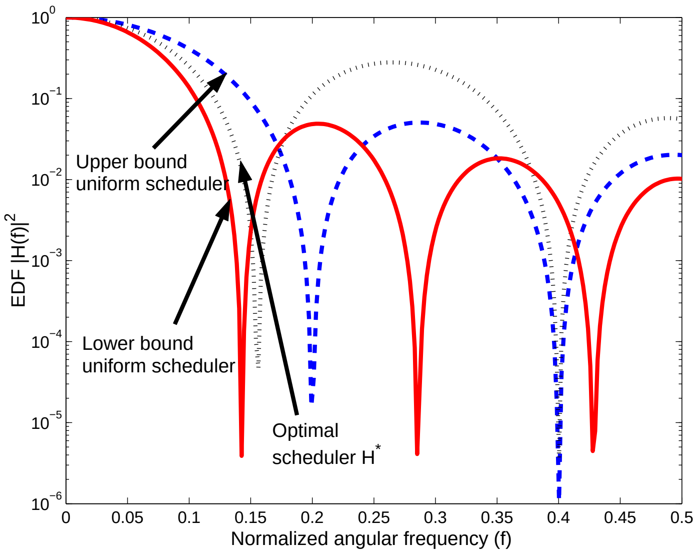


Figure 6.5: The illustration of upper bound and lower bound for server optimal scheduling function. Through convex scheduling and minimum power theorems, an optimal scheduler can be bounded by the uniform scheduler with same service deadline and a uniform scheduler with longer service deadline.

$$C(\epsilon) + \Theta' \epsilon^{-2-\beta} t_d^{-2} B(t'_d, X', \epsilon) \leq A_l(\pi)$$
$$\leq C(\epsilon) + \Theta \epsilon^{-2-\beta} t_d^{-2} B(t_d, X, \epsilon). \qquad (6.17)$$

Unlike previous results from queueing theorems, where the performance boundary is estimated asymptotically for tail of the buffer size distribution, our bounds are established on the power of server utilization process. Clearly, for pure LRD traffic, this power approaches to ∞ as $\epsilon \to 0$. For real Internet traffic, PSD function can never take an infinite value at zero frequency. This means that we can treat ϵ as the first starting point that PSD function begins the power-rate decreasing in (6.9).

Several conclusion can be drawn from these bounds directly:

Theorem 6.2. *When PSD function starts power-law decreasing at very low frequency $\epsilon < 1$, the stronger LRD, or higher value in β, will increase the average power in server utilization process with a relationship of $\epsilon^{-2-\beta}$.*

Since the condition for Theorem 6.2 states that ϵ is in lower frequency, it is equally to say that by increasing LRD, or value β, the power of server utilization process will increase in lower frequency. Also notice that as the PSD starts decreasing at the higher frequency, or ϵ is getting larger, the negative impact of LRD will becoming smaller, or disappear when $\epsilon \geq 1$.

Theorem 6.3. *When service deadline t_d increases, the power for server utilization will decrease.*

Finally, we remark that, in reality, Internet traffic cannot have singular point at $f = 0$. The value of ϵ becomes crucial to server performance evaluation. In another word, when evaluating log-log plot of the PSD function of the traffic, it is the feature at the lower-frequency end that dominates the performance of the queueing server. There has been little discussion about impact of value ϵ on server performance in the past. When using PSD function to estimate Hurst parameter of a traffic trace, a common mistake is to fit the straight line across the entire frequency domain, by ignoring the skewness at lower frequency.

Chapter 7

Summary

The target of our research is to construct a flexible and solvable model to facilitate future study of Internet server under diversified traffic environment. We have successfully built the model, with demonstration of its capabilities of predicting optimal scheduling policy, planning server capacity, and provisioning QoS of service deadline. This part of result was summarized in paper:“Decay Function Model for Resource Configuration and Adaptive Allocation on Internet Servers”, and was published in IWQoS’04.

We extend our model into frequency domain. By using Fourier transform, the PSD of server utilization process is the filtering output of the PSD of input traffic by the server scheduler. Similar to time domain, the existence of a minimum-power scheduler is identified in frequency domain, further more, the structure of the optimal scheduler must be “convex” for traffic with monotonically decreasing PSD functions. This condition in PSD function has proved to be very a very general situation in Internet. It has strong implication in designing and optimizing scheduling algorithms in Internet servers. We gave an example of enhancing GPS scheduler using the policy. With basic properties of scheduler and optimality, we derived performance bounds for average power in server utilization process under asymptotically second-order self-similar traffic. The previous research on the LRD traffic was focused on the Hurst parameter. Our model demonstrated that it can start directly from definition of LRD or self-similarity instead of using this parameter. This has important implication because Hurst parameter has been proved to be hardly measurable in practice. Above results were summarized into paper "Frequency Domain Filter Design and Analysis for Optimal Scheduling in Internet Servers", paper was published in ICDCS'08 and journal version in ITPDS'10.

The extension for the model to multiple service disciplines and time variant is accomplished with similar optimal conditions and corresponding algorithms to derive the optimal scheduling policies. The application of the model results for server performance evaluation, such as capacity planning and overload probability prediction, is also extended by narrowing the boundary conditions for unimodal distributions and for models with probability characteristic functions. These results were published in RTAS'05.

When breaking the compounded traffic model into modulated marked random point process, further detail between sizing factor and correlation factor in the traffic and their interaction with scheduling functions is revealed. As optimality condition for correlation factor is derived, corresponding optimality study was conducted in this research for sizing factor. It reveals that SRPT scheduling policy actually tends to reduce more in correction factor, while reducing the sizing factor, and which explains its optimality over other scheduling policies as studied in previous literatures. These results were summarized in technical report for CIC'2010.

The current result clearly indicates the bright future of using this model to study broad range of performance problem in Internet environment, either independently or as supplement to queueing models.

7.1 Future Work

There are challenges and open questions that deserve attention for future work along the line of this dissertation. Even with the extension to service differentition domain, our model is still covering a simplifed environment and topology for Internet server. For example, in reality, a request arriving at Interent server is normally served by multiple special service endpoints, with each one of them serving a subset of functions. Further, the topologies of Internet servers can even make cascading execution of the tasks in a pipeline, or fork the sebset of tasks at different endpoints in parallel. For this domain of problems, the model proposed in this dissertation has special advantage. For cascading services, we can have:

$$l(t) = \tilde{w}(t) * (h_1(t) * h_2(t)) = (\tilde{w}(t) * h_1(t)) * h_2(t). \tag{7.1}$$

We can see that Equation 7.1 can easily simplify the complicated problem into either a combination of multiple smaller systems, or a convolution operation sequencially, with one system's workload becoming another system's input. Similiar for parallel processing situation, the problem can be converted into:

$$l(t) = \tilde{w}(t) * [h_1(t) + h_2(t)] = \tilde{w} * h_1(t) + \tilde{w} * h_2(t). \tag{7.2}$$

However, when optimizing for operation cost or capacity cost under a guaranteed time scheduling situation, how to balance the allocation function accross different service endpoints can be an interesting problem.

Another area deserves attention is the influences to the variance of server workload process. In the dissertatin, we have derived that power spectral density function of server workload is influenced by three expressions:

$$\lambda E[|H_k(f)|^2], \tag{7.3}$$

$$\lambda |E[H_k(f)]|^2, \tag{7.4}$$

and

$$\lambda u_w^2 P_r(f) |E[H_k(f)]|^2. \tag{7.5}$$

These three factors clearly decide, (1) how does scheduling function influence the second order statistics of server? (2) how to optimal the sizing factor in Expression 7.3? and (3) how does $P_r(f)$ relate to the average of arrival process λ? We show that potentially these factors combined provides a good tool for studying traffic correlation and sizing continuously. classification in size and correlation

REFERENCES

[1] M. F. and C. L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5), oct 1997.

[2] T. F. Abdelzaher, K. G. Shin, and N. Bhatti. Performance guarantees for web server end-systems: A control-theoretical approach. *IEEE Transaction on parallel and Distributed Sysems*, 13(1):80–96, 2002.

[3] M. Arlitt and T. Jin. *Workload Characterization of the 1998 World Cup Web Site.* IEEE Network, vol. 14, No.3, pp.30-37, 2000

[4] Chenyang Lu, Ying Lu, Tarek F. Abdelzaher, John A. Stankovic and Sang Hyuk Son. *Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers.* IEEE Trans. Parallel Distrib. Syst., vol.17, No.9, pp.1014-1027, 2006

[5] P. Abry, D. Veitch. Wavelet analysis of long-rangedependent traffic. IEEE Transactions on Information Theory. vol. 44 pp. 2C15, 1998.

[6] A. Baiocchi and N. Blefari-Melazzi. Steady-state analysis of the mmpp/g/1/k queue. *IEEE Trans. Commun.*, pages 531–534, 1993.

[7] Barford, Paul and Crovella, Mark. *Measuring Web Performance in the Wide Area.* In Performance Evaluation Review, August 1999.

[8] S. Basu, A. Mukherjee and S. Klivansky. Time Series Models for Internet Traffic. In *IEEE INFOCOM* pp. 611 - 620, 1996.

[9] J. Beran. *Statistics For Long-Memory Processes*. Chapman and Hall, 1994.

[10] A. W. Berger and W. Whitt. Extending the effective bandwidth concept to networks with priority classes. *IEEE Communications Magazine*, Aug 1998.

[11] P. Bloomfield. Fourier Analysis of Time Series: An Introduction, 2nd Edition. John Wiley & Sons, 2000.

[12] Nina T. Bhatti, Anna Bouch and Allan Kuchinsky. *Integrating user-perceived quality into Web server design.* Computer Networks, vol.33, No.1-6, pp.1-16, 2000

[13] C. M. Cheng, H. T. Kung and K. S. Tan. Use of Spectral Analysis in Defense Against DoS attacks. In *Proc. of IEEE GLOBECOM 2002*

[14] L. Cherkasova, Y. Fu, W. Tang, and A. Vahdat. *Measuring and Characterizing End-to-End Internet Service Performance.* Journal ACM/IEEE Transactions on Internet Technology, (TOIT), November, 2003

[15] R. E. Clegg. *A Practical Guide to Measuring the Hurst Parameter*. International Journal of Simulation: Systems, Science & Technology 7(2) pp 3-14 2006

[16] Thomas Cover and Joy Thomas. *Elements of Information Theory* John Wiley & Sons, Inc. 1991

[17] Mark E. Crovella and Azer Bestavros. *Self-similarity in world wide web traffic: evidence and possible causes.* IEEE/ACM Transactions on Networking, vol.5, No.6, pp.835-846, 1997

[18] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic — evidence and possible causes. In *SIGMETRICS*, 1996.

[19] J. Cao, W.S. Cleveland, D. Lin, D.X. Sun. On the nonstationarity of Internet traffic. In *ACM SIGMETRICS, 2001, pp. 102C112*.

[20] T. D. Dang, S. Molnar, and I. Maricza. Queueing performance estimation for general multifractal traffic. *International Journal of Communication Systems*, 16(2):117–136, 2003.

[21] A. Dembo and O. Zeitouni. Large Deviation Techniques and Applications. Springer, 1998.

[22] Sudhakar Dharmadhikari. *Unimodality, convexity, and applications.* Academic Press, INC.,1988

[23] S. Elnikety, *et al.* A Method for Transparent Admission Control and Request Scheduling in e-Commerce WEb Sites. in *Proc. of WWW'04*

[24] D. H. J. Epema. Decay-usage scheduling in multiprocessors. *ACM Trans. on Computer Systems (TOCS)*, 16, 1998.

[25] A. Erramilli, O. Narayan, and W. Willinger. Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM Trans. on Networking*, 4(2):209–223, 1996.

[26] W. Feller. *An Introduction to Probability Theory and Its Applications (Vol II)*. John Wiley & Sons, Inc., 1971.

[27] L. L. Fong and M. S. Squillante. Time-function scheduling: A general approach to controllable resource management. In *SIGOPS*, 1995.

[28] K. Fukuda, L. A. N. Amaral, and H. E. Stanley. Dynamics of temporal correlation in daily Internet traffic. In *Global Telecommunications Conference*, 2003.

[29] Mingwei Gong; Williamson, C. *Simulation evaluation of hybrid SRPT scheduling policies* Modeling, Analysis and Simulation of Computer and Telecommunications, 2004 (MASCOTS'2004)

[30] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE J. Select Areas Commun.*, 9:968–981, 1991.

[31] , Mor Harchol-Balter, Nikhil Bansal and Bianca Schroeder, *Implementation of SRPT Scheduling in Web Servers*, ACM TRANSACTIONS ON COMPUTER SYSTEMS, 2000

[32] F. Hernndez-Campos, K. Jeffay, F.D. Smith. Tracking the Evolution of Web Traffic: 1995-2003. MASCOTS'2003.

[33] S. Jin and Z. Bestavros. Temporal locality in web request streams — sources, characteristics, and chaching implications. In *SIGMETRICS*, 2000.

[34] M. F. T. Karagiannis, M. Molle and A. Broido. A Nonstationary Poisson View of Internet Traffic. *Proceedings of IEEE INFOCOM*, Mar. 2004.

[35] F. P. Kelly. Effective bandwidths at multi-class queues. *Queueing Systems*, 9:5–16, 1991.

[36] G. Kesidis, J. Walrand, and C.-S. Chang. Effective bandwidths for multiclass markov fluids and other atm sources. *IEEE/ACM Tran. Networking*, 1:424–428, 1993.

[37] M. A. Khojastepour and A. Sabharwal. Delay-constrained scheduling: Power efficiency, filter design, and bounds. In *Proc. of the 23rd IEEE Infocom*, March 7-11 2004.

[38] M. M. Krunz and A. M. Ramasamy. The correlation structure for a class of scene-based video models and its impact in the dimensioning of video buffers. *IEEE Transactions on Multimedia*, 2(1):27–36, 2000.

[39] G. Latouche and V. Ramaswami. Introduction to Matrix Geometric Methods in Stochastic Modeling. ASA-SIAM Series on Statistics and Applied Probability. SIAM, Philadelphia PA, 1999.

[40] J. Y. Le Boudec and P. Thiran. *Network Calculus*. Springer Verlag Lecture Notes in Computer Science Volume 2050, 2001.

[41] W. Leland, M. Taqq, W. Willinger and D. Wilson. On the Self-Similar Nature of Ethernet Traffic. In *Proc of ACM SIGCOMM'93*, August 1993.

[42] S. Q. Li and C. L. Hwang. On the convergence of traffic measurement and queueing analysis: A statistical-matching and queueing(smaq) tool. *IEEE/ACM Transactions on Networking*, Feb 1997.

[43] S. Q. Li and C. Hwang. Queue Response to Input Correlation Functions: Continuous Spectral Analysis. In *IEEE/ACM Trans. Networking*, Vol. 1, No. 6, pp. 678-692, Dec. 1993.

[44] S. Q. Li and C. Hwang. Queue Response to Input Correlation Functions: Discrete Spectral Analysis. In *IEEE/ACM Trans. Networking*, Vol. 1 , No. 5, pp. 522-533, Oct. 1993.

[45] Jose Lopez-Ardao and Candido Lopez-Garcia and Andres Suarez-Gonzalez and Manuel Fernandez-Veiga and Raul Rodriguez-Ruibio. *On the Use of Self-Similar Processes in Network Simulation.* ACM Transactions on Modeling and Computer Simulation, vol.10, No.2, pp.125-151, 2000

[46] C. Lu, T. Abdelzaher, J. Stnkovic, and S. Son. A feedback control approach for guaranteeing relative delays in web servers. In *Proc. of the IEEE Real-Time Technology and Applications Symposium*, 2001.

[47] Y. Lu, T. Abdelzaher, and C. Lu. Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers. In *Proc. of the IEEE Real-Time Technology and Applications Symposium*, 2003.

[48] Dong Lu, Peter A. Dinda, Yi Qiao, Huanyuan Sheng and Fabin E. Bustamante. *Applications of SRPT Scheduling with Inaccurate Information.* Poster in Proc. of the 12th IEEE/ACM International Symposium on Modeling , Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), October 2004.

[49] S. Ma and C. Ji. Modeling heterogeneous network traffic in wavelet domain. In *Networking, IEEE/ACM Transactions on*, Vol. 9, Issue 5, pp. 634 - 649, 2001.

[50] D. G. Manolakis, V. K. Ingle, and S. M. Kogon. *Statistical and Adaptive Signal Processing*. McGraw-Hill Companies Inc., 2000.

[51] D. A. Menasce and V. A. F. Almeida. *Scaling for E-Business Technologies, Models, Performance, and Capacity Planning*. Prentice Hall Ptr., Prentice-Hall Inc., Upper Saddle River, New Jersey, 2000.

[52] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16:387–396, 1994.

[53] Andrew M. Odlyzko Internet traffic growth: Sources and implications Proc. SPIE, vol. 5247, pp. 1-15, 2003

[54] A.K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single node case. *IEEE/ACM Trans. Networking*, 1-3:344–357, Jun 1993.

[55] V. Paxson, S. Floyd. Wide Area Traffic: the Failure of Poisson Modeling. In IEEE/ACM Transactions on Network, 3(3), pp 226–244, 1995.

[56] V. Paxson. Fast approximation of self-similar network traffic. Technical report, EECS Division, University of California, Berkeley, 1995.

[57] A.L. Peressini and R.E. Sullivan and Jr. J.J. Uhl. *Convex programming and the Karish-Kuhn-Tucker conditions.* Springer-Verlag 1980

[58] James E. Pitkow. *Summary of WWW characterizations.* World Wide Web, v.2 n.1-2, p.3-13, 1999

[59] Idris A. Rai, Guillaume Urvoy-keller, Mary K. Vernon and Ernst W. Biersack. *Performance Analysis of LAS-based Scheduling Disciplines in a Packet Switched Network.* In SIGMETRICS/Performance 04, pp106-117.

[60] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. Practical solutions for qos-based resource allocation problems. In *proc. Real-Time Systems Symp.*, 1998.

[61] Dinesh Rajan and Ashutosh Sabharwal and Behnaam Aazhang. Delay-Bounded Packet Scheduling of Bursty Traffic Over Wireless Channels. IEEE Transactions on information theory, vol50, No. 1, pp. 125-144, 2004.

[62] Ritke, R., X. Hong, and M. Gerla. Contradictory Relationship between Hurst parameter and Queueing Performance (extended version). Telecommunication Systems, Vol 16 (1,2): pp. 159-175, Feb. 2001.

[63] V. Ribeiro, R. Riedi, M. S. Crouse, and R. G. Baraniuk. Multiscale queueing analysis of long-range-dependent network traffic. In *IEEE INFOCOM*, 2000.

[64] T. G. Robertazzi. *Computer Networks and Systems: Queueing Theory and Performance Evaluation*. Springer-Verlag New York Inc., 1990.

[65] K. W. Ross M. Reisslein and S. Rajagopal. A framework for guaranteeing statistical qos. *IEEE/ACM Trans. Networking*, 10-1:27–42, Feb 2002.

[66] O. Rose. Statistical properties of mpeg video traffic and their impact on traffic modeling in atm systems. In *Proc. of the IEEE 20th Conference on Local Computer Networks (LCN'95)*, pages 397–406, 1995.

[67] L. Sha, X. LU, Y. Lu, and T. Abdelzaher. Queueing model based network server performance control. In *Proc. of the 23rd IEEE Real-Time System Symposium*, 2002.

[68] L. P. Slothouber. A model of web server performance. In *Proc. of 5th Conference on WWW*, 1996.

[69] M. S. Squillante and L. Zhang. *WEb Traffic Modeling and Web Server Performance Analysis*. Research Report, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, 1999.

[70] A. Tanenbaum. *Modern Operating Systems.* Prentice Hall, 1992.

[71] ACM SIGCOMM. The Internet Traffic Archive. on *http://ita.ee.lbl.gov*

[72] D. F. Vysochanskij, Y. I. Petunin. Justification of the 3σ rule for unimodal distributions. Theory of Probability and Mathematical Statistics 21: 2536, 1980.

[73] X. J. Wang. *Statistical physics of temporal intermittency*. Phys. Rev. A, 40(11):6647-6661, 1989.

[74] Adam Wierman and Mor Harchol-Balter. *Classifying scheduling policies with respect to unfairness in an $M/GI/1$*, SIGMETRICS, pp.238-249 2003

[75] Adam Wierman, Mor Harchol-Balter and Takayuki Osogami. *Nearly insensitive bounds on SMART scheduling.* SIGMETRICS, pp.205-216 2005

[76] W. Willinger and V. Paxson. *Where mathematics meets the Internet.* Notices of AMS 45, pp 961-970.

[77] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. *Self-similarity through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level.* IEEE/ACM Transactions on Networking 5, pp. 7186, 1997.

[78] David Alderson, Lun Li, Walter Willinger and John C. Doyle *understanding internet topology: principles, models, and validation* IEEE/ACM Trans. Netw.,Vol 13(6), pp. 1205-1218, 2005.

[79] C. Xia and Z. Liu. Queueing systems with long-range dependent input process and subexponential service times. In *Proc. of the 2003 ACM SIGMETRICS*, pages 25–36, 2003.

[80] M. Xu and C. Z. Xu. Decay function model for resource configuration and adaptive allocation on Internet servers. In *12th IEEE Int. Workshop on Quality of Service (IWQoS'04).*

[81] Cheng-Zhong Xu, Minghua Xu, Le yi Wang and George Yin *Filter Design and Analysis in Frequency Domain for Server Scheduling and Optimization* IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 21, 2010

[82] Minghua Xu and Cheng-Zhong Xu Frequncy domain filter design and analysis for optimal scheduling in Internet servers. ICDCS'08.

[83] M. Xu and C.-Z. Xu. *I/O pipeling algorithms for fast classification in data mining* Proc. of the 12th IASTED Int'l Conference on PDCS, November 2000

[84] Z.-L. Zhang, V. Ribeiro, S. Moon, C. Diot. Small-time scaling behaviors of Internet backbone traffic: an empirical study. In *IEEE INFOCOM, vol. 3, pp. 1826C1836*, 2003.

[85] Xiliang Zhong and Cheng-Zhong Xu and Minghua Xu", Optimal Time-variant Resource Allocation for Internet Servers With Delay Constraints. Technical report, Department of Electrical and Computer Engineering, Wayne State University, 2004

[86] X. Zhong, C. Xu, M. Xu and J. Wei *Optimal time-variant resource allocation for Internet servers with delay constraints.* Proc. of the 11th IEEE Real-time and Embedded Technology and Applications Systems (RTAS'2005), 2005

ABSTRACT

FILTER SCHEDULING FUNCTION MODEL IN INTERNET SERVER: RESOURCE CONFIGURATION, PERFORMANCE EVALUATION AND OPTIMAL SCHEDULING

by

MINGHUA XU

December 2010

Advisor: Dr. Cheng-Zhong Xu

Major: Computer Engineering

Degree: Doctor of Philosophy

Internet traffic often exhibits a structure with rich high-order statistical properties like self-similarity and long-range dependency (LRD). This greatly complicates the problem of server performance modeling and optimization. On the other hand, popularity of Internet has created numerous client-server or peer-to-peer applications, with most of them, such as online payment, purchasing, trading, searching, publishing and media streaming, being timing sensitive and/or financially critical. The scheduling policy in Internet servers is playing central role in satisfying service level agreement (SLA) and achieving savings and efficiency in operations. The increasing popularity of high-volume performance critical Internet applications is a challenge for servers to provide individual response-time guarantees. Existing tools like queuing models in most cases only hold in mean value analysis under the assumption of simplified traffic structures.

Considering the fact that most Internet applications can tolerate a small percentage of

deadline misses, we define a decay function model characterizes the relationship between the request delay constraint, deadline misses, and server capacity in a transfer function based filter system. The model is general for any time-series based or measurement based processes. Within the model framework, a relationship between server capacity, scheduling policy, and service deadline is established in formalism. Time-invariant (non-adaptive) resource allocation policies are design and analyzed in the time domain. For an important class of fixed-time allocation policies, optimality conditions with respect to the correlation of input traffic are established. The upper bound for server capacity and service level are derived with general Chebshev's inequality, and extended to tighter boundaries for unimodal distributions by using VysochanskiPetunin's inequality.

For traffic with strong LRD, a design and analysis of the decay function model is done in the frequency domain. Most Internet traffic has monotonically decreasing strength of variation functions over frequency. For this type of input traffic, it is proved that optimal schedulers must have a convex structure. Uniform resource allocation is an extreme case of the convexity and is proved to be optimal for Poisson traffic. With an integration of the convex-structural principle, an enhance GPS policy improves the service quality significantly. Furthermore, it is shown that the presence of LRD in the input traffic results in shift of variation strength from high frequency to lower frequency bands, leading to a degradation of the service quality.

The model is also extended to support server with different deadlines, and to derive an optimal time-variant (adaptive) resource allocation policy that minimizes server load variances and server resource demands. Simulation results show time-variant scheduling algorithm indeed outperforms time-invariant optimal decay function scheduler.

Internet traffic has two major dynamic factors, the distribution of request size and the correlation of request arrival process. When applying decay function model as scheduler to random point process, corresponding two influences for server workload process is revealed as, first, sizing factor — interaction between request size distribution and scheduling functions, second, correlation factor — interaction between power spectrum of arrival process and scheduling function. For the second factor, it is known from this thesis that convex scheduling function will minimize its impact over server workload. Under the assumption of homogeneous scheduling function for all requests, it shows that uniform scheduling is optimal for the sizing factor. Further more, by analyzing the impact from queueing delay to scheduling function, it shows that queueing larger tasks *vs.* smaller ones leads to less reduction in sizing factor, but at the benefit of more decreasing in correlation factor in the server workload process. This shows the origin of optimality of shortest remain processing time (SRPT) scheduler.

AUTOBIOGRAPHICAL STATEMENT

MINGHUA XU

Minghua Xu is a graduate student of Department of Electrical and Computer Engineering at Wayne State University. He received his B.S. degree in biomedical engineering, M.S. degree in bioelectronics from Southeast University, China in 1994 and 1997 respectively. He received second M.S. degree in computer engineering from Wayne State University in 1999. He worked as technical specialist in Compuware Corp. from 1999 to 2007. He is working as software engineer and domain technical lead in PayPal Inc. since 2007.

His research interests include optimal scheduling, performance evaluation and resource management in Internet server. He has published one journal paper on IEEE Transaction on Parallel and Distributed Systems and four paper in proceedings of referred conferences. He also published one journal paper in neural network in Chinese during thesis work for M.S. degree.

He taught graduate digital signal processing course in one semester, served as teaching assistant for computer language&automata, and taught in the lab for object oriented C++ programming course.

His industrial background extends to design service-oriented Internet applications, rule/model execution engine, data modeling, Six-Sigma control and reporting engine.