Decay Function Model for Resource Configuration and Adaptive Allocation on Internet Servers

Minghua Xu and Cheng-Zhong Xu
Department of Electrical and Computer Engineering
Wayne State University, Detroit, Michigan 48202

***Abstract*— Server-side resource configuration and allocation for QoS guarantee is a challenge in performance-critical Internet applications. To overcome the difficulties caused by the high-variability and burstiness of Internet traffic, this paper presents a decay function model of request scheduling algorithms for the resource configuration and allocation problem. Under the decay function model, request scheduling is modelled as a transfer-function based filter system that has an input process of requests and an output process of server load. Unlike conventional queueing network models that rely on mean-value analysis for input renewal or Markovian processes, this decay function model works for general time-series based or measurement based processes and hence facilitates the study of statistical correlations between the request traffic, server load, and QoS of requests.**

Based on the model, we apply filter design theories in signal processing in the optimality analysis of various scheduling algorithms. We reveal a relationship between the server capacity, scheduling policy, service deadline, and other request properties in a formalism and present the optimality condition with respect to the second moments of request properties for an important class of fixed-time scheduling policies. Numerical experimental results verify the relationship and show that optimal fixed-time scheduling can effectively reduce the server workload variance and guarantee service deadlines with high robustness on the Internet.

Keywords: Quality of service, resource configuration, request scheduling, fixed-time scheduling.

I. Introduction

Performance-critical Internet applications, such as online trading and streaming services, are heavily dependent on scalable servers for guaranteed QoS provisioning. The server scalability is mostly stress tested by the use of application-specific benchmarks or synthetic workload [22]. 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. The following practical yet challenging problems, are largely remaining unsolved, or cannot be precisely resolved in theory: (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 doing effective scheduling? This paper aims to develop an analytical model for a formal treatment of these key resource configuration and scheduling issues.

Queueing network (QN) models are widely used to address similar issues in packet scheduling in networks and job scheduling in closed computer systems [28]. 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 [22], [31] for examples. But the model applicability and accuracy were found very limited because recent Internet workload characterization studies [2], [6], [13] all pointed to self-similarity and heavy tail as inherent properties of Internet traffic. Another inherent characteristic of Internet traffic is high variability. Impact of bursty traffic patterns is also beyond the capability of means value analysis of the traditional QN models.

There are researches that attempt to extend the QN models to overcome the above limitations. Three recent enhanced models are effective bandwidth based queueing analysis [4], [12], [14], [15], hierarchical renewal processes like Markov Modulated Poisson Process (MMPP) [17], [3], and Fractional Gaussian Noise (FGN) [7], [27], [9], [23]. The effective bandwidth approach is based on an assumption that the Internet traffic process has the normalized logorithmic moment generating function. It implies the traffic can be bounded by an "envelope" process in a much simpler mathematical representation. The envelope process is then studied under resource reservation based scheduling policies. We note that the effective bandwidth model is good at providing bounds

on the QoS domain. But the precision of the bounds heavily depends on the burstiness of underlying traffic. The MMPP based queueing models can include correlation statistics and other second order moments into Markov chain transitive matrices. However, the solvability of these advanced models remains open. FGN was introduced to study the self-similarity impact on the queueing system. Most of queueing analysis of the FGN traffic model were based on the assumption of constant service rate and the model was limited to Gaussian based martingale distributions. There were recent researches on the impact of long range dependent arrival process on waiting time in FGN/GI/1 and MG∞/GI/1 typed hierarchical queues [33]. Their models assumed fixed queueing disciplines and provided no control parameters for QoS provisioning.

On another track altogether, Abdelzaher, *et al.* recently developed feedback control approaches to deal with the impact of high order moments of Internet traffic [1]. They treated requests as aperiodic real-time tasks with arbitrary arrival times, computation times, and relative deadlines. It is known that such a group of tasks scheduled by a deadline-monotonic policy will always meet their deadline constraints as long as the server utilization is less than 5/8. The authors applied linear feedback control theories to admit an appropriate number of requests so as to maintain system utilization at the upper bound. This approach assumes a fixed-priority scheduling policy and considers no information about input request processes.

Feedback control operates by responding to measured deviations from the desired performance. It is oblivious to the change of workload input. When the residual errors are too big, the linear controller would lead to a poor controllability for a non-linear system. By contrast, queueing theories provide predictive frameworks for inferring expected delays according to the input load change. For a robust control in request admission, Sha and Lu, et al [30], [18], [19] most recently proposed to integrate a queueing model with feedback control for absolute and relative request delay guarantees in Web servers [18], [19], [30]. They applied PI (Proportional-Integral) control to adjust the number of processes allocated to each client (or client class) in persistent connected servers, based on the initial QN estimates.

We note that the control theoretical approaches aim to *reduce completion time by controlling the server load mean through admission control.* The server load mean can be estimated according to the first moment statistics of input request processes. The feedback control approaches deal with the impact of high order moments of Internet traffic, in particular high variability and auto-correlations in an algorithmic manner. Although the approaches ensure the robustness of the resource allocation mechanism, the impact of request scheduling on server performance under various input request processes is left undefined.

In this paper, we take a different approach, by constructing a decay function model for time varying request scheduling. The server system under consideration assumes a general request arrival process combined with a general request size distribution. Each request is scheduled by a decay function which determines how much resource is allocated dynamically during the request processing time. The server load changes with the resource allocation. The adapation of the decay function model reflects in the relationship between request completion time, request size and autocorrelation, and server load. Its goal is *to minimize the server load variance due to high order moments of Internet traffic by controlling completion time of individual requests (through resource allocation).* The decay function model is named for the fact that resource allocations of each request will eventually vanish to zero when the request exists the server. This “decay” concept differs fundamentally from the “decay” usage scheduling in literature [8]

In the decay function model, input request process and output server workload processes could be any general time-series based or measurement based processes. There is no need to reconstruct renewal queueing processes from the measurement-based models, as required by QN models and control theoretical approaches. We model the time varied scheduling function as a filter and applied filter design theories in signal processing in the optimality analysis of various scheduling algorithms. The decay function can be adapted to meet changing resource availability and the adaptation does not affect the solvability of the model.

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 [11]. 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 value in the queue is selected for execution at each scheduling epoch. Our decay function model facilitates the development of fundamentals of the time function abstraction on Internet servers.

We also note that there are established frameworks for modelling of highly variable traffic on network routers [24], [20], [5]. 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 modelling for two main reasons. First, routers take packets from the same connection as input, which can be buffered for smoothing the downstreaming 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 $10^{-5} \sim 10^{-7}$ [20]. 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.

In summary, major contributions of our work include a decay function model that characterizes resource allocation and request scheduling as a transfer-function based filter system in between input request traffic and output server load. By using filter design theories in signal processing, we analyzed the optimality of schedulers with the objective of minimizing the server load variance. We revealed the impact of high order moments of the request processes on the optimality of scheduling policies in a formalism and presented the optimality condition with respect to the second moments of the request properties for an important class of fixed-time scheduling.

The rest of the paper is organized as follows. Section 2 defines the decay function model for scheduling and Section 3 addresses the resource configuration and allocation problem for fixed-time scheduling algorithms. Section 4 deliberates the model with some numerical simulations. Future work is discussed in Section 5.

II. The Decay Function Model

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 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 [26].

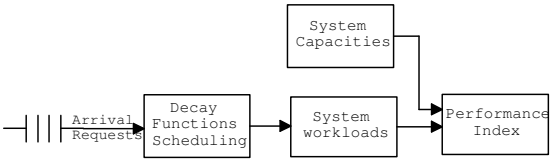

Fig. 1. Illustration of the decay function model.

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 $\mathbf{c}$. 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 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:

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

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 (1) can be re-written as:

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

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

$$d\big(t, t_s, w, l(t)\big) = h(t, t_s)g(w)f\big(l(t)\big), \tag{3}$$

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 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.

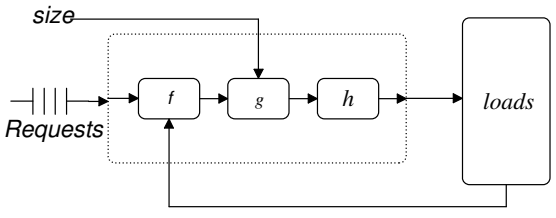

Fig. 2. Decomposition of scheduling

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,

$$\begin{aligned} w &= \sum_{t=t_s}^{t_s+t_d} d\big(t, t_s, w, l(t)\big) \\ &= \sum_{t=t_s}^{t_s+t_d} h(t, t_s)g(w)f\big(l(t)\big) \end{aligned} \tag{4}$$

We refer to (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,

$$\begin{aligned} l(t) &= \sum_{t_s=0}^{t} \sum_{k=1}^{n(t_s)} d\big(t, t_s, w_k, l(t)\big) \\ &= \sum_{t_s=0}^{t} \sum_{k=1}^{n(t_s)} h(t, t_s)g(w_k)f\big(l(t)\big) \end{aligned} \tag{5}$$

We refer to (5) as a *workload function*. The workload and scheduling functions together determine the dynamics of resource scheduling on the Internet server.

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 workload $l(t)$.
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 during 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 (5) in the case of fixed-time scheduling can be simplified as

$$\begin{aligned} l(t) &= \sum_{t_s=t-t_d}^{t} \sum_{k=1}^{n(t_s)} h(t-t_s) \cdot g(w_k) \\ &= \sum_{t_s=t-t_d}^{t} h(t-t_s) \cdot \sum_{k=1}^{n(t_s)} g(w_k) \\ &= \sum_{t_s=t-t_d}^{t} h(t-t_s) \cdot \tilde{w}(t_s), \end{aligned} \tag{6}$$

where

$$\tilde{w}(t_s) = \sum_{k=1}^{n(t_s)} g(w_k). \tag{7}$$

Given the arrival process in (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* [10].

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 (6) can then be rewritten as a convolution:

$$l(t) = h(t) * \tilde{w}(t). \tag{8}$$

Equation (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 (4) can be simplified as:

$$\sum_{t=t_s}^{t_s+t_d} h(t-t_s)g(w) = w. \tag{9}$$

That is,

$$\sum_{t=0}^{t_d} h(t) = \frac{w}{g(w)}. \tag{10}$$

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

$$\sum_{t=0}^{t_d} h(t) = 1. \tag{11}$$

According to theorems of random signal processing [21], 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 crosscorrelation between input and output of (8) can be established. The assumption of WSS is quite general in the modelling of random signal. This assumption is also valid for Internet traffic. In [32], the authors proved that the Internet arrival process could be at least modelled as a phase-pieced WSS processes.

III. Resource Configuration and Allocation

A. Resource Configuration

In this section, we apply the analytical results derived from preceding section 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 pre-defined quality requirement.

With *a priori* known mean and variance of the server load $l(t)$, we can estimate the probability distribution tail by Chebyshev's inequality [10].

Lemma 3.1: The upper bound of the probability of workload l exceeding capacity $\mathbf{c}$, $prob(l > \mathbf{c})$, is

$$\frac{\sigma_l^2}{\sigma_l^2 + (\mathbf{c} - \mu_l)^2}. \tag{12}$$

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,

$$prob(l > \mathbf{c}) \leq \frac{1}{(\mathbf{c}+x)^2} E((l(t) + x)^2). \tag{13}$$

Since

$$E((l(t) + x)^2) = \sigma_l^2 + \mu_l^2 + 2x\mu_l + x^2, \tag{14}$$

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. ■

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

Theorem 3.1: Let v be a pre-defined bound of the probability that workload exceeds the server capacity. The estimate of capacity is

$$\mathbf{c} = \sqrt{\frac{1-v}{v}} \cdot \sigma_l + \mu_l. \tag{15}$$

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 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 3. 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.

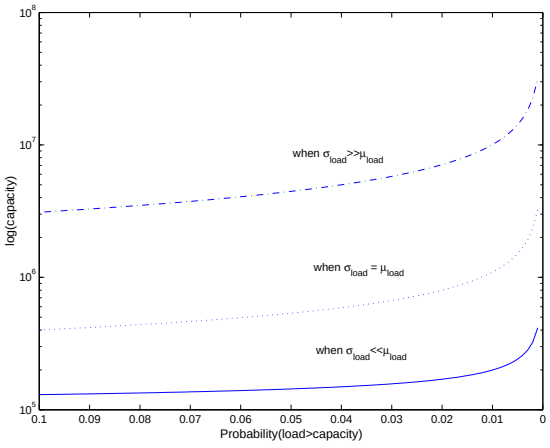

Fig. 3. The capacity and chance of overload

B. 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 3.2: If the compounded input process $\tilde{w}(t)$ is wide sense stationary, the optimal fixed-time scheduling is to find a $\mathbf{h}$ that

$$\text{Minimize} \quad \mathbf{h}'\Omega\mathbf{h} \tag{16}$$

$$\text{Subject to} \quad \sum_{i=0}^{t_d} \mathbf{h}_i = 1, \text{ and } \mathbf{h}_i \geq 0. \tag{17}$$

Moreover, the optimization problem (16) has a unique solution.

Proof: Write (6) in vector format as

$$l(t) = \mathbf{h}'(0)\mathbf{w}(t). \tag{18}$$

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

$$E[l(t)] = E[\mathbf{h}'(0)\mathbf{w}(t)] = E[w].$$

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

$$\begin{aligned} Var(l(t)) &= E[l^2(t)] - (E[l(t)])^2 \\ &= E[\mathbf{h}'(0)\mathbf{w}(t)\mathbf{h}'(0)\mathbf{w}(t)] - E[w] \\ &= E[\mathbf{h}'(0)\mathbf{w}(t)\mathbf{w}'(t)\mathbf{h}(0)] - E[w] \quad (19) \\ &= \mathbf{h}'(0)\Omega\mathbf{h}(0) - E[w]. \quad (20) \end{aligned}$$

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 [10]. 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 $y = \mathbf{U}'\Lambda\mathbf{U}$. This is a standard form. It is minimized when $\lambda_i g_i^2 = \lambda_j g_j^2$ for any $1 \le i, j \le t_d$. 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 (16). ■

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 3.2 and its proof is omitted. We demonstrate the calculus of the results by examining input traffic with different autocorrelation functions.

Corollary 3.1: 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.

Corollary 3.2: 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

$$t_d = \frac{(1-v)\sigma_{\tilde{w}}^2}{v(\mathbf{c} - \mu_{\tilde{w}})^2}. \tag{21}$$

Proof: According to Corollary 3.1, the function $h(t)$ should be set to a constant value $1/t_d$ for period $0 < t \le t_d$ 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 3.1, we know that

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

Combining the two equations completes the proof. ■

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

$$\frac{t_{d_2}}{t_{d_1}} = \frac{\sigma_2^2 \cdot (\mathbf{c} - \mu_1)^2}{\sigma_1^2 \cdot (\mathbf{c} - \mu_2)^2}. \tag{22}$$

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

C. Examples

Example 3.1: 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 [16]. 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 [25]. 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 [29]. 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 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 3.2) are shown on Figure 5. A uniform decay function for independent traffic (Corollary 3.1) is included in Figure 5 for reference.

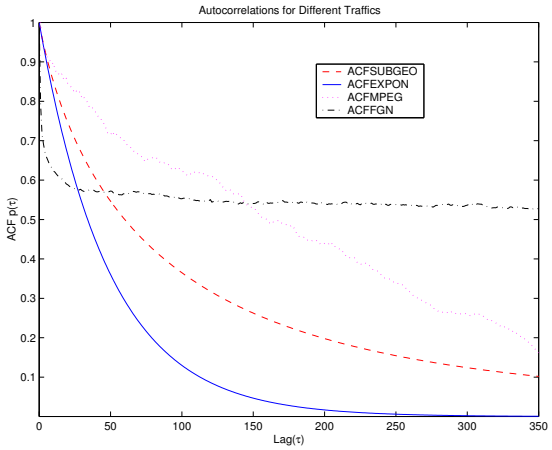

Fig. 4. Four distinct patterns of auto-Correlation function.

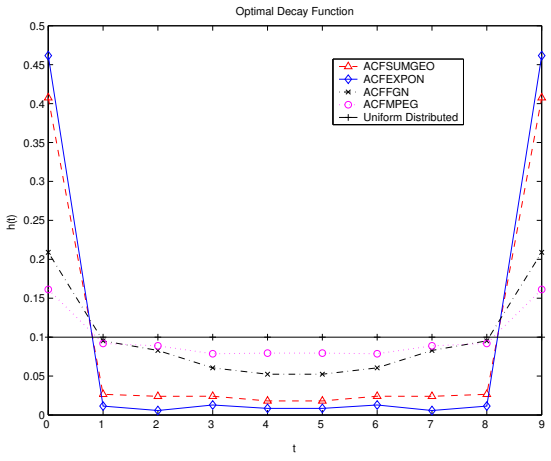

Fig. 5. Comparison of optimal decay functions for traffic with different ACFs.

Example 3.2: 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 3.1 , the workload variance

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

Thus the server capacity can be calculated according to Theorem 3.1 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 3.2, the target deadline can be calculated as

$$\begin{aligned} t_d &= \frac{(1-v)\sigma^2_{\tilde{w}}}{v(\mathbf{c}-\mu_l)^2} \\ &\approx \frac{0.99 \cdot (1.4884e+10)}{0.01 \cdot (400,000-41864)^2} \\ &\approx 12. \end{aligned} \quad (23)$$

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 22 is very precise.

IV. Numerical Experiments

To verify the above analytical results, we conducted numerical simulations of the Internet server based on the decay function model. The simulation was conducted in three aspects: (1) Verify the relationship between server capacity, scheduling and QoS for an i.i.d. input request process; (2) Compare the decay function scheduling with another popular GPS scheduling [24]; (3) Analyze the sensitivity of the decay function to request traffic.

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 time in the backlogged 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

TABLE I

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 $\mathbf{c}$

TABLE II

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%

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 [21]. In this section, we use notations as listed in Table I.

A. Capacity Configurations and Variances

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 3.2. The compounded input pattern is shown in Figure 6.

Table II gives the statistics of the simulated server according to the Chevyshev estimation in Lemma 3.1. 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 target 99% server capacity satisfactory rate.

As we mentioned in the preceding section, the Chevyshev estimation can be tightened by taking into consideration of the empirical distribution tail. Figure 7 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

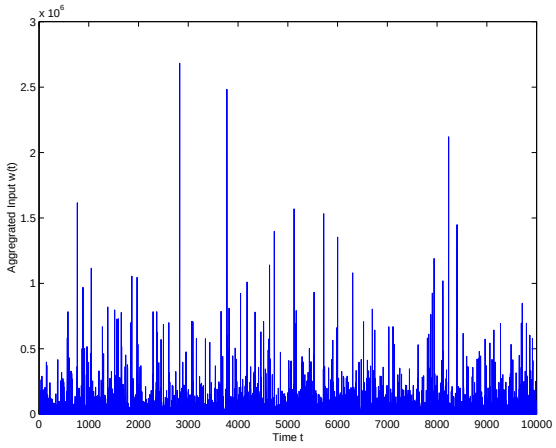

Fig. 6. The compounded input $\tilde{w}(t)$

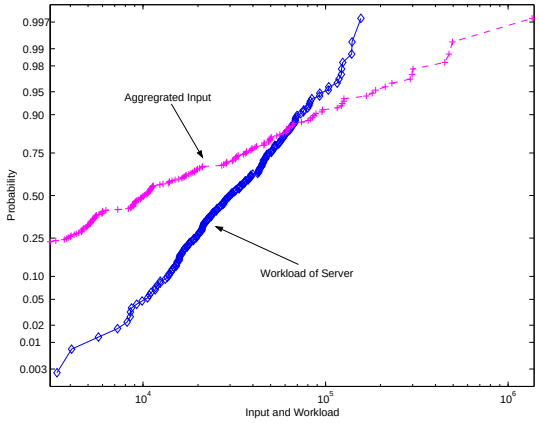

Fig. 7. Normal plots of l and $\tilde{w}$

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 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 III. 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.

B. Comparison of Fixed-time decay function scheduling and GPS scheduling

Using the same set of parameters as of last experiment, we compared the optimal fixed-time scheduling with the GPS

TABLE III

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 (x10^9)	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%

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 III presents their comparative results. 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 8 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.

C. 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 $n(t)$

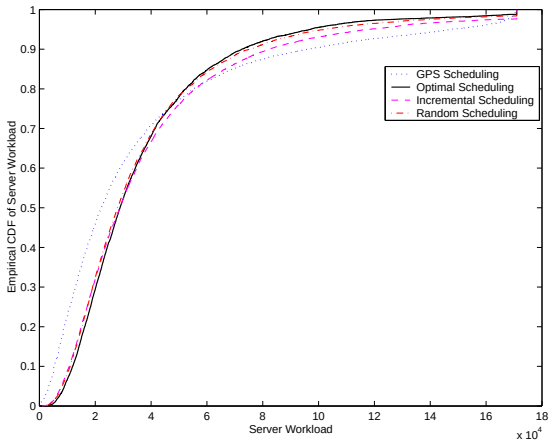

Fig. 8. The empirical CDF of server workload with different scheduling algorithms.

TABLE IV

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 (x10^9)	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%

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 IV 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 is significantly lighter than GPS scheduling in Figure 9. With a light-tail and small variance, the optimal fixed-time scheduling tends to provide better deadline guarantees in terms of both maximum and

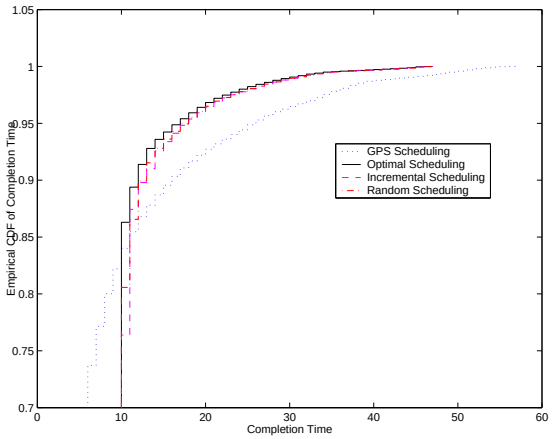

Fig. 9. The CDF plot of completion time when $n(t) \sim ln(1, 1.7)$

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.

D. Performance Prediction

In the last 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 3.2, 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 V. 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 V

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 3.2, 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 VI 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 3.2.

TABLE VI

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%

V. CONCLUDING REMARKS

In this paper, we introduced a decay function model to study the QoS provisioning problems on Internet servers. The model abstracts QoS-aware scheduling into a transfer-function based filter system that has general time-series based or measurement request process as input and server load process as output. By using filter deign theories in signal processing, it is able to reveal the relationships between Internet workloads, resources configuration and scheduling, and server capacity. The parameters of the model have strong physical meanings, especially the decay function that describes the detailed scheduling activities on the server, We analyzed and solved the model for an important class of QoS-aware scheduling: fixed-time scheduling. We derived the optimality

conditions with respect to the statistical properties of the input traffic. We verified the analytical results via numerical simulations in both light and heavy loaded cases.

The decay function model paves a new way for the derivation of optimal scheduling policies by investigating the impact of the second moments of input traffic on server load. We used the Chebyshev's inequality for an estimation of required server capacity for general traffic and derived a relationship between request process and server load in a formalism for non-adaptive scheduling. We will further our study to extend the decay function model to include the feedback of server load in the analysis of adapative scheduling. The current model assumes a proportional relationship between the request size and resource requirement. This may not be the case in real Internet servers due to the presence of caching and locality. The scheduling model needs to be deliberated to take into account their effects as well.

REFERENCES

[1] 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.

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

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

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

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

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

[7] 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.

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

[9] 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.

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

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

[12] 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.

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

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

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

[24] 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.

[25] V. Paxson. Fast approximation of self-similar network traffic. Technical report, EECS Division, University of California, Berkeley, 1995. http://town.hall.org/Archives/pub/ITA/html/contrib/fft-fgn.html.

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

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

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

[29] O. Rose. Statistical properties ofr 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.

[30] 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.

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

[32] 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.

[33] 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.