Processing Rate Allocation for Proportional Slowdown Differentiation on Internet Servers

Xiaobo Zhou
Department of Computer Science
University of Colorado, Colorado Springs, 80933
zbo@cs.uccs.edu

Jianbin Wei and Cheng-Zhong Xu
Department of Electrical & Computer Engg.
Wayne State University, Detroit, 48202
{jbwei, czxu}@ece.eng.wayne.edu

Abstract

A proportional differentiation model states that quality of service of different classes of Internet traffic should be kept proportional to their pre-specified differentiation parameters, independent of the class loads. The model has been applied in the proportional queueing delay differentiation (PDD) in both network core and network edges. However, in the server side, an important and interesting performance metric is slowdown, the ratio of a request's queueing delay to its service time. Slowdown is important because it is desirable that a request's delay be proportional to its processing requirement.

In this paper, we investigate the problem of processing rate allocation for proportional slowdown differentiation (PSD) on Internet servers. Existing algorithms for PDD provisioning in the network side are not applicable to PSD provisioning in the server side because slowdown is not only dependent on a job's queueing delay but also on its service time, which varies significantly depending on the requested services. We first derive a closed form expression of the expected slowdown in an $M/G_P/1$ FCFS queue, which is an $M/G/1$ FCFS queue with a typical heavy-tailed service time distribution (Bounded Pareto distribution). PSD provisioning is realized by deploying a task server for handling each request class in a FCFS way. We then develop a strategy of processing rate allocation for the task servers in support of PSD provisioning. Simulation results have showed that the proposed rate allocation strategy can provide predictable and controllable PSD services on the servers.

1 Introduction

Internet applications and clients have very diverse service expectations, demanding for provisioning of different levels of quality of service (QoS) on the Internet. This service differentiation provisioning problem was firstly addressed in the network core. Differentiated Services (DiffServ) architecture, which aims to provide different QoS levels among multiple classes of aggregated traffic flows, has been an active research topic since it was formulated by IETF in 1998 [5]. Basically, there are two types of DiffServ scheme. One is absolute DiffServ, in which each class receives an absolute share of resource usages. The other is relative DiffServ, in which a class with a higher desired QoS level (referred to as a higher priority class) will receive better (at least no worse) service than a lower priority class. Although absolute DiffServ is desired to Internet services like audio/video streaming applications that have hard time constraints, relative DiffServ is sufficient and more attractive for soft real-time Web applications. In order for a relative DiffServ scheme to be effective, the scheme must satisfy two basic properties: *predictability* and *controllability*. Predictability requires that higher priority classes receive better or no worse service quality than lower priority classes, independent of the system load conditions. Controllability requires that the scheduler contain a number of controllable parameters that are adjustable for the control of quality spacings between classes.

The proportional differentiation model [7] states that certain class performance metrics should be proportional to the differentiation parameters the network operator chooses, independent of the class loads. The model has been accepted as an important relative DiffServ model and been applied in the proportional delay differentiation (PDD) in packet scheduling [8]. In this model, the network traffic is categorized into N classes of service. Each class is assigned a queueing delay differentiation parameter. The packet scheduler of a router gives different priorities to different classes. The objective is to keep the ratio of average delay of a higher priority class to that of a lower priority class equal to the pre-specified value.

Since the PDD model was formulated in [8], many algorithms have been designed to achieve the PDD provisioning in the network routers. They can be classified into two categories: rate-based [8, 18], and time-dependent priority

based [9, 11, 17, 19, 23]. The end-to-end time performance of Internet services is not only due to the packet transmission delay in the network core, but also due to the processing and queueing delay on the servers. Therefore, servers are a major force in DiffServ provisioning. Those algorithms can be tailored for PDD provisioning on servers [15]. However, in the server side, an important and interesting performance metric is slowdown, the ratio of a request's queueing delay to its service time. Both queueing delay and response time are major performance metrics. But they are not suitable to compare requests that have very different resource demands. Actually, clients are likely to anticipate short delays for "small" requests, and are willing to tolerate long delays for "large" requests. Thus, it is desirable that a request's delay be proportional to its processing requirement. A high slowdown can also indicate that the system is heavily loaded [25].

Slowdown is being used as a fundamental performance metric of responsiveness on Internet servers [13, 21, 25, 24]. However, few work exists for slowdown differentiation. Existing time-dependent priority based PDD packet scheduling algorithms cannot be tailored to achieve proportional slowdown differentiation (PSD) because they adjust the priority of a backlogged request class according to the experienced queueing delay of its head-of-line request or backlogged requests, or both. Queueing delay aside, slowdown is also dependent on the service time of a request, which is costly, if not impossible, to predict *a priori* for priority request scheduling. Rate-based PDD packet scheduling algorithms can be applied for server-side proportional delay differentiation, but not PSD services because PSD provisioning needs to consider queueing delay together with service time.

In this paper, we investigate the problem of PSD provisioning on Internet servers. The problem is important because the proportional model is a widely accepted DiffServ model and slowdown is a key QoS metric in the server side. It is challenging because in order to meet the needs of differentiation predictability and controllability, a closed form expression of expected slowdown is required for the formulation of resource allocation and scheduling. We consider a popular heavy-tailed distribution, *Bounded Pareto*, for the service time distribution. In the paper, an $M/G/1$ queue with Bounded Pareto service time distribution is referred to an $M/G_P/1$ queue. We give the expected slowdown of an $M/G_P/1$ FCFS queue in a closed analytic form. We then propose a processing rate allocation strategy for PSD provisioning on servers, based on the assumption that the processing rate of a server can be proportionally allocated to a number of *task servers*. Each task server i ($i = 1 \ldots N$) represents a processing unit that handles requests from the same class in a FCFS way. Recently, there has been a renewal of interests in achieving the proportional-share resource scheduling among multiple queues in both operating systems and networks; see GPS [14], PGPS [20], and Lottery Scheduling [22] for examples. They provide a base for the processing rate allocation in our work. Note that a task server is an abstract concept in the sense that it can be a child process in a multi-process server, or a thread in a multi-thread server [6].

The structure of the paper is as follows. Section 2 gives a slowdown model for $M/G_P/1$ FCFS queues. Section 3 presents the processing rate allocation strategy for PSD provisioning. Section 4 focuses on simulation issues and performance evaluation. In Section 5, we review other related processing rate allocation and scheduling disciplines in the DiffServ areas. Section 6 concludes the paper.

2 Slowdown Modeling in an $M/G_P/1$ Queue

2.1 Preliminaries of Slowdown

Recent Internet workload measurements indicate that, for many Web applications, a heavy-tailed distribution is a more accurate model for service time distribution than the exponential distribution [3, 13]. In general, a heavy-tailed distribution is one for which $\Pr\{X \leq x\} \sim x^{-\alpha}, 0 < \alpha < 2$, where X denotes the service time density distribution.

The *Pareto* distribution is a typical heavy-tailed distribution, with probability mass function

$$f(x) = \alpha k^\alpha x^{-\alpha-1} \qquad \alpha, k > 0, x \geq k, \tag{1}$$

and cumulative distribution function $F(x) = \Pr\{X \leq x\} = 1 - (\frac{k}{x})^\alpha$.

In practice, there is some upper bound on the maximum size of a job. As the work in [13], throughout this paper, we model job sizes as being generated i.i.d. from a distribution that is heavy-tailed, but has an upper bound. This *Bounded Pareto* distribution is characterized by three parameters: α, the shape parameter; k, the shortest possible job; and p, the upper bound of jobs. In the distribution, $k \leq x \leq p$. It follows that

$$\int_k^p f(x)dx = \int_k^p \alpha k^\alpha x^{-\alpha-1} dx = 1 - (k/p)^\alpha.$$

Thus, as in [13], the probability density function for the Bounded Pareto is defined as:

$$f(x) = \frac{1}{1-(k/p)^\alpha} \alpha k^\alpha x^{-\alpha-1} \qquad \alpha, k > 0, k \leq x \leq p.$$

Since α, k, and p are parameters of the Bounded Pareto distribution, for derivation simplicity, we define a function $\mathcal{K}(\alpha, k, p) = \frac{\alpha k^\alpha}{1-(k/p)^\alpha}$ in the following. The probability density function $f(x)$ is rewritten as:

$$f(x) = x^{-\alpha-1}\mathcal{K}(\alpha, k, p) \qquad \alpha, k > 0, k \leq x \leq p. \tag{2}$$

From (2), we have:

$$E[X] = \int_k^p f(x)xdx = \begin{cases} \frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha-1,k,p)} & \text{if} \quad \alpha \neq 1; \\ (\ln p - \ln k)\mathcal{K}(\alpha,k,p) & \text{if} \quad \alpha = 1. \end{cases} \quad (3)$$

$$E[X^2] = \int_k^p f(x)x^2dx = \frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha-2,k,p)}. \quad (4)$$

$$E[X^{-1}] = \int_k^p f(x)x^{-1}dx = \frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha+1,k,p)}. \quad (5)$$

According to Pollaczek-Khinchin formula [14], we have

Lemma 1. *Given an $M/G_P/1$ FCFS queue on a server, where the arrival process has rate λ and X denotes the Bounded Pareto service time density distribution. Let W be a job's queueing delay, and S be a job's slowdown. Then,*

$$E[S] = E[W] \cdot E[X^{-1}] = \frac{\lambda E[X^2]E[X^{-1}]}{2(1-\lambda E[X])}. \quad (6)$$

Note that the slowdown formula follows from the fact that W and X are independent from a FCFS queue.

2.2 Slowdown on Internet Servers

Based on the proportional-share resource scheduling mechanisms like GPS [14], PGPS [20], and Lottery Scheduling [22], we assume that the processing rate of an Internet server can be proportionally allocated to a number of *task servers*. Each task server i ($i = 1...N$) represents a processing unit that handles requests in a service class in a FCFS way. Let c_i denote the normalized processing rate of the task server i. Then,

$$\sum_{i=1}^{N} c_i = 1 \qquad 0 < c_i \leq 1 \text{ for } 1 \leq i \leq N. \quad (7)$$

Lemma 2. *Given an $M/G_P/1$ FCFS queue on a task server i with processing capacity c_i, X_i denotes the Bounded Pareto service time density distribution on the task server. Then,*

$$E[X_i] = \frac{1}{c_i}E[X]. \quad (8)$$

$$E[X_i^2] = \frac{1}{c_i^2}E[X^2]. \quad (9)$$

$$E[X_i^{-1}] = c_i E[X^{-1}]. \quad (10)$$

Proof. On the task server i, the lower bound and upper bound for the Bounded Pareto distribution is k/c_i and p/c_i, respectively. According to (1), we have

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

Thus, on the task server i, we define the probability density function for the Bounded Pareto as:

$$f(x) = \frac{\alpha k^\alpha}{c_i^\alpha(1-(k/p)^\alpha)}x^{-\alpha-1}, \alpha, k > 0, \frac{k}{c_i} \leq x \leq \frac{p}{c_i}.$$

Recall the definition of $\mathcal{K}(\alpha,k,p) = \frac{\alpha k^\alpha}{1-(k/p)^\alpha}$, the probability density function for the Bounded Pareto on the task server i is rewritten as:

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

We have:

$$E[X_i] = \int_{k/c_i}^{p/c_i} f(x)dx = \begin{cases} \frac{1}{c_i}\frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha-1,k,p)} & \text{if} \quad \alpha \neq 1; \\ \frac{1}{c_i}(\ln p - \ln k)\mathcal{K}(\alpha,k,p) & \text{if} \quad \alpha = 1. \end{cases} \quad (11)$$

$$E[X_i^2] = \int_{k/c_i}^{p/c_i} f(x)x^2dx = \frac{1}{c_i^2}\frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha-2,k,p)}. \quad (12)$$

$$E[X_i^{-1}] = \int_{k/c_i}^{p/c_i} f(x)dx = c_i\frac{\mathcal{K}(\alpha,k,p)}{\mathcal{K}(\alpha+1,k,p)}. \quad (13)$$

From (3) and (11), we have $E[X_i] = \frac{1}{c_i}E[X]$. From (4) and (12), we have $E[X_i^2]\frac{1}{c_i^2}E[X^2]$. From (5) and (13), we have $E[X_i^{-1}] = c_i E[X^{-1}]$. □

According to Lemma 1 and 2, we have

Theorem 1. *Given an $M/G_P/1$ FCFS queue on a task server i with processing capacity c_i, where λ_i denotes the arrival rate and X_i denotes the Bounded Pareto service time density distribution on the task server. Let W_i be a job's queueing delay and S_i be a job's slowdown on the task server. Then,*

$$E[S_i]\frac{\lambda_i E[X^2]E[X^{-1}]}{2(c_i - \lambda_i E[X])}. \quad (14)$$

Note that the $M/G_P/1$ queue is reduced to an $M/D/1$ queue when requests' service times are equal to a constant b. This kind of queueing model is interesting in a session-based E-commerce workload model. 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. They can be modeled as an $M/D/1$ queue. In the $M/D/1$ FCFS system, (14) is reduced to

$$E[S_i] = \frac{\lambda_i b}{2(c_i - \lambda_i b)}. \quad (15)$$

3 Processing Rate Allocation for Proportional Slowdown Differentiation

A proportional differentiation model ensures the quality spacing between class i and class j to be proportional to certain pre-specified differentiation parameters δ_i and δ_j [7]; that is,

$$\frac{q_i}{q_j} = \frac{\delta_i}{\delta_j} \qquad 1 \le i, j \le N,$$

where q_i and q_j are the QoS factor of class i and class j, respectively. So it is up to applications and clients to select appropriate QoS levels in terms of differentiation parameters that best meets their requirements, cost, and constraints.

The PSD model aims to control the ratios of the average class slowdown based on the differentiation parameters $\{\delta_i, i = 1, \ldots, N\}$. Specifically, the PSD model requires that the ratio of average slowdown between class i and j is fixed to the ratio of the corresponding differentiation parameters

$$\frac{E[S_i]}{E[S_j]} = \frac{\delta_i}{\delta_j} \qquad 1 \le i, j \le N. \tag{16}$$

The differentiation predictability property requires that higher classes receive better service, i.e., lower slowdowns. Without loss of generality, we assume that class 1 is the 'highest class' and set $0 < \delta_1 < \delta_2 < \ldots < \delta_N$.

For feasible rate allocation, we must ensure that the system utilization $\sum_{i=1}^{N} \lambda_i E[X] \le 1$. That is, the total processing requirement of the N classes of traffic is less than the server processing capacity.

According to Theorem 1, the set of (16), in combination with (7), lead to

$$c_i = \frac{\lambda_i}{\delta_i} \frac{1 - \sum_{i=1}^{N} \lambda_i E[X]}{\sum_{i=1}^{N} \frac{\lambda_i}{\delta_i}} + \lambda_i E[X]. \tag{17}$$

From this equation, we can observe that the remaining capacity of the server is fairly allocated to different classes according to their scaled arrival rates with respect to their differentiation parameters.

It follows that the expected slowdown of class i, $E[S_i]$, is calculated as:

$$E[S_i] = \frac{\delta_i E[X^2] E[X^{-1}] \sum_{i=1}^{N} \frac{\lambda_i}{\delta_i}}{2(1 - E[X] \sum_{i=1}^{N} \lambda_i)}. \tag{18}$$

From (18), we have the following three basic properties regarding the predictability and controllability of the proportional slowdown differentiation given by the allocation strategy:

1. Slowdown of a request class increases with its request arrival rate.

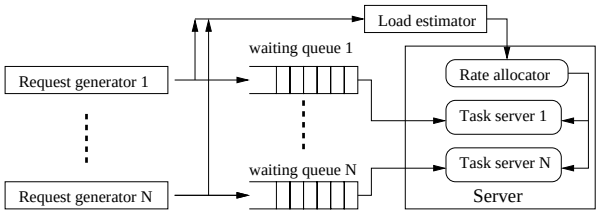

Figure 1. The simulation model's structure.

2. With the increase of the differentiation parameter of a request class, its slowdown increases but all other request classes have lower slowdowns.

3. Increasing the workload (request arrival rate) of a higher request class causes a larger increase in slowdown of a request class than increasing the workload of a lower request class.

4 Performance Evaluation

In order to evaluate the performance of the proposed processing rate allocation strategy for PSD provisioning under an $M/G_P/1$ traffic model, we conducted the simulations with requests generated by using GNU scientific library [12]. In this section, after introducing the simulation model, we first illustrate the effectiveness of the rate-allocation PSD strategy by comparing the achieved slowdowns with those calculated from the PSD model. We then show the differentiation predictability and controllability of the proposed rate-allocation strategy. Finally, we discuss the influence of the shape parameter α and the upper bound p of the Bounded Pareto distribution on PSD provisioning.

4.1 Simulation Model

We built a simulation model which consisted of a number of request generators, waiting queues, a load estimator, a processing rate allocator, and a number of task servers. Figure 1 outlines the basic structure of the simulation model.

The request generators produced requests with appropriate inter-arrival distribution and size distribution. In the simulation model, we generated the Bounded Pareto service time distribution by modifying GNU scientific library. In the $M/G_P/1$ traffic model, the request sizes of each class followed a Bounded Pareto distribution. Note that the number of classes in PSD provisioning on the server is usually rather limited. It varied from 2 to 3 in many similar experiments for PDD provisioning in packet scheduling [8, 9, 11, 17]. Each request was sent to the server and stored in a waiting queue according to its class type. Requests from the same class were processed by a task server in a FCFS manner.

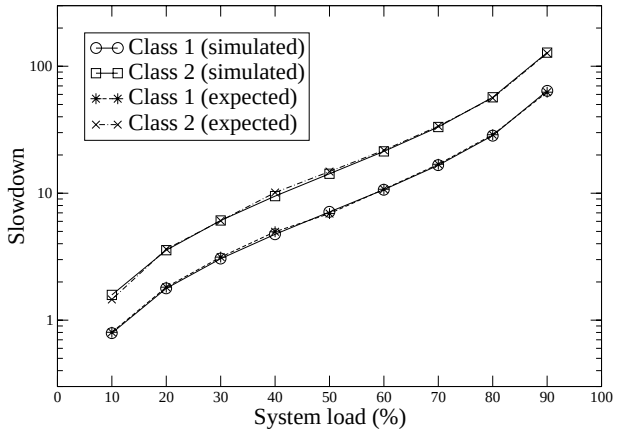

Figure 2. Simulated and expected slowdowns of two classes ($\delta_1 : \delta_2 = 1 : 2$).

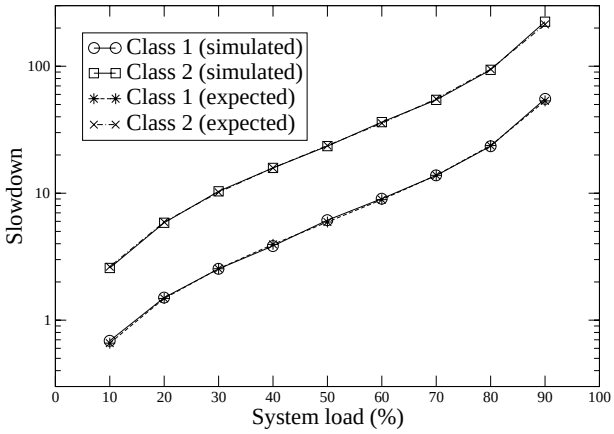

Figure 3. Simulated and expected slowdowns of two classes ($\delta_1 : \delta_2 = 1 : 4$).

The load estimator measured the arrival rate and the incurred load for every class. In the simulation, the load was estimated for every thousand time units. A time unit was set equal to the processing time of an average-size request. We estimated a class's load based on its history; that is, the load for next thousand time units was the average load in past five thousand time units. The rate allocator performed the proposed rate-allocation strategy according to (17) for every class. Meanwhile, the processing rate was reallocated for every thousand time units.

Simulation parameters were set as follows. The shape parameter (α) of the Bounded Pareto distribution was set equal to 1.5, as suggested in [9]. As indicated in [13], its lower bound and upper bound were set equal to 0.1 and 100, respectively. We also did experiments for larger upper bound settings to evaluate the influence. We assumed that all classes had the same load. In the experiments, the simulator first warmed up for 10,000 time units. The slowdown of a class was then measured for every thousand time units. After 60,000 time units, we calculated the slowdowns of classes. Each reported result is an average of 100 runs.

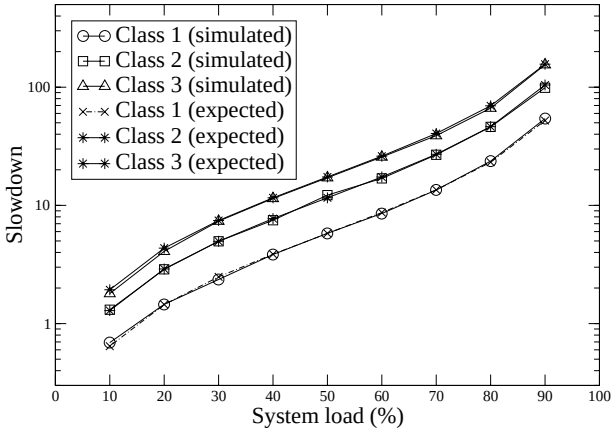

Figure 4. Simulated and expected slowdowns of three classes ($\delta_1 : \delta_2 : \delta_3 = 1 : 2 : 3$).

4.2 Effectiveness of the Rate-allocation Strategy

In this section, we show the effectiveness of the proposed rate-allocation strategy by comparing the simulated slowdowns with the expected values calculated by (18) under various load conditions. Figure 2 shows the result of two classes. Their differentiation parameters (δ_1, δ_2) are (1, 2). To show the results in a comparable way, the logarithmic y-axis is used. The figure shows very small differences between the simulated and expected slowdowns under various load conditions. It also shows the achieved system slowdowns, which are the weighted slowdowns of the two classes, are also very close to the expected ones. We then change the differentiation parameters of the two classes (δ_1, δ_2) to (1, 4). The results are shown in Figure 3. We also show the results of the experiment with three classes in Figure 4, where the differentiation parameters ($\delta_1, \delta_2, \delta_3$) are (1, 2, 3). From all these figures, we can observe that the proposed rate-allocation strategy is effective in achieving expected slowdowns under various load conditions. This provides the base for the following discussions on the properties of the PSD rate-allocation strategy.

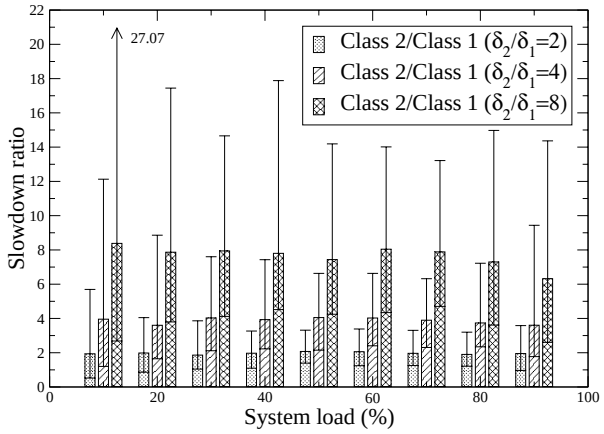

Figure 5. Percentiles of simulated slowdown ratios for two classes.

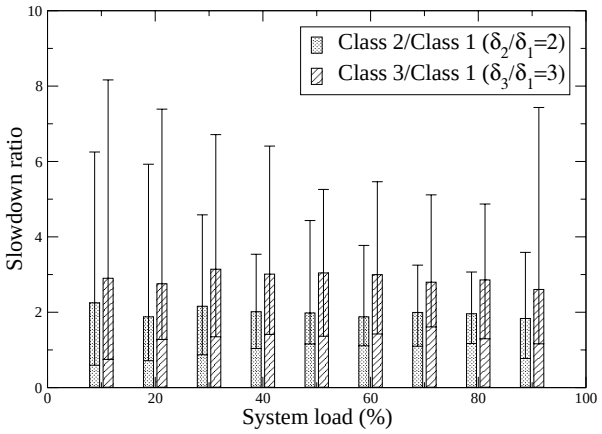

Figure 6. Percentiles of simulated slowdown ratios for three classes.

4.3 Differentiation Predictability of the Rate-allocation Strategy

In this section, we show experiment results to demonstrate the differentiation predictability of the proposed rate-allocation strategy. Recall that the predictability property means the slowdown of a higher class is proportionally smaller than that of a lower class under various system load conditions in various timescales.

We first show the results in long timescales. Figure 5 and 6 shows the results of different parameter settings for two and three classes, respectively. In both figures, the upper line is the 95th percentile; the bar is the 50th percentile; and the lower line is the 5th percentile. When the percentile is too large, we give its value in the figure directly. Both figures show that under various system load conditions, the proposed rate-allocation strategy can guarantee that the achieved ratios of the average slowdown are close to the corresponding pre-specified differentiation parameter ratios. It means that a higher class has proportionally smaller average slowdown than a lower class in long timescales, although there exist some variances. From these results, we find that the proposed rate-allocation strategy can achieve the objective of providing PSD services to different classes with different differentiation parameters under various system load conditions in long timescales.

From the long-timescale results shown in the figures, we can also observe that the achieved slowdown ratios are not distributed equally around the 50th percentile. For example, Figure 5 shows when the pre-specified differentiation ratio of the two classes (δ_2/δ_1) is 4 and system load is 10%, the 95th percentile of the experienced slowdown ratio is about 12 while that of the 5th percentile is 1.2. We believe this behavior is caused by the heavy-tail property of the Bounded Pareto distribution. More experiments will be conducted in our future work to further investigate this behavior.

From Figure 5, we unexpectedly find when the pre-specified differentiation ratio is small (i.e., $\delta_2/\delta_1 = 2$), the experienced slowdown of a higher class (class 1) can be larger than that of a lower class (class 2). Such behavior can be obviously observed when system load is 10%. That is, the ratio of experienced slowdowns of Class 2 to Class 1 is lower than 1 at the 5th percentile, while the pre-specified differentiation ratio is 2.

In order to demonstrate the differentiation predictability due to the proposed rate-allocation strategy in short timescales, we show the slowdowns of individual requests in Figures 7 and 8. From Figure 7, we can observe that the slowdown difference between class 1 and class 2 is small in moderate-load conditions. Some requests from class 1 have large slowdowns while some from class 2 have large slowdowns, although the pre-specified slowdown ratio of class 2 to class 1 is 2. Figure 8 shows the results in heavy-load conditions. We can observe that requests from class 1 experienced larger slowdowns than those from class 2. This behavior contradicts their pre-specified differentiation 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. We find that sometimes the behavior of individual requests is consistent with their slowdown parameters, and sometimes not. They suggest that the proposed rate-allocation strategy can only provide weak predictability in short timescales. Actually, the strategy only determines the processing rate allocated to one class periodically. In other words, it acts according to the macro-behavior (class load) of a class rather than its micro-behavior, such as experienced slowdowns of individual requests. Improving the performance of the rate-

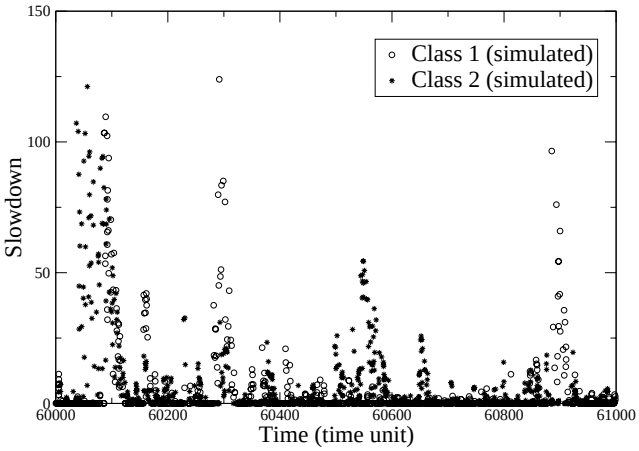

Figure 7. Slowdown of individual requests when the system load is 50%.

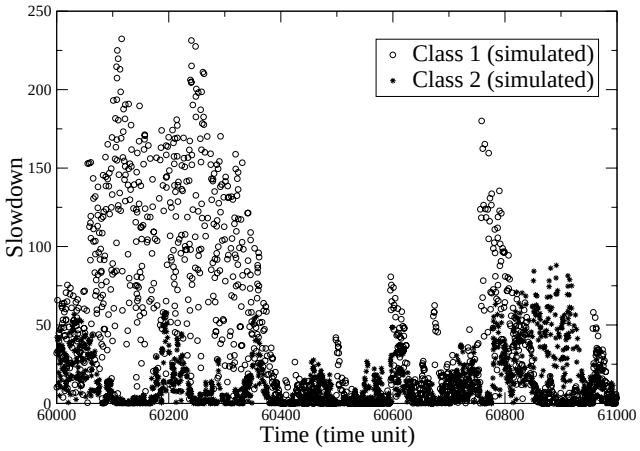

Figure 8. Slowdown of individual requests when the system load is 90%.

allocation strategy in achieving differentiation predictability in short timescales will be part of our future work.

4.4 Differentiation Controllability of the Rate-allocation Algorithm

In this section, we show the differentiation controllability due to the proposed rate-allocation strategy. Figure 9 shows the achieved slowdown ratios of two classes with different differentiation parameter settings. It can be seen that when the pre-specified differentiation parameter ratios are small (i.e., δ_2/δ_1=2 and 4), the rate-allocation strategy can accurately achieve the corresponding slowdown ratios at various load conditions. As the pre-specified differentiation parameter ratio increases (δ_2/δ_1=8), the difference between the achieved slowdown ratios and pre-specified ones become large at various load conditions. Such behavior, we believe, is caused by the load estimation error.

From (17), we can see that the processing rate allocated to a class is determined by its class load, its differentiation parameter, and system load. Therefore, it is important for the rate-allocation strategy to accurately estimate a class's load. Meanwhile, it is necessary to do such estimation in short timescales, such as one thousand time units used in our experiments, so as to govern the adaptiveness and short-timescale predictability of the PSD provisioning. Such estimation, however, is difficult because of the burstiness of the Bounded Pareto distribution and the variance of Poisson distribution. According to (17), such estimation error has larger influence on the achieved slowdown ratio with the increase of the differentiation parameter.

Figure 10 depicts the simulated slowdown ratios for system with three classes. In comparison with Figure 9, we can see that the variance of these ratios is larger than those

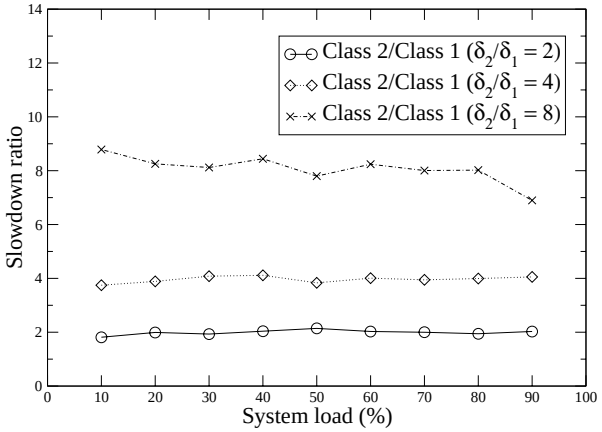

Figure 9. Simulated slowdown ratios of two classes.

in system with two classes. This behavior is also caused by the estimation error. When the load of one class is estimated inaccurately, it affects not only the processing rate allocated to this class, but other classes as well. Therefore, the situation may become worse as the number of classes to be differentiated increases.

Although there exist some variances for the simulated slowdown ratios under various system conditions, from both figures, we can observe that the rate-allocation strategy can achieve the pre-specified differentiation ratios. They illustrate that the strategy can control the slowdown ratios between classes adaptively.

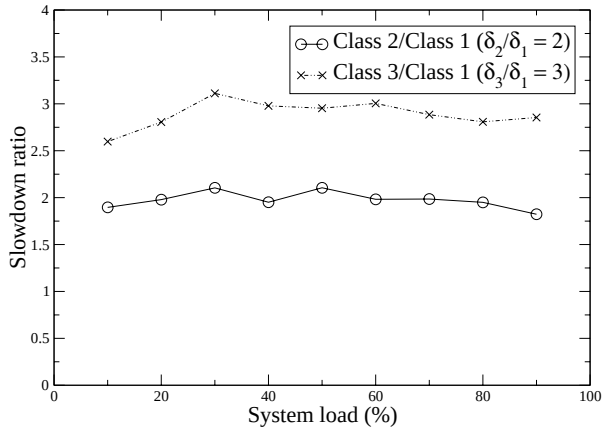

Figure 10. Simulated slowdown ratios of three classes.

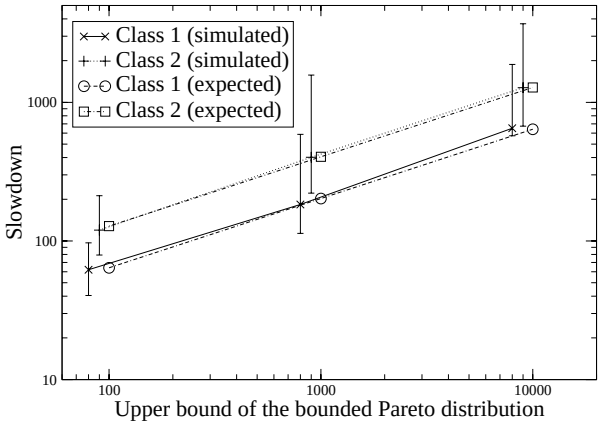

Figure 12. Influence of the upper bound of the Bounded Pareto distribution.

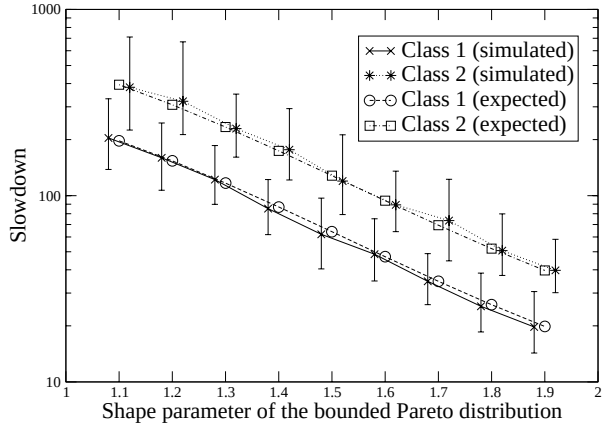

Figure 11. Influence of the shape parameter of the Bounded Pareto distribution.

4.5 Influence of the Shape Parameter and Upper Bound

In this part, we examine the influence of the shape parameter α and the upper bound p of the Bounded Pareto distribution on the performance of the proposed rate-allocation 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 11 shows the experienced slowdowns of two classes ($\delta_2/\delta_1 = 2$) due to various shape parameter settings ($1.0 \leq \alpha \leq 2.0$). The first observation is that the shape parameter has little influence on the differentiation predictability due to the proposed rate-allocation strategy. Both classes can receive the differentiated slowdowns as expected. The differences between the experienced slowdowns and expected ones are not dependent on the shape parameter. This behavior is 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, for the given lower and upper bounds, the smaller the shape parameter α, the burstier the generated traffic [16]. A request may experience larger queueing delay than that from a "smooth" traffic. Formally, by (4), (5), we know that, when the shape parameter decreases, its second moment $E[X^2]$ increases, its $E[X^{-1}]$ decreases, and its expected slowdown also increases.

The upper bound of the Bounded Pareto distribution (p) also affects the experienced slowdown of the $M/G_P/1$ traffic. We show the results in Figure 12. The ratio of the differentiation parameter of two classes (δ_2/δ_1) is 2. It can be seen that the upper bound has little influence on the differentiation predictability due to the rate-allocation strategy. The differences between the experienced slowdowns and expected ones are not dependent on 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 increases. By (4), (5), as the upper bound increases, the second moment of the traffic $E[X^2]$ increases and $E[X^{-1}]$ 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.

5 Related Work

The DiffServ provisioning problem was first addressed in the network core. The proportional differentiation model [7] has been accepted as an important DiffServ model and been applied in the PDD model in packet scheduling [8]. Many algorithms have been designed to achieve the PDD provisioning in the network routers. They can be classified into two categories: rate-based; see BPR [8] and JoBS [18] for examples, time-dependent priority based; see WTP, PAD, and HPD [9], adaptive WTP [11, 17], MDP [19], and VirtualLength [23] for examples. Servers play an important role in end-to-end DiffServ provisioning. Those algorithms can be tailored for request scheduling for PDD provisioning in the server side [15]. However, the algorithms are not applicable to PSD provisioning in the server side because slowdown is not only dependent on a job's queueing delay but also on its service time, which varies significantly depending on the requested services.

In the server side, a primary focus of DiffServ provisioning has been on priority-based request scheduling for responsive time differentiation [2, 4, 6, 10]. For example, in [2], the authors addressed strict priority scheduling strategies for controlling CPU utilization in Web content hosting servers. QoS was introduced by assigning priorities to requests for different contents. Requests of lower priority classes were only executed if no requests existed in any higher priority classes. 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. Therefore, this kind of priority-based scheduling schemes cannot achieve PSD provisioning.

Admission control is often used in combination with priority-based scheduling for DiffServ provisioning. For example, in [1], the authors used classical feedback control theory to achieve overload protection, performance guarantees, and service differentiation in Web servers. The strategy was based on real-time scheduling theory which states that response time can be guaranteed if server utilization is maintained below a pre-computed bound. Thus, control-theory approaches, in combination with content adaptation strategies, were formulated to keep server utilization at or below the bound. In [15], the authors proposed admission control algorithms in combination with time-dependent priority scheduling for proportional queueing-delay differentiation on a Web server. Therefore, this kind of admission control itself is not sufficient in PDD provisioning and is not applicable to PSD provisioning.

Stretch factor, a variant of slowdown, was adopted in [25] as the performance metric for DiffServ provisioning in a cluster of Internet servers. The authors proposed a demand-driven DiffServ strategy by adopting an $M/M/1$ queueing model to guide node-based resource allocation optimization. They implicitly applied processor sharing scheduling strategy for the modeling of stretch factor. However, in a single queue, a realistic scheduling strategy is FCFS. We note that for an $M/M/1$ FCFS queue with the unbounded exponential service time density distribution X, there is no valid stretch factor or slowdown because $E[X^{-1}]$ is not existent. For an $M/M/1$ FCFS queue with a bounded exponential service time density distribution X, there is no closed form expression for the stretch factor or slowdown because $E[X^{-1}]$ only has a definite value when the lower bound and the upper bound of service time are given. 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, 13]. In this paper, we investigate the problem of processing rate allocation for PSD provisioning under a popular heavy-tailed traffic pattern (Bounded Pareto).

In [13], Harchol used slowdown as a primary performance metric in evaluating task assignment strategies in a distributed server system, where the workload was heavy-tailed (Bounded Pareto) and job size was unknown to the scheduler. The primary objective was to minimize mean slowdown of all job classes in the distributed system. In this paper, we give a closed analytic form of expected slowdown in an $M/G_P/1$ FCFS queue for processing rate allocation on servers for PSD provisioning among different job classes. Our work is complementary to the previous work.

6 Conclusion

Slowdown is an important performance metric on Internet servers because it takes into account both the delay and service time of a request simultaneously. Although proportional delay differentiation has been studied extensively in the literatures, there are few research work models for PSD provisioning in the server side. 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 $M/G/1$ FCFS queues with Bounded Pareto service time distribution, referred to $M/G_P/1$ queues. We have deployed a task server for handling each request class in a FCFS way and presented the strategy of processing rate allocation for the task server in support of PSD provisioning. We have built a simulation model for the processing rate allocations. Simulation results have showed that the allocation strategies can achieve the objective of providing predictable, controllable and fair proportional slowdown differentiation on the servers. Our future work will be on improving the performance of the rate-allocation strategy in providing short-timescale differentiation predictability.

Acknowledgement

This research was supported in part by NSF grants CCR-9988266 and ACI-0203592.

References

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

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

[3] M. Arlitt, D. Krishnamurthy, and J. Rolia. Characterizing the scalability of a large Web-based shopping system. *ACM Trans. on Internet Technology*, 1(1):44–69, 2001.

[4] N. Bhatti and R. Friedrich. Web server support for tiered services. *IEEE Network*, 13(5):64–71, 1999.

[5] S. Blake, D. Black, M. Carlson, E. Davies, W. Z., and W. Weiss. An architecture for differentiated services. *IETF RFC 2475*, 1998.

[6] X. Chen and P. Mohapatra. Performance evaluation of service differentiating internet servers. *IEEE Trans. on Computers*, 51(11):1,368–1,375, 2002.

[7] C. Dovrolis and P. Ramanathan. A case for relative differentiated services and the proportional differentiation model. *IEEE Network*, 13(5):26–34, 1999.

[8] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proc. ACM SIGCOMM*, 1999.

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

[10] L. Eggert and J. Heidemann. Application-level differentiated services for Web servers. *World Wide Web Journal*, 3(2):133–142, 1999.

[11] L. Essafi, G. Bolch, and A. Andres. An adaptive waiting time priority scheduler for the proportional differentiation model. In *Proc. of the High Performance Computing Symposium*, April 2001.

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

[13] M. Harchol-Balter. Task assignment with unknown duration. *Journal of ACM*, 29(2):260–288, 2002.

[14] L. Kleinrock. *Queueing Systems, Volume II*. John Wiley and Sons, 1976.

[15] S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau. Admission control and dynamic adaptation for a proportional-delay DiffServ-enabled Web server. In *Proc. ACM SIGMETRICS*, 2002.

[16] 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.

[17] M. K. H. Leung, J. C. S. Lui, and D. K. Y. Yau. Adaptive proportional delay differentiated services: Characterization and performance evaluation. *IEEE/ACM Trans. on Networking*, 9(6):908–817, 2001.

[18] J. Liebeherr and N. Christin. JoBS: Joint buffer management and scheduling for differentiated services. In *Proc. of the Int'l Workshop on Quality of Service (IWQoS)*, pages 404–418, June 2001.

[19] 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, April 2000.

[20] A. K. Parehk and R. G. Gallager. A generalized processor sharing approach to flow control in Integrated Sevices networks: the single-node case. *IEEE/ACM Trans. on Networking*, 1(3):344–357, 1993.

[21] A. Riska, E. Smirni, and G. Ciardo. ADAPTLOAD: effective balancing in clustered Web servers under transient load conditions. In *Proc. IEEE Int'l Conf. on Distributed Computing Systems (ICDCS)*, 2002.

[22] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: flexible proportional-share resource management. In *Proc. the 1st USENIX Symposium on Operating System Design and Implementation*, 1994.

[23] J. Wei, Q. Li, and C.-Z. Xu. Virtuallength: A new packet scheduling algorithm for proportional delay differentiation. In *Proc. of IEEE Int'l Conf. on Computer Communications and Network (ICCCN)*, 2003.

[24] X. Zhou, J. Wei, and C.-Z. Xu. Modeling and analysis of 2D service differentiation on E-commerce servers. In *Proc. of the 24th IEEE Int'l Conference on Distributed Computing Systems (ICDCS)*, March 2004.

[25] H. Zhu, H. Tang, and T. Yang. Demand-driven service differentiation for cluster-based network servers. In *Proc. IEEE INFOCOM*, pages 679–688, 2001.