Robust Processing Rate Allocation for Proportional Slowdown Differentiation on Internet Servers

Jianbin Wei†, Xiaobo Zhou‡, and Cheng-Zhong Xu†

† Department of Electrical and Computer Engineering
Wayne State University, Detroit, MI 48202
Email: {jbwei, czxu}@wayne.edu

‡ Department of Computer Science
University of Colorado at Colorado Springs, CO 80933
Email: zbo@cs.uccs.edu

***Abstract*— A desirable behavior of an Internet server is that a request's queueing delay depends on its service time in a linear fashion. Measuring the quality of service in terms of *slowdown*, the ratio of a request's queueing delay to its service time, provides a simple way to attain the objective. Moreover, it treats client requests equally regardless of their service time, whereas response time favors requests that need more processing resources. In this paper, we propose a proportional slowdown differentiation (PSD) service model on Internet servers. It aims to maintain pre-specified slowdown ratios between different classes of client requests. To provide PSD services, we first derive a closed-form expression of the expected slowdown in an $M/G/1$ FCFS queueing system with the most typical heavy-tailed, the bounded Pareto, service time distribution. Based on the closed-form expression, we design a queueing-theoretic strategy of processing-rate allocation. The rate allocation is realized by deploying a virtual server for each class. Simulation results show that the strategy can provide controllable PSD services on Internet servers. It, however, comes along with large variance and weak predictability due to the dynamics of Internet traffic. To address these issues, we design an integral feedback controller and integrate it into the queueing-theoretic strategy. Simulation results demonstrate that the integrated strategy is robust and can deliver predictable PSD services at a fine-grained level. We implement the processing-rate allocation strategies in an Apache Web server. Experimental results further demonstrate their feasibility in practice.**

Keywords: Quality of Service, Slowdown, Queueing theory, Feedback control, Processing-rate allocation

I. Introduction

The past decade has seen an increasing demand for provisioning of different levels of quality of service (QoS) to various network applications and customers. Differentiated services (DiffServ) [7] is a major service architecture for this requirement. Many algorithms have been proposed to provide differentiated services in terms of queueing delay in network core, such as proportional average delay [17], adaptive waiting-time priority [32], and Little's average delay [47]. For example, in proportional average delay, a class's priority is set according to the average delay of its departed packets; a packet from the highest priority class is transmitted. There are also some efforts in differentiated loss service provisioning [15], [26]. For example, in [26], the authors proposed an algorithm in which a packet is dropped randomly based on its arrival process so as to provide differentiated loss services in short time-scales.

The end-to-end performance of Internet services is not only affected by network core, but also by Internet servers. Moreover, a desirable behavior of an Internet server is that a request's queueing delay depends on its service time in a linear fashion, which implies that a request twice as long as some other will spend on the average twice as long in the server [27]. Measuring quality of service in terms of *slowdown* [22], the ratio of a request's queueing delay to its service time, provides a simple way to attain the objective. It treats client requests equally regardless of their service time, whereas response time favors requests that need more processing resources [22]. For example, it is unreasonable to get the same average response time (say 10 seconds) for a 10K bytes JPG file and also a 600K bytes PDF file. It becomes worse in Web servers since most of Web objects are small in size [3].

Slowdown or its variant has been adopted as an important metric of quality of service in recently designed QoS-aware systems, including [6], [9], [14], [18], [19], [22], [23], [24], [25], [43], [49]. For example, in [25], the authors proposed a size-based scheduling algorithm that assigned high priorities to requests with small service time so as to improve the performance of Web servers in terms of mean slowdown and mean response time. In these systems, a request experienced smaller slowdown receives better quality of service than one with larger slowdown.

Few work exists for providing differentiated services in terms of slowdown. Although existing algorithms for proportional delay differentiation services in network core can be tailored for provisioning of differentiated delay services on Internet server [30], they are unable to provide PSD services. Slowdown depends on the service time of a request, which is costly, if not impossible, to predict *a priori*. In [51], the authors proposed a service differentiation model in terms of stretch factor, a variant of slowdown, for a cluster of servers. Their strategy, however, is limited to servers in an $M/M/1$ queueing system.

In this paper, we propose a proportional slowdown differentiation (PSD) service model on Internet servers. It aims to maintain pre-specified slowdown ratios between different

classes of client requests. As in [22], we model Internet servers as an $M/G/1$ queueing system with bounded Pareto service time distribution. We refer to this queueing model as $M/G_P/1$ in the rest of this paper. To provide PSD services, we first derive a closed analytic form of the expected per-class slowdown. Based on the expression, we develop a queueing-theoretic strategy for processing-rate allocation. The processing rate of a class is realized by *virtual servers*. Each virtual server processes requests from the same class in an FCFS manner. Simulation results verify that the queueing-theoretic strategy can guarantee the controllability of PSD services on average. Such services, however, come along with large variance and weak predictability due to the dynamics of Internet traffic. As a remedy, we design an integral feedback controller and integrate it into the queueing-theoretic strategy. Simulation results demonstrate that the integrated strategy is robust and can deliver predictable PSD services at a fine-grained level. We also implement the processing-rate allocation strategies in an Apache Web server. Experimental results further demonstrate their feasibility in practice.

The structure of the paper is as follows. Section II presents the PSD service model. Section III discusses the slowdown in an $M/G_P/1$ queueing system. Section IV presents the queueing-theoretic strategy of processing-rate allocation and evaluates its performance. Section V introduces an integral feedback controller, integrates it into the queueing-theoretic strategy, and demonstrates the integrated strategy's robustness in providing superior fine-grained control over slowdown ratios. Section VI presents our application-level implementations of these strategies and experimental results. Section VII gives the related work in providing differentiated services and Section VIII concludes the paper.

II. Proportional Slowdown Differentiation Service Model

The PSD service model is to maintain pre-specified slowdown ratios between different classes. In the model, incoming requests are classified into N classes. Note that the classification is based on service providers' policies, such as service fee paid by clients. It is independent of characteristics of requests. Let $S_i(k)$ denote the average slowdown of class i computed at sampling period k and δ_i its pre-specified differentiation parameter. For class i and j, it requires

$$\frac{S_i(k)}{S_j(k)} = \frac{\delta_i}{\delta_j}, \quad 1 \le i, j \le N. \tag{1}$$

Due to the constraint of the conservation law, the sum of processing rates that are allocated to all classes must equal to the capacity of the system. Let c_i denote the processing rate allocated to class i. The rate allocation subjects to the constraint

$$\sum_{i=1}^{N} c_i = 1. \tag{2}$$

The service model requires the provided PSD services to be predictable, controllable, and fair.

- *Predictable*. It requires the PSD services to be consistent and independent of variations of class workloads.
- *Controllable*. It means the quality differences between classes should be adjustable via their differentiation parameters.
- *Fair*. It means low priority classes should not be over-compromised, especially when workloads are high.

Furthermore, to ensure the feasibility of PSD services, the system should not be overloaded. Otherwise, a class's slowdown can become infinite according to queueing theory.

III. Slowdown Modeling in an $M/G_P/1$ Queueing System

As suggested in [22], we assume that an Internet server can be modeled as an $M/G/1$ queueing system. In practice, the client sessions follow an exponential distribution [33]; many measured computer workloads are heavy-tailed, such as sizes of documents on a Web server [3], [4], and sizes of FTP transfers on the Internet [42]. In general, a heavy-tailed distribution is one for which

$$Pr\{X \le x\} \sim x^{-\alpha}, \quad 0 < \alpha < 2,$$

where X denotes the service time density distribution.

The most typical heavy-tailed distribution is Pareto distribution. Its probability density function is

$$f(x) = \alpha p^{\alpha} x^{-\alpha-1}, \quad \alpha, p > 0, x \ge p, \tag{3}$$

where α is the shape parameter. With respect to the fact that there is some upper bound on a client request's required processing resource in practice, a bounded Pareto distribution has been adopted to characterize the workloads on Internet servers [4], [22]. Let x be a request's service time. The probability density function of the bounded Pareto distribution is defined as

$$f(x) = \frac{\alpha p^{\alpha}}{1-(p/q)^{\alpha}} x^{-\alpha-1}, \quad \alpha, p, q > 0,$$

where $p \le x \le q$. We refer to an $M/G/1$ queueing system where the service time follow a bounded Pareto distribution as an $M/G_P/1$ system.

A. Preliminaries of Slowdown

Since α, p, and q are parameters of the bounded Pareto distribution, for simplicity in notation, we define a function

$$\mathcal{K}(\alpha, p, q) = \frac{\alpha p^{\alpha}}{1-(p/q)^{\alpha}}.$$

The probability density function $f(x)$ is rewritten as

$$f(x) = \mathcal{K}(\alpha, p, q) x^{-\alpha-1}. \tag{4}$$

Let m_1, m_{-1}, m_2 be its first moment (mean), the moment of its reverse, and the second moment, respectively. Let w denote a request's queueing delay in an $M/G_P/1$ FCFS queueing

system and λ the arrival rate of the requests. From (4), we have:

$$m_1 = E[X] = \begin{cases} \frac{\mathcal{K}(\alpha,p,q)}{\mathcal{K}(\alpha-1,p,q)} & \text{if } \alpha \neq 1, \\ (\ln q - \ln p)\mathcal{K}(\alpha,p,q) & \text{if } \alpha = 1; \end{cases} \quad (5)$$

$$m_{-1} = E[X^{-1}] = \frac{\mathcal{K}(\alpha,p,q)}{\mathcal{K}(\alpha+1,p,q)}; \quad (6)$$

$$m_2 = E[X^2] = \frac{\mathcal{K}(\alpha,p,q)}{\mathcal{K}(\alpha-2,p,q)}. \quad (7)$$

According to Pollaczek-Khinchin formula, we obtain the expected slowdown shown in the following lemma.

Lemma 1: Given an $M/G_P/1$ FCFS queue where the arrival process has rate λ on a server. Let w be a request's queueing delay. Then its expected queueing delay and slowdown are

$$E[w] = \frac{\lambda m_2}{2(1-\lambda m)}, \quad (8)$$

$$S = E[\frac{w}{x}] = E[w]E[x^{-1}] = \frac{\lambda m_2 m_{-1}}{2(1-\lambda m_1)}. \quad (9)$$

Note that the slowdown formula follows from the fact that w and x are independent random variables in an FCFS queue.

B. Slowdown on Internet Servers

We assume that the processing rate of an Internet server can be proportionally allocated to a number of *virtual servers* using the proportional-share resource scheduling mechanisms, such as PGPS [40] and lottery scheduling [45]. Each virtual server processes requests from a class in an FCFS manner. It is an abstract concept in the sense that it can be a group of child processes or threads in a multi-process or multi-thread server [10].

Lemma 2: Given an $M/G_P/1$ FCFS queue on a virtual server i with processing capacity c_i, let m_1^i, m_{-1}^i, and m_2^i be the first moment (mean), the moment of its reverse, and the second moment of the class i's service time, respectively. Then,

$$m_1^i = \frac{m_1}{c_i}, \quad (10)$$

$$m_{-1}^i = c_i m_{-1}, \quad (11)$$

$$m_2^i = \frac{m_2}{c_i^2}. \quad (12)$$

Proof: On a virtual server with processing capacity 1, the lower bound and upper bound of the bounded Pareto distribution are p and q, respectively. On virtual server i with processing capacity c_i, because each class receives c_i fraction of processing capacity, a job takes $1/c_i$ as long. The bounds become p/c_i and q/c_i, respectively. According to (3), we have

$$\int_{p/c_i}^{q/c_i} f(x)dx = c_i^\alpha(1-(p/q)^\alpha).$$

Thus, on the virtual server i, we define the probability density function of the bounded Pareto distribution as

$$f(x) = \frac{\alpha p^\alpha}{c_i^\alpha(1-(p/q)^\alpha)} x^{-\alpha-1},$$

where $\alpha > 0$ and $p/c_i \leq x \leq q/c_i$. With the notation $\mathcal{K}(\alpha,p,q)$, the probability density function is rewritten as

$$f(x) = \mathcal{K}(\alpha,p,q)c_i^{-\alpha}x^{-\alpha-1}.$$

It follows that

$$m_1^i = \begin{cases} \frac{1}{c_i}\frac{\mathcal{K}(\alpha,p,q)}{\mathcal{K}(\alpha-1,p,q)} & \text{if } \alpha \neq 1, \\ \frac{1}{c_i}(\ln q - \ln p)\mathcal{K}(\alpha,p,q) & \text{if } \alpha = 1. \end{cases} \quad (13)$$

$$m_{-1}^i = c_i \frac{\mathcal{K}(\alpha,p,q)}{\mathcal{K}(\alpha+1,p,q)}. \quad (14)$$

$$m_2^i = \frac{1}{c_i^2}\frac{\mathcal{K}(\alpha,p,q)}{\mathcal{K}(\alpha-2,p,q)}. \quad (15)$$

By substitution with m_1, m_{-1}, and m_2 in (5), (6), and (7), we obtain the result of this lemma. This concludes the proof. ■

According to Lemma 1 and Lemma 2, we have the following theorem.

Theorem 1: Given an $M/G_P/1$ FCFS queue on virtual server i with processing capacity c_i, and arrival rate λ_i, its expected slowdown is

$$S_i = E[s_i] = \frac{\lambda_i m_2 m_{-1}}{2(c_i - \lambda_i m_1)}. \quad (16)$$

Note that the $M/G_P/1$ queue reduces to an $M/D/1$ queue when the service time of all requests are the same. In an $M/D/1$ FCFS system, (16) becomes

$$S_i = \frac{\lambda_i d}{2(c_i - \lambda_i d)}, \quad (17)$$

where d is the constant service time. This fixed-time queueing model is valid in session-based e-Commerce applications. A session is a sequence of requests of different types made by a single customer during a single visit to a site. Requests at some states such as home entry or register take approximately the same service time.

IV. Queueing-theoretic Rate-Allocation Strategy

In this section, we present a queueing-theoretic strategy for PSD service provisioning. We evaluate the effectiveness of the strategy with respect to the fairness, controllability, and predictability requirements via comprehensive simulations.

A. Rate Allocation based on Queueing Theory

According to Theorem 1, we solve the processing-rate allocation problem as defined by the PSD requirements in (1), the constraint in (2), and class i's expected slowdown (16). The processing rate that needs to be allocated to class i thus is

$$c_i = \lambda_i m_1 + \frac{\lambda_i/\delta_i}{\sum_{i=1}^N \lambda_i/\delta_i}(1 - \sum_{i=1}^N \lambda_i m_1). \quad (18)$$

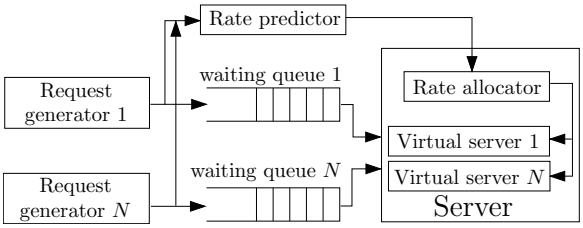


Fig. 1. The simulation model's structure.

The first term of (18) is a baseline that prevents the class from being overloaded. By (16), if the class is overloaded, then its average slowdown becomes infinite according to queueing theory. The second term is a portion of surplus processing rate determined by the class's differentiation parameter and the load condition (i.e., its *normalized arrival rate*). It controls the quality differences between classes.

By substituting (18) into (16), we obtain the expected slowdown of class i as

$$S_i = \frac{\delta_i m_2 m_{-1} \sum_{i=1}^{N} \lambda_i/\delta_i}{2(1 - m_1 \sum_{i=1}^{N} \lambda_i)}. \quad (19)$$

Equation (19) implies the following three basic properties regarding the predictability and controllability of PSD services due to the allocation strategy.

- Slowdown of a class increases with its request arrival rate.
- With the increase of the differentiation parameter of a class, its slowdown increases while slowdown of all other classes decreases.
- With the increase of a class's load (request arrival rate), a higher priority class causes a larger increase in its slowdown than that of a lower priority class.

B. Simulation Model

We built a simulation model of an $M/G_P/1$ queueing system. The model consisted of a number of request generators, waiting queues, a rate predictor, a processing-rate allocator, and a number of virtual servers. Figure 1 outlines the basic structure of the simulation model.

The request generators produced requests with exponential inter-arrival distribution and bounded Pareto size distribution by modifying GNU scientific library [21]. We assume that a request's service time is proportional to its size. Note that the number of classes in the PSD service model is usually rather limited. It varied from 2 to 3 in many similar experiments for proportional delay differentiation service provisioning [17], [32]. In addition, as recommended in [38], a service provider should support two or three different levels of services: premium, assured, or best-effort services. In the simulations, each request was sent to the server and stored in its corresponding waiting queue. Requests from the same class were processed by a virtual server in an FCFS manner.

The rate predictor calculates the processing rate of a class using predicted load of the class according to (18). As pointed out in [44], to provide differentiated services, we need to manipulate the probability distribution of a random process, where it is the number of events (request arrivals) that affects mean slowdown. Meanwhile, the number of events should be reasonable large so that the scheduler can determine the effect of resource allocation on the achieved slowdown ratio. Therefore, in the simulation, the load was predicted for every sampling period, which was measured as a thousand departed requests. In these simulations, we applied moving average method to predict a class's load. A class's predicted load is the average of its past five sampling periods. The rate allocator enforces the rate allocation set by the rate predictor for every sampling period.

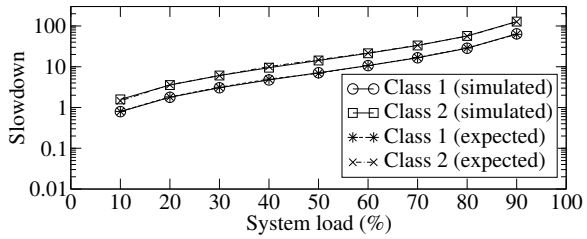
Simulation parameters were set as follows. The shape parameter (α) of the bounded Pareto distribution was set to 1.5, as suggested in [17]. As indicated in [22], its lower bound and upper bound were set to 0.1 and 100, respectively. We also carried out experiments with larger upper bound settings to evaluate its impact. All classes were assumed to have similar load in long time-scales. As we shall see in Figure 7, the load of a class is dynamic in short time-scales. In every experiment, the simulator was first warmed up for 10 sampling periods, the slowdown of a class was then measured for every following sampling period. After 60 sampling periods, the measured slowdown was averaged. Each reported result is an average of 100 runs.

We have evaluated the performance of the queueing-theoretic strategy and the integrated strategy under different workload settings. We varies the number of class, the arrival rate ratio of the classes, the differentiation parameters. We have also examined different methods, such as exponential moving average, for load prediction. While we do not present all the results due to space limitation, it worths noting that we did not find any qualitative difference between them.

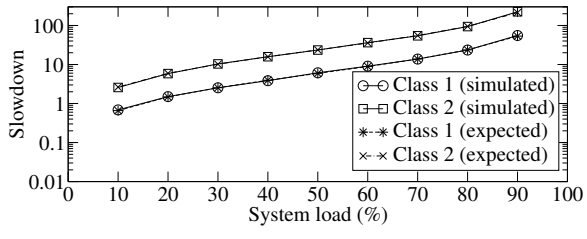
C. Effectiveness of the Queueing-theoretic Strategy

In this section, we show the effectiveness of the queueing-theoretic strategy by comparing the simulated slowdown with the expected values calculated by (19) under various load conditions. We first carried out simulations with two classes. The results are depicted in Figure 2(a). To show the results in a comparable way, the logarithmic y-axis is used. From this figure, it is clear that there are very small differences between the simulated and expected slowdown under various load conditions. This indicates that the queueing-theoretic strategy is able to guarantee the target slowdown ratio between these two classes. We also evaluated the strategy's effectiveness under different target slowdown ratios and number of classes. Their results are shown in Figure 2(b) and Figure 2(c). From both figures we can observe that the target slowdown ratios are always achieved. They demonstrate that the rate-allocation strategy is able to achieve the expected slowdown effectively. This provides the base for the following discussions on the properties of the queueing-theoretic strategy.

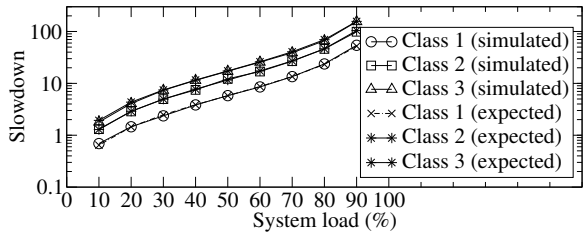
Notice that the slowdown of a class increases with the increase of system load. This can be explained by (19). With the increase of request arrival rates (system load), $1 - m_1 \sum_{i=1}^{N} \lambda_i$ decreases and $\sum_{i=1}^{N} \lambda_i/\delta_i$ increases. Therefore, S_i increases.



(a) Two classes ($\delta_2/\delta_1 = 2$).

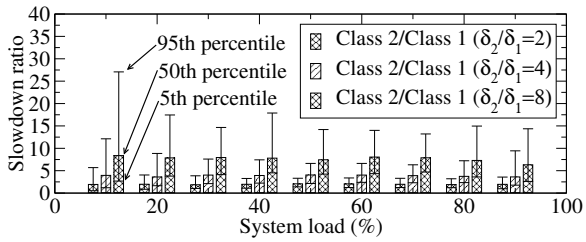


(b) Two classes ($\delta_2/\delta_1 = 4$).

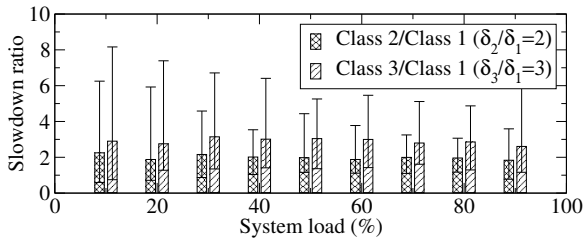


(c) Three classes ($\delta_3/\delta_2/\delta_1 = 3/2/1$).

Fig. 2. Simulated and expected slowdown of different number of classes and differentiation parameters.



(a) Two classes.



(b) Three classes.

Fig. 3. Achieved slowdown ratios due to the queueing-theoretic strategy.

D. Differentiation Predictability of the Strategy

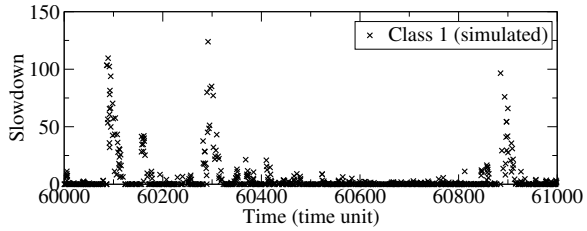
Recall that the predictability of PSD services means the slowdown of a higher priority class is proportionally smaller than that of a lower priority class under different system load conditions and time-scales. We carried out experiments to examine the capability of the queueing-theoretic strategy in supporting this property.

We first conducted experiments with different number of classes (i.e., two classes and three classes). Figure 3(a) shows the simulation results of two classes with different target slowdown ratios. Figure 3(b) shows the simulation results with three classes. In both figures, it can be observed that the strategy guarantees the achieved slowdown ratios to be around the target ones, which means a higher priority class has proportionally smaller average slowdown than a lower priority class in long time-scales.

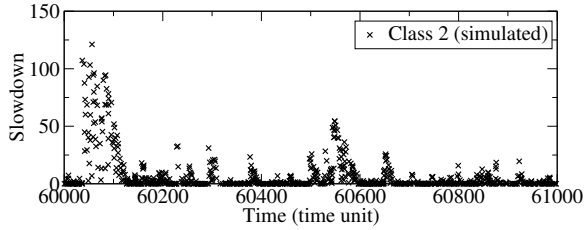
We can observe from this figure that the variance of the achieved slowdown ratios is large. For example, when the system load is 50% and the target slowdown ratio is 4, the difference between the 95th and the 5th percentile is 5. There are two reasons for this. First, although the strategy is able to control the average slowdown of a class by adjusting the processing rate based on queueing theory, it provides no way to control the variance of the slowdown simultaneously. Second, the workload of a class is stochastic and may change abruptly [42], [48]. In the queueing-theoretic strategy, the processing rate of a class is calculated according to a class's predicted load, which is difficult to predict accurately from history information. Such inaccuracy in load prediction results in large variance along with the achieved slowdown ratio.

Figure 3(a) also shows the provided PSD services violate the predictability requirement when the target slowdown ratio is small. For example, when δ_2/δ_1 is 2 and the system load is 10%, the 5th percentile of the achieved slowdown ratios is smaller than 1. It means that the higher priority classes received worse services than the lower priority class sometimes. In Section V we shall show how to provide predictable PSD services at a fine-grained level.

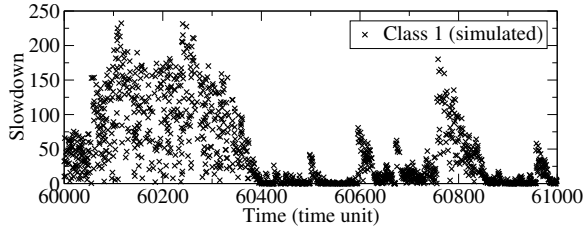
In order to demonstrate the differentiation predictability due to the queueing-theoretic strategy in short time-scales, we present the slowdown of individual requests in Figures 4. It can be observed from Figure 4(a) and Figure 4(b) that, although the target slowdown ratio of class 2 to class 1 is 2, sometimes requests from class 1 have larger slowdown than class 2 and vice versa. Figure 4(c) and Figure 4(d) show the results in heavy load conditions. It can be observed that requests from class 1 experienced larger slowdown than those from class 2. This behavior contradicts their target slowdown ratios. Close analysis shows that the slowdown ratio of class 2 to class 1 is 0.33 instead of 2 during this period. More experiments have been carried out to verify this behavior. It is found that sometimes the behavior of individual requests is consistent with their slowdown parameters, and sometimes not. The results shown in these figures suggest that the strategy is unable to provide predictable PSD services in short time-scales. This is because the queueing-theoretic strategy determines the processing rate allocated to a class periodically according to its macro-behavior (predicted class load) rather than the micro-behavior (individual requests' slowdown).



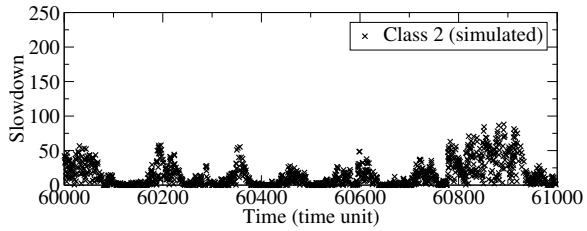
(a) System load is 50%.



(b) System load is 50%.

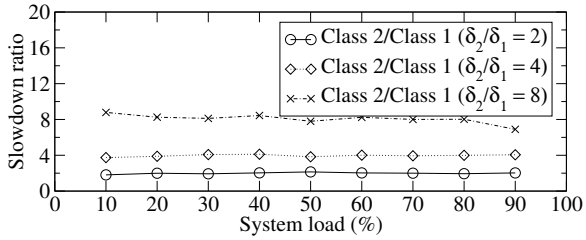


(c) System load is 90%.

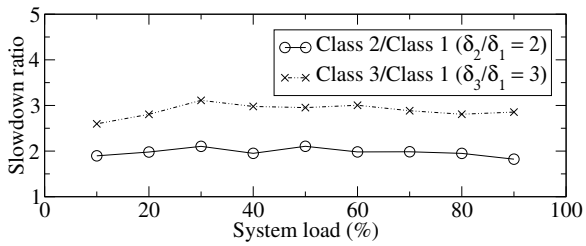


(d) System load is 90%.

Fig. 4. Slowdown of individual requests with different system load conditions.



(a) Two classes.



(b) Three classes.

Fig. 5. Simulated slowdown ratios with different number of classes.

E. Differentiation Controllability of the Strategy

In this section, we demonstrate the differentiation controllability due to the queueing-theoretic strategy. Figure 5(a) plots the achieved slowdown ratios of two classes with different differentiation parameters. It can be seen that the target slowdown ratios can be accurately achieved when they are small (e.g., δ_2/δ_1=2 or 4). As the increase of the target slowdown ratio, the error becomes large due to load-prediction errors. From (18), it is clear that the processing rate allocated to a class is determined by its class load, its differentiation parameter, and the system load. According to (18), the prediction errors have larger impact on the achieved slowdown ratio with the increase of the differentiation parameter.

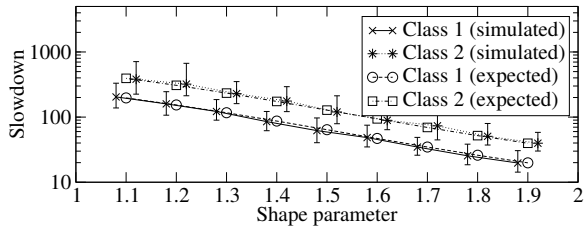
Figure 5(b) depicts the simulated slowdown ratios for a system with three classes. In comparison with Figure 5(a), it can be seen that the variance of these ratios is larger than that of the ratios with two classes. This behavior is also caused by the prediction error. When the load of one class is predicted inaccurately, it affects not only the processing rate allocated to this class, but other classes as well. Hence, the situation may become worse as the number of classes to be differentiated increases.

It is obvious that the strategy can achieve the target slowdown ratios on average in both figures. They demonstrate the capability of the strategy to control the slowdown ratios between classes.

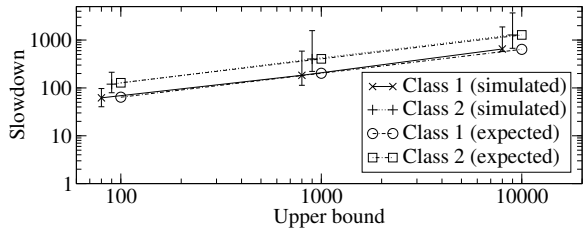
F. Impact of the Shape Parameter and the Upper Bound

In this part, we examine the impact of the shape parameter α and the upper bound q of the bounded Pareto distribution on the performance of the strategy. The shape parameter determines the correlation of traffic requests. A large shape parameter implies that the requests are independent with each other in size. The upper bound reflects the heavy-tail property of the bounded Pareto distribution.

Figure 6(a) shows the slowdown of two classes due to different shape parameters ($1.0 \leq \alpha \leq 2.0$). The first observation is that the shape parameter has little impact on the differentiation predictability due to the proposed rate-allocation strategy. Both classes experienced the differentiated slowdown as expected. The differences between the target slowdown and the achieved ones are independent of the shape parameter. This behavior can be explained by the fact that there is no assumption about the shape parameter in the rate-allocation strategy. The second important observation is that, the slowdown of a class decreases as the shape parameter increases. Intuitively, with the given lower and upper bounds, the smaller the shape parameter α, more bursty the traffic [31]. A request may experience larger queueing delay than that from a "smooth" traffic. Formally, by (7), (6), when the shape parameter decreases, its second moment m_2 increases, its m_{-1} decreases, and then its expected slowdown increases.



(a) Impact of the shape parameter.



(b) Impact of the upper bound.

Fig. 6. Impact of the parameters of the bounded Pareto distribution.

The upper bound of the bounded Pareto distribution (p) also affects the experienced slowdown of the $M/G_P/1$ traffic. Figure 6(b) presents the simulation results with different upper bounds. It can be seen that the upper bound also has little impact on the differentiation predictability due to the rate-allocation strategy; that is, the differences between the experienced slowdown and expected ones are independent of the upper bound. Second, the higher the upper bound, the larger the expected slowdown of the classes. Note that the lower bound of the bounded Pareto distribution (k) remains the same. Intuitively, as the upper bound increases, the bounded Pareto distribution becomes more heavy-tailed and the slowdown then increases. By (7), (6), as the upper bound increases, the second moment of the traffic m_2 increases and m_{-1} remains almost unchanged. We find that, in the $M/G_P/1$ traffic model, as the shape parameter increases, the slowdown decreases; as the upper bound increases, the slowdown increases.

We can also observe that the variance of achieved slowdown ratios is affected by the shape parameter and upper bound settings. Notice that the y-axis is logarithmic. The variance increases with the decreasing of the shape parameter and the increasing of the upper bound. On one hand, for given lower and upper bounds, a small shape parameter causes large variance of the generated workload [31]. On the other hand, as the upper bound increases, the bounded Pareto distribution becomes more heavy-tailed. In both cases, more requests that demand large amount of system resource are generated. Recall that requests from the same class are processed using FCFS discipline. Requests that behind a large request experience large delay and slowdown. This further demonstrates that the variance observed in Figure 5 is caused by traffic dynamics.

G. Limitations of Queueing-theoretic Strategy

Despite the queueing-theoretic strategy can provide PSD services in long time-scales, the inaccurate workload prediction caused by traffic dynamics limits its performance in short time-scales. We carried out simulations to quantify the strategy's limitations. Figure 7 show the simulated workloads, the predictions made using moving average method and exponential moving average method, and the differences between the real workloads and predictions. In the moving average method, the workload of a class was predicted as the average of sampled workloads in previous $WindowSize$ sampling periods. In the exponential moving average method, the workload of a class was predicted as

$$PredictLoad = (1 - Weight) \times PredictLoad + Weight \times SampleLoad,$$

where $0 < Weight < 1$ and $SampleLoad$ is the workload when a sample measurement is made. Different parameters of these methods were examined. Because no qualitative difference is observed, the results presented here are due to $WindowSize = 5$ and $Weight = 0.25$. The simulated workload is 0.8 on average.

It is clear that both prediction methods are able to capture the trend of workload changes. In addition, the sum of the differences is very close to zero. Therefore, the queueing-theoretic strategy is able to provide PSD services in long time-scales.

In short time-scales, however, there are large differences between the real workloads and the predictions. Such differences cause the rate allocation in queueing-theoretic strategies inaccurate. Due to the dynamics of workloads, it is very difficult to accurately predict the workload of a class from history. Furthermore, queueing-theoretic strategies provide no means to compensate the effect of the prediction errors. Therefore, large variance is observed in Figure 3(a).

The prediction errors are the main reason behind large variance and weak predictability of the PSD services provided by the queueing-theoretic strategy. Aforementioned, it is difficult to predict a class's load from history information. How to reduce the impact of such prediction errors motivates the design of our integrated processing-rate allocation strategy.

V. Integrated Rate-Allocation Strategy with Feedback Control

Feedback control provides a sound way to measure and compensate the effect of the disturbances introduced by the prediction errors. Therefore, we take advantage of feedback control theory to address the limitations of the queueing-theoretic strategy so as to provide predictable and fine-grained PSD services.

A. Structure of the Integrated Strategy

In our strategy, feedback control theory is integrated into queueing theory. Its basic structure is presented in Figure 8. There are two major components in the system: a rate predictor and a feedback controller. The rate predictor is to estimate the processing rate of a class by (18). The feedback controller is to adjust the rate allocation according to the difference between the target slowdown ratio and the achieved one using integral control.

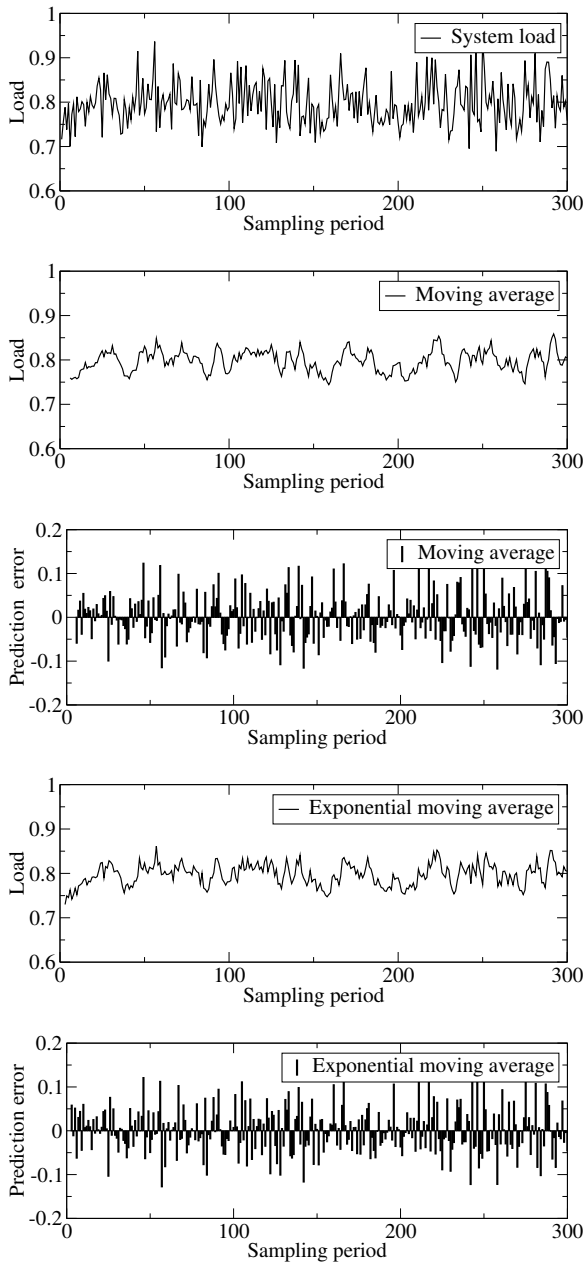


Fig. 7. The performance of different workload prediction methods.

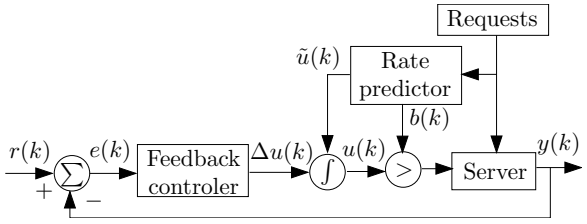


Fig. 8. The structure of the integrated strategy for PSD service provisioning.

1) The Feedback Controller: The feedback controller adjusts a class's processing rate using integral control. Proportional integral derivative (PID) control is one of the most classical control design techniques [20]. The advantage of proportional controller lies on its simplicity. It, however, cannot eliminate the steady-state error introduced in the process, which causes the target slowdown ratio unachievable. Derivative control takes into account the change of errors and has good responsiveness. Its drawback is that it is sensitive to measurement noises, which are common in provisioning of PSD services since we are essentially controlling a probability. Integral control is able to eliminate the steady-state error and avoid over-reactions to measurement noises.

To translate the PSD service model into the feedback control framework shown in Figure 8, the control loops must be determined first. In every loop, following aspects need to be determined: (1) the reference input, (2) the error, and (3) the output of the feedback control. In the integrated strategy, one class, such as class 1, is selected as the base class, and a control loop is associated with every other class. This leads to a total of $N-1$ control loops in the system.

In the control loop associated with class i, the reference input in the kth sampling period is

$$r_i(k) = \frac{\delta_i(k)}{\delta_1(k)}. \tag{20}$$

The output of the server is the achieved slowdown ratio of class i to the base class

$$y_i(k) = \frac{S_i(k)}{S_1(k)}. \tag{21}$$

If the rate allocation suggested by the rate predictor is accurate enough, the slowdown ratio error can be reduced to zero. In practice, because of the variance of traffic, the rate predictor cannot correct the error by itself. We define the error associated with class i as

$$e_i(k) = r_i(k) - y_i(k) = \frac{\delta_i(k)}{\delta_1(k)} - \frac{S_i(k)}{S_1(k)} \tag{22}$$

For class 1, we have

$$e_1(k) = 0 \qquad \text{for all } k. \tag{23}$$

Recall that the PSD service model aims to maintain prespecified slowdown ratios between different classes. By (19), we have

$$\frac{S_{i+1}}{S_1} = \frac{\lambda_{i+1}}{\lambda_1}\frac{c_1 - \lambda_1 m_1}{c_{i+1} - \lambda_{i+1} m_1}. \tag{24}$$

It suggests that slowdown ratios between classes can be controlled by adjusting their processing-rate ratios. Therefore, the integral feedback controller sums up the errors caused by previous rate allocation and adjusts the processing-rate ratios of class i to class 1 accordingly. The feedback controller output then is

$$\Delta u_i(k) = \Delta\frac{c_1(k)}{c_i(k)} = \Delta\frac{c_1(k-1)}{c_i(k-1)} + g \cdot e_i(k), \tag{25}$$

where g is the control gain, which determines the adjustment of the processing-rate ratio corresponding to the error.

The adjustment is incorporated with the rate predictor output

$$\tilde{u}_i(k) = \frac{c_1}{c_i}.$$

Thus, the processing-rate allocation for next sampling period $k+1$ is calculated as

$$\frac{c_1(k+1)}{c_i(k+1)} = \tilde{u}_i(k) + \Delta u_i(k) = \frac{c_1}{c_i} + g \int e_i(k)dk. \quad (26)$$

By (2), (23), and (26), the processing rate of class i of sampling period $k+1$ thus is

$$c_i(k+1) = \frac{1}{\left(\frac{c_1}{c_i} + g\int e_i(k)dk\right) \cdot \sum_{i=1}^{N} 1/\left(\frac{c_1}{c_i} + g\int e_i(k)dk\right)}. \quad (27)$$

There are three points worth noting here. First, it indicates that the processing rate of a class is affected by all errors incurred in the past, which is the feature of integral control. Second, a class's processing rate is also affected by the errors associated with other classes because of the requirement of the conservation law. Third, it suggests that the control gain g has great impact on the performance of the integrated strategy, which we shall discuss in Section V-G.

B. Design Issues

In this subsection, we discuss three design issues of the integrated strategy. First, recall that PSD services should be predictable and fair, it requires the maximal processing rate allocated to a class must be bounded so that no other classes will be overloaded. Let b_i denote bound of class i. The processing rate of class i then is subject to constraint

$$c_i \leq b_i, \quad 1 \leq i \leq N. \quad (28)$$

The determination of the rate limit of a class will be presented in Section V-F

Second, the control gain affects the performance of the integrated strategy significantly. A controller with large gain can respond quickly to errors. It, however, may cause large oscillations. Therefore, to reduce the overshoot introduced by the integral control, a small gain is preferred [44]. As we shall demonstrate in Section V-G, a small control gain can reduce the overshoot and the target slowdown ratio can be quickly achieved as well.

Finally, according to (24), the slowdown ratio between two classes depends on their class workloads. Such observation implies that the processing rates should also be adjusted according to class workloads. In our strategy, this is realized by the rate predictor so as to respond to the workload changes quickly.

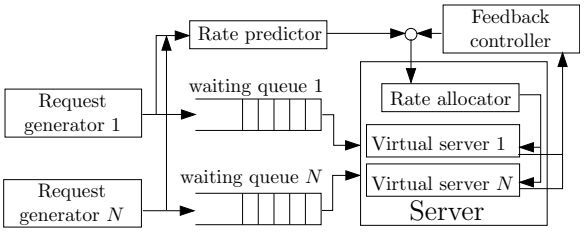


Fig. 9. The structure of the integrated rate allocation simulation model.

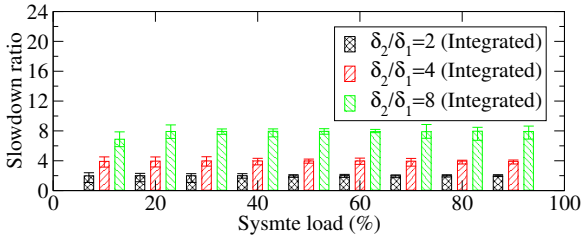


Fig. 10. Achieved slowdown ratios due to the integrated strategy.

C. Simulation Model

We extended the simulation model shown in Figure 1 to include a feedback controller. Its structure is outlined in Figure 9.

The feedback controller performs the integral control. It measures the difference between the target slowdown ratio and the achieved one for each class and then calculates rate allocations based on (27). The result then is compared with its rate limit. In the case that the calculated rate is larger than its rate limit, the actual processing rate is set to the rate limit.

Simulation parameters were set as the same as those in queueing-theoretic strategy except that the rate predictor estimates a class's processing rate every 10 sampling periods.

In order to prevent sudden spikes in the sampled slowdown ratio from causing large reactions in the controller, the slowdown ratio is smoothed using an exponential moving average method, which is similar as the one discussed in Section IV-G for workload prediction. Since there is no qualitative difference between different settings of the weight, in this paper we only present the result where the weight is 0.25.

D. Effectiveness of the Integrated Strategy

We first discuss the effectiveness of the integrated strategy by investigating its macroscopic and microscopic behaviors. The target slowdown ratios were set to 2, 4, and 8. Figure 10 gives the percentiles of the results due to the integrated strategy.

In comparison with the results due to the queueing-theoretic strategy as shown in Figure 3(a), it is clear that the integrated strategy has much finer-grained control over slowdown ratios than the queueing-theoretic strategy. For example, when the system load is 50% and the target is set to 4, the difference between the 95th and the 5th percentile is reduced from 5 in the queueing-theoretic strategy to 0.5 in the integrated strategy.

Moreover, the observation that the slowdown ratios are always larger than 1 means the PSD services are predictable,

whereas the queueing-theoretic strategy fails to meet this requirement. These observations conclude that the integrated strategy outperforms the queueing-theoretic strategy.

We next investigate the microscopic behaviors of these two strategies under a system with different number of classes. In these simulations, the system load was assumed to be 0.8. Figure 11(a) presents the results with two classes. It shows that, in the integrated strategy, the achieved slowdown ratio starts to converge to the target value within 60 sampling periods because the processing-rate allocation is adjusted adaptively. On the other hand, the queueing-theoretic strategy fails to provide PSD services because of traffic dynamics. Note that this figure shows the microscopic behaviors (e.g., one run) of these two strategies. The queueing-theoretic strategy is able to achieve the target slowdown ratio on average, which is demonstrated in Figure 3(a).

We also examine the impact of the processing-rate allocation strategies on a class's average queueing delay. Figure 11(b) depicts the queueing delay ratios due to these two strategies. It can be observed that the integrated strategy is able to provide proportional delay differentiation services. According to (9), the average slowdown of a class is proportional to its average queueing delay. Therefore, when two classes have the same m_{-1}, their average queueing delay ratio will be the same as their slowdown ratio. This is a salient property of the integrated strategy: providing differentiated services in terms of slowdown and delay simultaneously. On the other hand, the queueing-theoretic strategy cannot provide such differentiated services because of its inaccurate rate allocation in one run. Note that when requests of different classes follow different service time distributions, the differentiated delay services are not guaranteed.

Figure 12 shows the results with three classes. The long term workload of a class was assumed to be 0.25. From this figure we can see that, Although the achieved slowdown ratios converge to the targets slower than that with two classes, the integrated strategy can always achieve them. The queueing-theoretic strategy experiences difficulties in achieving the target slowdown ratios. These results further prove the superiority of the integrated strategy.

E. Robustness of the Rate Allocation Strategy

We carried out simulations to investigate the strategy's robustness under changing workload conditions. In the simulation, the initial workloads of both classes were assumed to be 0.4. At the 65th sampling period, the workload of class 1 was decreased to 0.2 and then changed back after the 115th sampling period. Figure 13(a) shows the average slowdown of all processed requests. For example, at the 65th sampling period, the result is the achieved slowdown average over all requests processed between the 10th and the 65th sampling periods. Figure 13(b) presents the average slowdown of a class for every two sampling periods (e.g., at the 115th sampling period, the value is the average slowdown of requests processed during the 114th and the 115th sampling periods).

It can be observed from Figure 13(b) that the average slowdown of both classes changes according to their workload conditions. In addition, as shown in Figure 13(a), such workload changes have little impact on the performance of the integrated strategy. This is because the rate predictor can quickly capture the changes of a class's workload and estimate its processing rate accordingly. Based on the estimation made by the rate predictor, the feedback controller is able to reduce the errors caused by changing workloads quickly.

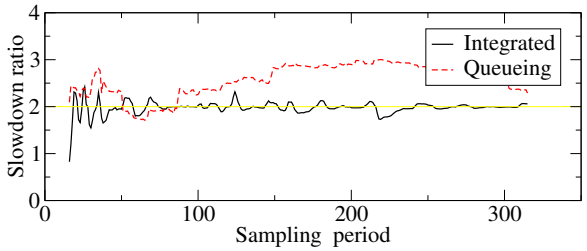
We also investigated the integrated strategy's robustness with real trace. In the simulation, the workload was generated based on the World Cup Soccer 98 server logs [3], which is publicly available. The results are presented in Figure 14. Because the simulated workload does not strictly meet the $M/G_P/1$ model, we can observe that the queueing-theoretic strategy fails to provide PSD services. On the other hand, in the integrated strategy, although the variation is clearly larger than that with accurate model, the integral controller can compensate the effect of the model inaccuracy and keep the achieved slowdown ratio around the target. We also show the results with rate predictor turned off. In comparison with the integrated results, there are little differences between them. This further suggests that robustness of the integrated strategy comes from the integral controller.

F. Effect of the Rate Limit

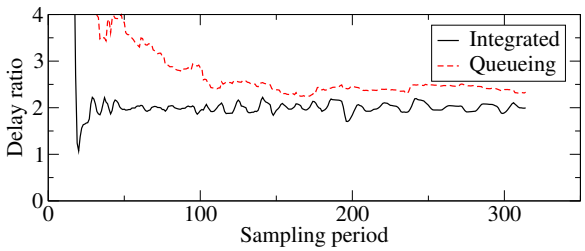
As discussed in Section V-B, to provide predictable and fair PSD services, the rate allocated to a class should have an upper bound. In this subsection, we examine the effect of the rate limit on the performance of the integrated strategy and then discuss how to obtain an appropriate rate limit of a class. To make analysis simple, in the simulations, we assumed there existed two classes that have the same average load in the system with the same rate limit. Moreover, the same request stream was used in these simulations to make the results comparable. Figure 15 shows the simulation results.

From this figure we can observe that, the larger the rate limit, the slower the achieved slowdown ratio converges to the target and the larger the variance is. For example, when the rate limit was set to 0.8, it took more than 100 sampling periods before the achieved slowdown ratio converged to the target; when the rate limit was set to 0.6, it converged in 30 sampling periods. It is because that, with a large rate limit of one class, another class may be overloaded because its allocated rate is not able to process the arrived requests in time and vice versa. Such occasional overload causes large errors. In order to reduce the errors, the rate allocation in next sampling period is adjusted, which leads to large oscillations and slows the convergence.

As the rate limit decreases, the converging rate is increased. For example, with the rate limit was set to 0.52, the target slowdown ratio is reached more quickly than when the limit was set to 0.6. It, however, yields steady errors. We argue that a too small rate limit can damage the capability of the integrated strategy to reduce the variance. For example, when the rate limit was set to 0.52, recall that both classes had the

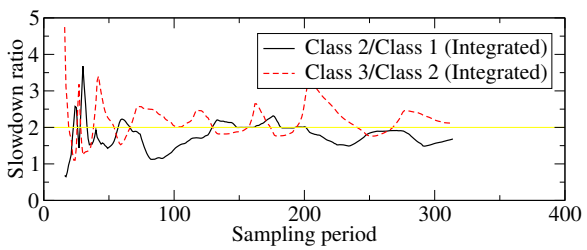


(a) Slowdown ratio.

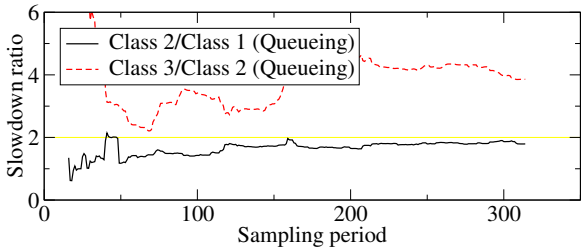


(b) Delay ratio.

Fig. 11. Simulation results with two classes. The target ratio was set to 2.

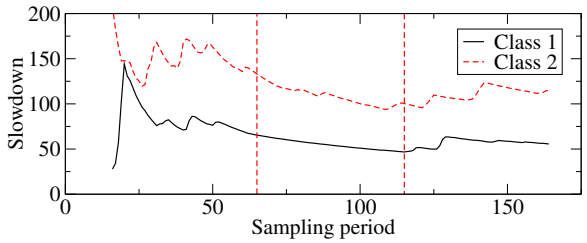


(a) Integrated strategy.

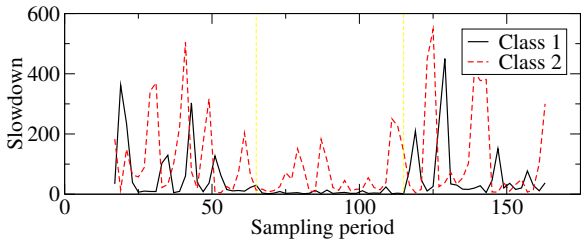


(b) Queueing-theoretic strategy.

Fig. 12. Simulation results with three classes and $\delta_2/\delta_1 = \delta_3/\delta_2 = 2$.

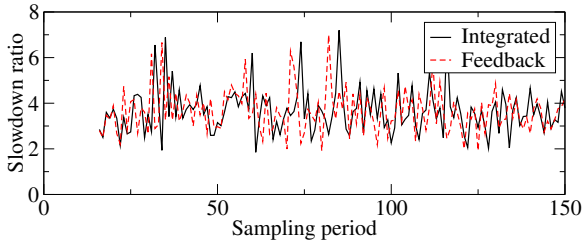


(a) Average slowdown over all requests.



(b) Average slowdown over two sampling periods.

Fig. 13. The performance of the integrated strategy under changing environments. ($\delta_2/\delta_1 = 2$)



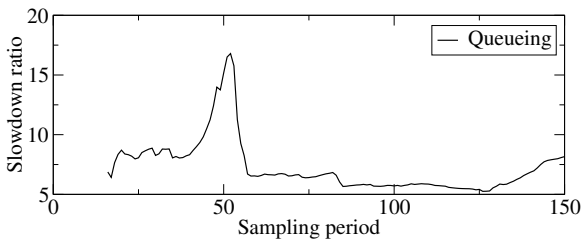
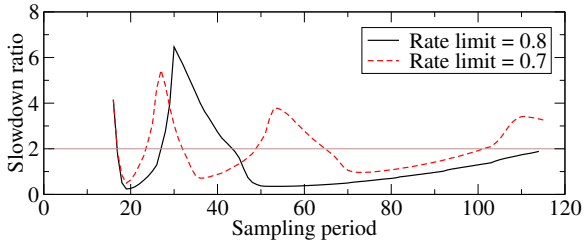


Fig. 14. Simulation results with real trace. $\delta_2/\delta_1 = 4$.



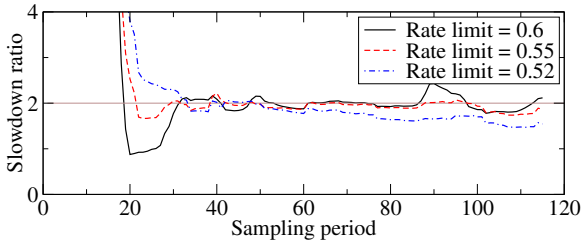


Fig. 15. Effect of rate limits on the performance of the integrated strategy. ($\delta_2/\delta_1 = 2$)

same rate limit and due to the constraint of (2), the processing rate that could be allocated to a class ranged from 0.52 to 0.48. Hence, when an error happened, the integrated strategy was unable to correct it effectively.

Essentially, the rate limit of a class depends on the basic resource (e.g., the processing rate) demanded by other classes. Aforementioned, the first term of (18) is the baseline required by a class. Therefore, it is necessary that the rate allocated to a class should be small enough to prevent other classes from being overloaded. The rate limit of class i then is

$$b_i = 1 - \sum_{j=1...N, j \neq i} \lambda_j m_1. \quad (29)$$

Accordingly, the rate limit of both classes should be 0.6 in the simulations. This constraint is verified by Figure 15: With the rate limit was set to 0.6, the achieved slowdown ratio quickly converges to the target with small variance.

G. Determination of the Control Gain

Aforementioned, the control gain g has significant impact on the performance of the integrated strategy. We conducted simulations to determine an appropriate setting. In these simulations, two classes were assumed. The results with different control gains are shown in Figure 16.

From these results we can observe that, the larger the control gain, the faster it converges to the target slowdown ratio. With a large gain, the error is amplified. The feedback controller, thus, responds more quickly than with a small control gain. In addition, due to the rate limit of every class, the variance is small even with large gains. Simulation with different workload conditions were also carried out. They yielded similar results as shown in this figure.

A large gain, however, may cause excessive slowdown oscillation of a class. In Figure 17, the average slowdown of class 2 with different gains is presented. The results are averaged over every 5 sampling periods. It is clear that the average slowdown becomes relative stable after the 75th sampling period when the control gain is set to 1 or 2. In the situations where the control gain is larger than 4, a large variance can always be observed.

In essence, the oscillation of average slowdown is caused by the changes of processing-rate allocation. Figure 18 shows the results of the rate allocation with the control gain as 2 and 16. Recall that a class has a limit on its maximal possible processing rate. The rate allocated to other classes, therefore, has a lower bound. For example, in the simulation, the rate limit of either class was around 0.75 and its minimal processing rate then was around 0.25. These results show that, with a large control gain, the processing rate of a class oscillates between 0.75 and 0.25 for a longer time and more frequently with a large gain 16 than a small gain 2. As a result, excessive slowdown oscillations are observed.

As pointed out in Section V-B, to reduce the overshoots, a small control gain is preferred. Aforementioned, slowdown is used to measure the service quality received by clients. In practice, a small slowdown oscillation is desirable. Large and frequent changes of slowdown in short time-scales frustrate end users. Moreover, as shown in Figure 18(b), large control gain causes excessive rate-allocation fluctuations and results in high scheduling overhead. With respect to these considerations, a small control gain, such as 1 or 2, is preferred.

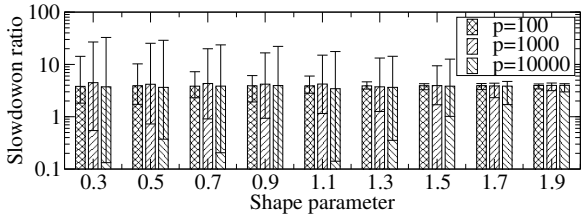


Fig. 19. Achieved slowdown ratios due to the integrated strategy with different shape parameters and upper bounds. $\delta_2/\delta_1 = 4$.

H. Impact of the Shape Parameter and Upper Bound

In this subsection, we examine the impact of the shape parameter (α) and upper bound (q) of the bounded Pareto distribution on the performance of the integrated strategy. Figure 19 shows the 5th, 50th, and 95th percentiles of achieved slowdown ratios between two classes with different shape parameter ($0 < \alpha < 2$) and upper bound settings ($q = 100, 1000, 10000$).

The first observation is that, with different shape parameter and upper bound settings, the integrated strategy can always achieve the target; that is, the 50th percentiles are always close to the target. This further demonstrates the effectiveness and robustness of the proposed strategy.

We can also observe that the variance of achieved slowdown ratios is affected by the shape parameter and upper bound settings. Similar as the queueing-theoretic strategy's behavior as shown in Figure 6, the variance increases with the decreasing of the shape parameter and the increasing of the upper bound. How to further reduce the variance is one of our future work. One possible solution is using shortest job first or shortest remaining processing time first algorithms to schedule requests from the same class. Thus, we can reduce the mean response time and slowdown of small requests [11], [13], [22].

VI. Implementations and Experimental Results

We implemented the integrated strategy on Apache Web server 1.3.31 running on Linux 2.6. Apache Web server is normally executed with multiple processes (or threads). At the start-up, a parent server process sets up listening sockets and creates a pool of child processes. The child processes then listen on and accept client requests from these sockets. Recall that, by (26), we can control the slowdown ratio between two classes by manipulating their processing-rate ratio. Since every child process in Apache Web server is identical, in the implementation, we realize the processing-rate allocation by controlling the number of child processes that a class is allocated.

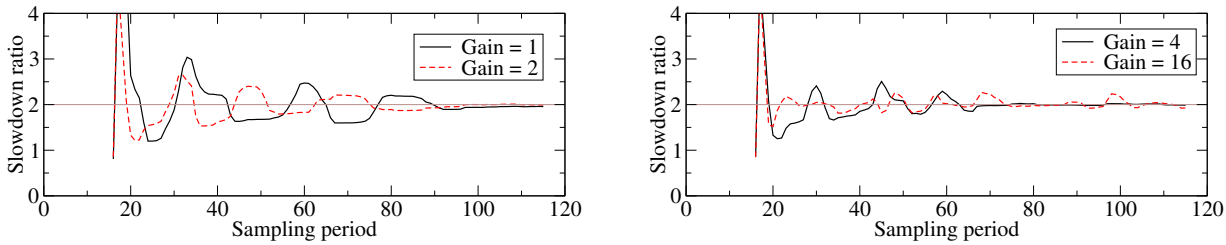


Fig. 16. The effect of control gain on slowdown ratio.

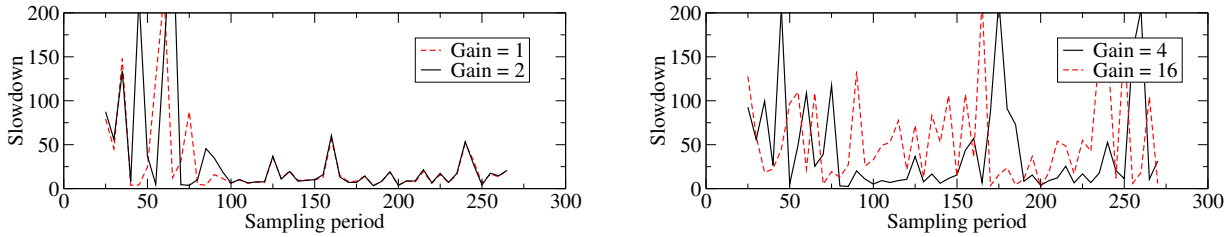
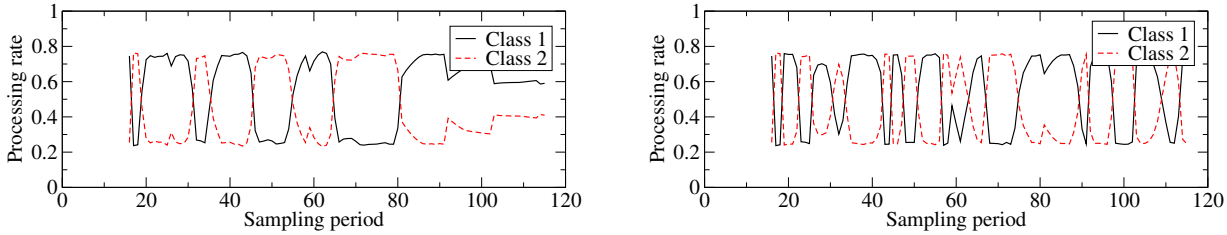


Fig. 17. The effect of the control gain on average slowdown.



(a) $g = 2$. (b) $g = 16$.

Fig. 18. Rate allocation under different control gains. ($\delta_2/\delta_1 = 2$)

A. Implementation Structure

The implementation structure of the integrated strategy is presented in Figure 20. It consists of a classifier, a rate predictor, a feedback controller, and a rate allocator.

The classifier determines a request's class according to rules defined by service providers. In our implementation, the rules are based on the request's header information (e.g., IP address and port number). To make simulation simple, a request's class type can also be determined randomly according to the arrival ratio between classes. For example, for two classes, the probability that a request to be classified as class 1 is 25% if the arrival ratio of class 1 to class 2 is $1/3$. After being classified, a request is stored in its corresponding waiting queue. Associated with each waiting queue is a record of traffic information, such as arrival rate and service time.

The rate predictor obtains the arrival rate and service time from waiting queues and estimates the processing rate that a class should be allocated according to (18).

The feedback controller inferences the achieved slowdown ratios between different classes and calculates the processing rate of a class according (27) by incorporating the rate estimation from the rate predictor. It then sets the processing rate (number of allocated processes) on the rate allocator.

The rate allocator enforces the processing-rate allocation. In our implementation, the Apache Web server was modified to accept requests from the unix domain socket. When a child process calls *accept()* on the unix domain socket, a signal is sent to the rate allocator. Upon receiving such signal, the rate allocator scans the waiting queues to check if there is a backlogged class whose number of allocated processes is larger than that it is occupying. If such class exists, its head-of-line request and the class type are passed to the child process through the unix domain socket; otherwise, a flag is set. Whenever a new request arrives, the flag is checked first and the waiting queues are rescanned. The rate allocator also increases a counter that records the number of occupied processes by the class. After handling a client request, the child process sends the request's class type back to the rate allocator, which then decreases the corresponding counter. In addition, the Apache Web server was modified so that the processing time of an object is proportional to its size.

In practice, client-perceived performance of a Web server is based on a whole web page, which can be a single file or an html file with multiple embedded objects. According to HTTP 1.1, a client first establishes a persistent connection with the Web server and sends requests for the web page and possible embedded objects. The queue delay of a web page thus is defined as the time interval between the arrival of a

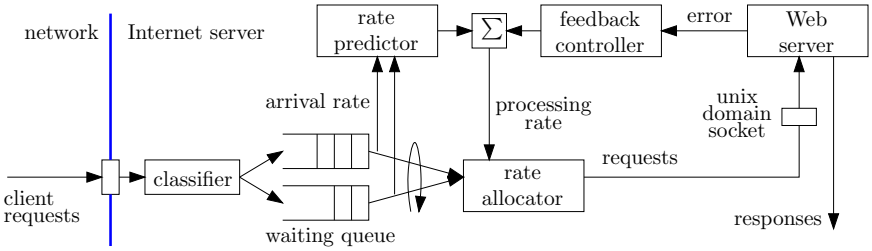


Fig. 20. The implementation structure of the integrated strategy for PSD service provisioning.

connection request and the time the connection is passed to a child process of Apache Web server. The service time is defined as the time interval between the passing of connection request and the time the response of the whole web page is sent to the client. We determine the end of response based on the "referer" header field of an HTTP request. An embedded object uses this to indicate which web page it is contained. Similar method is also proposed in [39]. Both the queueing delay and the service time are measured on the Web server. In order to provide end-to-end differentiated services, support from network side is necessary [17], [47].

B. Experimental Results

We conducted experiments to demonstrate the performance of the integrated strategy. As shown in Figure 14(b), queueing-theoretic strategy fails to provide PSD services because the realistic traffic has different characteristics from the queueing model. The experimental environment consisted of four PCs running on a 100 Mbps Ethernet. Each PC was a Dell PowerEdge 2450 configured with dual-processor (1 GHz Pentium III) and 512 MB main memory. We installed one Apache Web server on one PC while a commonly used Web traffic generator SURGE [5] ran on other PCs. In the generated web objects, the maximum number of embedded objects in a given page was 150 and the percentage of base, embedded, and loner objects were 30%, 38%, and 32%, respectively. The workload of emulated requests was controlled by adjusting the total number of concurrent user equivalents (UEs) in SURGE. Note that the fixed number of UEs does not affect the representativeness (i.e., self-similarity and high dynamics) of the generated Web traffic [5].

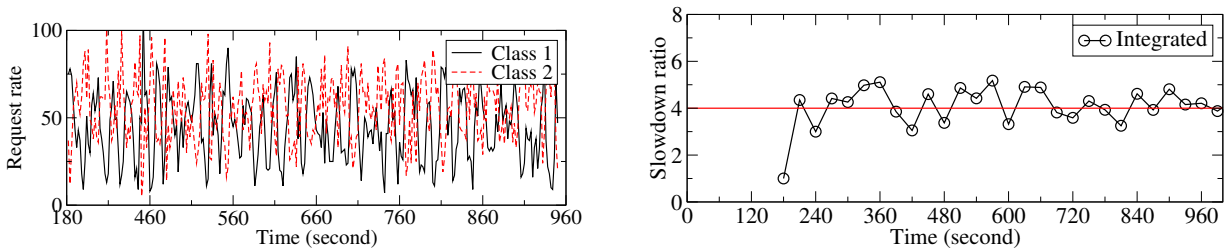
The Apache Web server was executed with support of HTTP 1.1. The maximal concurrent child server processes was set to be 128. The system was first warmed up for 180 seconds. After that, the controller was turned on. Since the sampling period should be reasonable large so that the scheduler can determine the effect of resource allocation on the achieved slowdown ratio, the sampling period was set to 3 seconds and the rate predictor estimates processing rate every 10 sampling periods. The control gain in these experiments was set to 1 as suggested in Section V-G.

We first conducted experiment to evaluate the effectiveness of the integrated strategy. Figure 21 shows the workload configurations and corresponding results. In this experiment, we assumed there were two classes and the target slowdown ratio was set to 4. The number of UEs was set to 300. The arrival ratio of class 1 to class 2 is randomly changed within $1/9$ and $9/1$ for every 3 seconds. Figure 21(a) depicts the arrival rate of both classes for every 3 seconds. It indicates that the workload of a class changes frequently and abruptly. From Figure 21(b) we can observe that, although the workload generated by SURGE does not strictly meet the $M/G_P/1$ model, the achieved slowdown ratios can still be kept around the target. This demonstrates that the integrated strategy is effective in providing PSD services under realistic workload configurations. It also shows that the suggested small control gain is a good choice in a real system.

We also carried out experiments to evaluate the performance of the integrated strategy under different target slowdown ratios, different workloads, and multiple classes. Figure 22 shows the 5th, 50th, and 95th percentiles of achieved slowdown ratios. In these experiments, the workload has similar configuration as shown in Figure 21(a). In Figure 22(a) we can observe that the target ratios (2, 4, and 8) can be achieved under different workload conditions. Figure 22(b) suggests that the target ratios can also be achieved under multiple classes. Furthermore, in both figures we can observe that the variance of achieved ratios becomes relative large with the increase of workloads. For example, in Figure 22(b), when the number of UEs is 600 and 700, the difference between the 95th and the 5th percentiles is 6.5 and 9.3, respectively. In the implementation, to control the average slowdown of a class, we control its average queueing delay by adjusting the number of its allocated processes. Because all processes have the same priority, this method has little impact on the service time of a request. With the increase of workload, a class's average queueing delay increases. Therefore, the slowdown and its variance increases. One possible solution is to control the service time of a request by adjusting the priority of child processes according to workload conditions. This will be part of our future work.

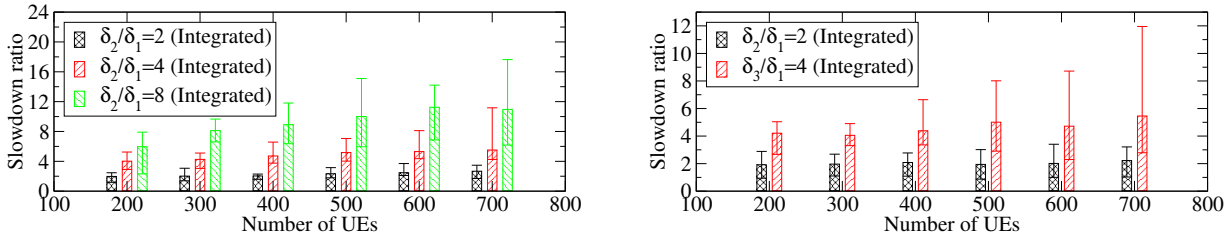
VII. Related Work

Providing differentiated services to network applications and clients has become popular. In order to provide end-to-end differentiated services, issues on both network-side and server-side should be addressed. On network side, many previous efforts focused on providing queueing delay differentiation and many packet scheduling algorithms have been proposed to achieve proportional delay differentiation in network routers. Representative algorithms include waiting time priority [16], proportional average delay [17], mean delay proportional [37],



(a) Workload configuration. (b) Achieved slowdown ratios.

Fig. 21. Performance of the integrated strategy where the target slowdown ratio is set to 4.



(a) Two classes. (b) Three classes.

Fig. 22. Performance of the integrated strategy under different target ratios, different workloads, and multiple classes.

VirtualLength [46], and Little's average delay [47]. These kinds of algorithms can be tailored for providing differentiated delay services on Internet servers [29].

Another important performance metric is loss rate. In [15] and [26] the focus is on providing differentiated loss services to different classes. For example, in [26], the authors proposed an algorithm in which a packet's drop probability is calculated based on its arrival process so as to provide differentiated loss services.

On the server side, a primary focus has been on priority-based request scheduling with admission control for differentiated services in terms of response time. For example, in an early work [2], the authors addresses strict priority scheduling strategies for controlling CPU utilization in Web content hosting servers. Service differentiation was introduced by assigning strict priorities to requests for different content. The results showed that service differentiation can be achieved but the quality spacings among different classes cannot be guaranteed by this kind of strict priority scheduling. One objective of our work is to guarantee the quality spacings.

When a system is overloaded, admission control drops requests from lower priority classes so as to guarantee service quality received by those from higher priority classes [10], [12], [29]. In this paper, we mainly focus on providing differentiated services when resource demands of workloads are within resource capacity of the system. This is complementary to the previous work on admission control for overload protection and service differentiation.

Feedback control theory has been applied as an analytic method for providing differentiated services. In [1], [34], [41], the authors first used system identification to model Internet servers based on their past behaviors. Then a closed-loop system with an integral control law was applied to provide differentiated services in which different classes had different average queue length or queueing delay. Another example is differentiated caching services in which different classes have proportional cache hit ratios [28], [36]. In contrast, using feedback control theory aside, our strategy takes advantage of queueing theory to predict the processing rate that should be allocated to a class according to its predicted load conditions.

The integration of queueing theory and feedback control has been applied in providing differentiated delay services. In [35], the authors extended the strategy presented in [44] to implement relative delay differentiation services. The strategy predicted the expected delay from a class's load using queueing theory, and a feedback controller was used to adjust the resource allocation based on the error between the achieved delay ratio and the target one. A connection scheduler was used as an actuator to control the relative delays of different classes. In contrast, the objective of this paper is to provide differentiated service in terms of slowdown, which needs consideration of queueing delay as well as a request's service time.

Slowdown is an important performance metric of responsiveness because it is desirable that a request's delay be proportional to its processing requirement. In [22], the authors evaluated online job assignment strategies in a distributed server system, where the workload was also heavy-tailed and job size was unknown to the scheduler. The primary objective was to minimize mean slowdown of all jobs in the distributed system. In contrast, we want to provide proportional slowdown differentiation services to requests from different classes. Slowdown is also commonly known as normalized response time or stretch factor [25]. In [51], the authors adopted stretch factor as the performance metric for service differentiation in a cluster of Internet servers, which are modeled as an

$M/M/1$ queueing system. We note that for an $M/M/1$ FCFS queue with the unbounded exponential service time density distribution, there is no valid stretch factor or slowdown because m_{-1} is not existent. For an $M/M/1$ FCFS queue with a bounded exponential service time distribution, there is no closed-form expression for the stretch factor or slowdown because m_{-1} only has a definite value when the lower bound and the upper bound of service time are given. In addition, recent Internet workload measurements indicate that for many Web applications the exponential distribution is a poor model for service time distribution and that a heavy-tailed distribution is more accurate [3], [22]. In this paper, we investigate the problem of processing-rate allocation for PSD provisioning under a popular heavy-tailed traffic pattern (bounded Pareto).

There are also efforts in resource management for service differentiation on multimedia servers; see [8], [50] for examples. Multimedia connections impose very different load characteristics compared to those in traditional Web servers. The quality of multimedia content, instead of responsiveness, is often used as the primary QoS metric in service differentiation provisioning, such as image size and resolution, video streaming bandwidth, etc. Content adaptation techniques, such as video transcoding, are the enabling technologies. Note that the work in this paper focuses on responsiveness differentiation on traditional Web servers.

VIII. Conclusion

Slowdown is an important performance metric of Internet services because it takes into account the delay and service time of a request simultaneously. Although delay and loss rate have been studies in QoS-aware design for a long time, there was little work in providing differentiated services in terms of slowdown. In this paper, we have investigated the problem of processing-rate allocation for PSD provisioning on Internet servers. We have derived a closed-form expression of the expected slowdown in an $M/G/1$ FCFS queueing system with bounded Pareto service time distribution. We have proposed a queueing-theoretic strategy for processing-rate allocation. In order to improve the predictability of the services and reduce the variance, we have developed and integrated a feedback controller into the strategy. Simulation results have showed that the queueing-theoretic strategy can guarantee PSD services on average. The integrated strategy achieves the objective at a fine-grained level. Our future work will focus on implementing the strategies in Web servers and evaluating their performance using real traces.

References

[1] T. F. Abdelzaher, K. G. Shin, and N. Bhatti. Performance guarantees for Web server end-systems: A control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):80–96, January 2002.

[2] J. Almeida, M. Dabu, A. Manikutty, and P. Cao. Providing differentiated levels of service in web content hosting. In *Proceedings of ACM SIGMETRICS Workshop on Internet Server Performance*, pages 91–102, June 1998.

[3] M. Arlitt and T. Jin. A workload characterization study of the 1998 world cup web site. *IEEE Network*, 14(3):30–37, May-June 2000.

[4] M. F. Arlitt and C. L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transaction on Networking*, 5(5):631–645, October 1997.

[5] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of ACM Sigmetrics '98 Conference*, pages 151–160, June 1998.

[6] M. A. Bender, S. Chakrabarti, and S. Muthukrishnan. Flow and stretch metrics for scheduling continuous job streams. In *Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 270–279, 1998.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. *An Architecture for Differentiated Services*. IETF, Request for Comments 2475, December 1998.

[8] S. Chandra, C. S. Ellis, and A. Vahdat. Application-level differentiated multimedia web services using quality aware transcoding. *IEEE Journal on Selected Areas in Communications*, 18(12):2544–2265, December 2000.

[9] C. Chekuri, A. Goel, S. Khanna, and A. Kumar. Multi-processor scheduling to minimize flow time with ϵ resource augmentation. In *Proceedings of ACM Symposium on Theory of Computing*, pages 363–372, 2004.

[10] X. Chen and P. Mohapatra. Performance evaluation of service differentiating Internet servers. *IEEE Transactions on Computers*, 51(11):1368–1375, November 2002.

[11] L. Cherkasova. Scheduling strategy to improve response time for web applications. In *Proceedings of High Performance Computing and Networking, HPCN'98-Europe*, pages 305–314, 1998.

[12] L. Cherkasova and P. Phaal. Session-based admission control: A mechanism for peak load management of commerical web sites. *IEEE Transactions on Computers*, 51(6):669–685, June 2002.

[13] M. E. Crovella, R. Frangioso, and M. Harchol-Balter. Connection scheduling in web servers. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, Boulder, Colorado, USA, October 1999.

[14] M. E. Crovella, M. Harchol-Balter, and C. D. Murta. Task assignment in a distributed system: Improving performance by unbalancing load. In *Proceedings of ACM Sigmetrics '98 Conference*, pages 268–269, June 1998.

[15] C. Dovrolis and P. Ramanathann. Proportional differentiated services, part ii: Loss rate differentiation and packet dropping. In *Proceedings of International Workshop on Quality of Service*, Pittsburgh, June 2000.

[16] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proceedings of SIGCOMM*, pages 109–120, Cambridge, Massachusetts, United States, 1999.

[17] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. *IEEE/ACM Transactions on Networking*, 10(1):12–26, 2002.

[18] A. B. Downey. A parallel workload model and its implications for processor allocation. In *Proceedings of High Performance Distributed Computing*, pages 112–123, August 1997.

[19] D. G. Feitelson and L. Rudolph. Metrics and benchmarking for parallel job scheduling. In *Proceeds of International Parallel Processing Symposium 98 Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1459, pages 1–24. Springer-Verlag, 1998. Lecture Notes in Computer Science.

[20] G. F. Franklin, J. D. Powell, and A. Emami-naeini. *Feedback Control of Dynamic Systems*. Prentice Hall, 4th edition, 2002.

[21] Free Software Foundation. *GSL – GNU Scientific Library*. Available: http://www.gnu.org/software/gsl/.

[22] M. Harchol-Balter. Task assignment with unknown duration. *Journal of ACM (JACM)*, 49(2):260–288, 2002.

[23] M. Harchol-Balter, M. E. Crovella, and C. D. Murta. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 59(2):204–228, 1999.

[24] M. Harchol-Balter and A. B. Downey. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Sysmtems*, 15(3):253–285, 1997.

[25] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(2):207–233, May 2003.

[26] Y. Huang and R. Guérin. A simple fifo-based scheme for differentiated loss guarantees. In *Proceedings of International Workshop on Quality of Service*, pages 96–105, 2004.

[27] L. Kleinrock. *Queueing Systems. Volume II: Computer Applications*. John Wiley & Sons, 1975.

[28] B.-J. Ko, K.-W. Lee, K. Amiri, and S. Calo. Scalable service differentiation in a shared storage cache. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, pages 184–193, May 2003.

[29] S. C. Lee, J. C. Lui, and D. K. Yau. Admission control and dynamic adaptation for a proportional-delay diffserv-enabled web server. In *Proceedings of Sigmetrics*, pages 172–182, 2002.

[30] S. C. Lee, J. C. Lui, and D. K. Yau. A proportional-delay diffserv-enabled web server: Admission control and dynamic adaption. *IEEE Transactions on Parallel and Distributed Systems*, 15(5):385–400, May 2004.

[31] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

[32] M. K. Leung, J. C. Lui, and D. K. Yau. Adaptive proportional delay differentiated services: Characterization and performance evaluation. *IEEE/ACM Transactions on Networking*, 9(6):801–817, December 2001.

[33] Z. Liu, N. Niclausse, and C. Jalpa-Villanueva1. Traffic model and performance evaluation of web servers. *Performance Evaluation*, 46(2–3):77–100, October 2001.

[34] C. Lu, T. F. Abdelzaher, J. A. Sankovic, and S. H. Son. A feedback control approach for guaranteeing relative delays in web servers. In *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2001.

[35] Y. Lu, T. F. Abdelzaher, C. Lu, L. Sha, and X. Liu. Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers. In *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 208–217, May 2003.

[36] Y. Lu, A. Saxena, and T. F. Abdelzaher. Differentiated caching services: A control-theoretical approach. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, pages 615–622, April 2001.

[37] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. Delay differentiation and adaptation in core stateless networks. In *Proceedings of IEEE Infocom*, pages 421–430, Tel-Aviv, Israel, April 2000.

[38] K. Nichols, V. Jacobson, and L. Zhang. *A Two-bit Differentiated Services Architecture for the Internet*. IETF, Request for Comments 2638, July 1999.

[39] D. P. Olshefski, J. Nieh1, and E. Nahum. ksniffer: Determining the remote client perceived response time from live packet streams. In *Proceedings of Usenix Operating Systems Design and Implementation (OSDI)*, 2004.

[40] 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 Transactions on Networking*, 1(3):344–357, 1993.

[41] S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus. Using control theory to achieve service level objectives in performance management. *Journal of Real-time Systems*, pages 127–141, 2002.

[42] V. Paxson and S. Floyd. Wide area traffic: The failure of possion modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[43] A. Riska, W. Sun, E. Smirni, and G. Ciardo. Adaptload: Effective balancing in clustered web servers under transient load conditions. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, July 2002.

[44] L. Sha, X. Liu, Y. Lu, and T. F. Abdelzaher. Queueing model based network server performance control. In *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, pages 81–90, 2002.

[45] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource manangement. In *Proceedings of First Symposium on Operating System Design and Implementation*, pages 1–11, Monterey, California, USA, November 1994.

[46] J. Wei, Q. Li, and C. Xu. VirtualLength: A new packet scheduling algorithm for proportional delay differentiation. In *Proceedings of International Conference on Computer Communications and Network (ICCCN)*, pages 331–336, October 2003.

[47] J. Wei, C. Xu, and X. Zhou. A robust packet scheduling algorithm for proportional delay differentiation services. In *Proceedings of Globecom*, November 2004.

[48] W. Willinger and V. Paxson. Where mathematics meets the Internet. *Notices of the American Mathematical Society*, 45(8), August 1998.

[49] X. Zhou, J. Wei, and C. Xu. Processing rate allocation for proportional slowdown differentiation on Internet servers. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, April 2004.

[50] X. Zhou and C. Xu. Harmonic proportional bandwidth allocation and scheduling for service scheduling on streaming servers. *IEEE Transactions on Parallel and Distributed Systems*, 15(9), 2004.

[51] H. Zhu, H. Tang, and T. Yang. Demand-driven service differentiation in cluster-based network servers. In *Proceedings of IEEE Infocom*, pages 679–688, 2001.