EURECOM
Department of X
2229, route des Crêtes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-11-251

Time-dependent Priority Disciplines: A Heavy Traffic Analysis

March 2th, 2011
Last update March 2th, 2011

Damiano Carra, Pietro Michiardi

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : michiard@eurecom.fr

[1]EURECOM's research is partially supported by its industrial members: BMW Group, Cisco, Monaco Telecom, Orange, SAP, SFR, Sharp, STEricsson, Swisscom, Symantec, Thales.

Time-dependent Priority Disciplines: A Heavy Traffic Analysis

Damiano Carra, Pietro Michiardi

Abstract

In this work, we revisit the theory underlying a range of time-dependent priority disciplines, and extend it to include the requirements of a class of applications that has not been studied in the past. Specifically, we target applications and services in which a scarce resource, or a fraction thereof, has to be awarded to a large number of concurrent requests. We thus consider an *heavy traffic* regime, and derive closed form expressions to evaluate the relative difference – in terms of average waiting times – of requests that belong to different priority classes.

Our analysis allows to derive the expected waiting time of requests in single server queueing systems, under the assumption of *finite buffers* for both non-preemptive and preemptive cases. Additionally, we extend our results to a distributed scenario composed of an interconnected set of a large number of unreliable single server queueing systems.

We then describe a number of applications and illustrate how to use our theoretic foundation to analyze and tune their performance. The accuracy of our analysis is validated through a set of experiments that we performed in a synthetic setting.

Index Terms

Dynamic Priority, Finite Buffer, Closed Systems, Fundamental Results

Contents

List of Figures

1 Introduction

Service differentiation in the Internet has been a fertile area of research and commercial activity in the past two decades (see for instance [1, 2] and the references therein). The quest to design scalable resource management and admission control mechanisms [3–6] to handle Internet traffic has contributed to a large number of approaches that have been extensively studied and debated [7–9]. Motivated by the sensitivity of absolute differentiation schemes to small variations in the load distribution of the traffic to differentiate, a more recent approach to service differentiation has considered the concept of *relative differentiation* [10]. Network traffic is divided into priority classes that are ordered such that class i is better (or at least no worse) than class $i-1$ in terms of delay or packet loss.

In this work, we revisit the theory underlying a range of priority disciplines that achieve differentiation and extend it to include the requirements of a class of applications that go beyond the transport of network traffic. Specifically, we target applications and services in which a scarce resource, or a fraction thereof, has to be awarded to a large number of concurrent requests. The ability to predict the performance of applications deployed in scenarios ranging from desktop environments to large scale distributed systems, is of crucial importance, especially when the resources that come into play are scarce and the demand to access them is high. We thus consider an *heavy traffic* regime and derive closed form expressions to evaluate the relative performance of customer requests.

In the following, we study a single server queuing system that services customers from different priority classes. Our focus is on the analysis of queuing disciplines in which the priority of a customer is dynamic and determined by the amount of time it has waited in the queue. First, we analyze *proportional differentiation* priority disciplines and we derive simple laws that describe the average waiting time of customer requests in different priority queues. Armed with the realization that the distribution of the waiting times is characterized by heavy tails that penalize low priority requests, we study an *additive differentiation* priority discipline that mitigates this effect, while conserving the desirable property of a simple formulation to establish the relative performance of requests. Our analysis allows to derive the expected waiting time of requests, under the assumption of *finite* buffers for both *non-preemptive* and *preemptive* cases. Additionally, we extend our results to a distributed scenario composed of an interconnected set of a large number of *unreliable* single server queueing systems.

There are many practical situations in which the ability to control the priority of customers through delay-dependent disciplines is important. In this work we survey peer-to-peer file-sharing and operating systems scheduling disciplines. Our work helps in understanding how the additional degrees of freedom provided by delay-dependent disciplines can be used to tune the performance of the different groups of requests. In particular, we show where our analytic results can be used to build relative differentiated services architectures that are *predictable* and *controllable* [10].

In summary, the contributions of this work are the following:

- We provide a comprehensive theoretic foundation for the analysis of a range of time-dependent priority disciplines that operate in an heavy traffic regime, covering both non-preemptive and preemptive scheduling approaches. Additionally, we provide the first analytic tractation of an additive differentiation scheme in heavy traffic, and show that it is not liable to the problem of mistreatment of low priority customers;

- We define models of both centralized and distributed applications and services, and study their accuracy through experiments performed in a synthetic environment. Our results indicate a very good match between analytical and experimental results: as such, our tools allow system designers to focus on the configuration of a handful set of parameters, and enable them to predict the relative performance achieved by customer requests that compete to access the services of a variety of applications.

- We illustrate a broad range of applications of time-dependent priority disciplines, going beyond IP networking, including P2P file-sharing applications and OS scheduling. We show how our analysis can be helpful in understanding how to operate and tune the priority mechanisms implemented in such applications.

The remainder of the paper is organized as follows. In Sec. 2 we overview the literature of time-based priority disciplines and position our work with respect to prior art in Internet traffic differentiation. Sec. 3 and 4 constitute the body of our work, where we detail our theoretic framework to study proportional and additive service differentiation schemes, both in a centralized and distributed settings. Sec. 5 is dedicated to the applications we cover in this work, which are studied through the lenses of our analytical framework. We validate the accuracy of our tools in Sec. 6 and conclude in Sec. 7.

2 Background and Related Work

We consider a $M/M/1/k+1$ queue, where jobs, which hereinafter we call *requests*, arrive according to a Poisson process, and their service times are exponentially distributed. In this work we examine a system composed of a single server queue and a buffer with k positions. The single server queue allows P different priority classes (or groups): requests for group i ($i = 1, 2, \ldots, P$) arrive according to independent Poisson processes with rate $p_i\lambda$, where λ is the total arrival rate and p_i is the probability that the requests belong to group i, with $\sum_i p_i = 1$.

The request processing time is exponentially distributed with parameter μ_i. We define:

$$\frac{1}{\mu} = \sum_{i=1}^{P} \frac{p_i}{\mu_i}, \qquad \rho_i = \frac{p_i \lambda}{\mu_i}, \qquad \rho = \sum_{i=1}^{P} \rho_i = \frac{\lambda}{\mu}$$

$$\text{and} \qquad W_0 = \sum_{i=1}^{P} \frac{\rho_i}{\mu_i},$$

where W_0 is the expected completion time for the request (job) in service.

Differently from the usual convention, we assume that a request i has priority over another request j if its priority value is bigger than the priority value of request j.

We consider three different types of queues: (i) infinite buffer ($k \to \infty$), (ii) finite buffer ($k < \infty$) and (iii) *closed* systems. In closed systems, the number of requests inside the system is constant (equal to $k+1$), and a new request is accepted only when the request in service leaves the system. In this case the request arrival rate equals the service rate, *i.e.*, $\lambda = \mu$.

We focus on applications that operate in a *heavy traffic* regime, i.e., the offered load approaches the service rate. In case of infinite buffer, the heavy traffic regime translates into $\lambda \to \mu$. In case of finite buffer, we consider as offered load the traffic that is accepted by the system. It is clear that the finite buffer case under heavy traffic tends to the closed system case, which is by definition in the heavy traffic regime.

Finally, we assume that requests do not leave the system until they are served. The literature is rich of studies that consider $M/M/1$ or $M/G/1$ single server queues that execute a variety of priority queueing disciplines [11–14]. In the remainder of this Section we will briefly review such theoretic results. However, prior works mainly focus on systems with an infinite buffer size. Instead, in this work we are interested in studying applications under the more realistic assumption which accounts for a limited buffer and we develop the appropriate tools to analyze such applications in Section 3.

2.1 Fixed Priority

The simplest priority discipline assumes that the priority of each group of requests is fixed and defined a-priori. Starting from [11], this type of system has been extensively treated in the literature. The mean waiting time of a class p request in the queue, denoted by W_p, when the buffer is infinite, can be expressed as [11]:

$$W_p = \frac{W_0}{\left(1 - \sum_{i=p}^{P} \rho_i\right)\left(1 - \sum_{i=p+1}^{P} \rho_i\right)} \tag{1}$$

A closer look at Eq. 1 indicates the main drawback of such a simple priority discipline. When the system operates in *heavy traffic* and the buffer is infinite, the requests that belong to the group with the lowest priority (i.e., $p = 1$) may suffer

from *starvation*, that results in an infinite waiting time for such customers. In fact, we have that $W_1 = W_0/(1-\rho)(1-\rho+\rho_1)$. When $\rho \to 1$, then we have that $W_1 \to \infty$.

Instead, all the other requests that belong to higher priority groups ($1 < p \leq P$) are characterized by a *finite* waiting time.

The fixed priority discipline has another limitation: for a generic group p the mean waiting time of requests that belong to this group depends on the traffic composition, i.e. the values of ρ_i, $\forall i \in P$. As such, the mean waiting time cannot be tuned to provide a differentiated service to priority groups, which is a desirable property required for a wide range of applications.

The limitations of the simple fixed priority discipline have been overcome by the *time-dependent* priority disciplines. With such disciplines, either all or none groups have an infinite average waiting time. Moreover, the disciplines come with a set of "knobs" which allow to differentiate the average waiting time among groups.

2.2 Time-Dependent Priority

When a time-dependent discipline is used in a single server queueing system, the priority of a requests depends both (i) on the specific group it belongs to and (ii) on the amount of time spent by such requests in the system. As such, these schemes have the desirable property that request *starvation* is not present (if $\rho < 1$): indeed, as the time progresses, the priority of a request grows, and it is eventually served by the system. The single server queue executes a simple scheduling process that selects the next request to be served based solely on its instantaneous priority.

Let T_{arrival} be the arrival time of a request and let T_{leave} be the time when the request leaves the system. We consider a class of priority schemes in which the priority $q_i(t)$ at time t assigned to a request belonging to group i is given by the following general expression:

$$q_i(t) = b_i(t - T_{\text{arrival}})^r - a_i, \tag{2}$$

with $T_{\text{arrival}} \leq t \leq T_{\text{leave}}$. Each priority group can be identified by the coefficients b_i and a_i, with $i = 1, 2, \ldots, P$, $0 < b_1 \leq b_2 \leq \ldots \leq b_P$ and $a_1 \geq a_2 \geq \ldots \geq a_P \geq 0$.

In Figure 1 we show two different cases. On the left hand side we show the case where $a_i = 0$ and $r > 1$: the priority over time of the requests follows a convex function. In case of $r < 1$ we have a similar behaviour, with concave functions. We define this type of priority as *proportional scheme*.

On the right hand side we show the case where $b_i = b$, $\forall i \in P$ and $r = 1$. The difference in terms of priority between two requests remains constant over time. We define this type of priority as *additive scheme*.

In the next sections we show the results for system in heavy traffic for the proportional and additive schemes, with finite buffers, and for services with or without preemption.

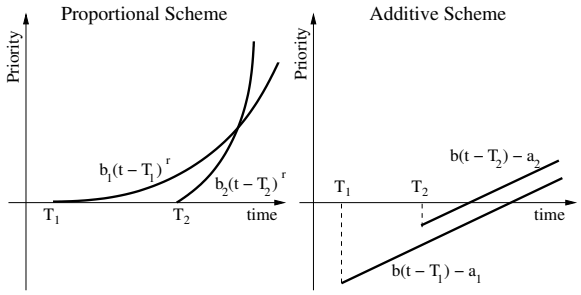

Figure 1: Examples of the evolution of the priority in case of proportional and additive schemes.

2.3 Related Work

Time-dependent priority disciplines have been studied originally in [12] for the linear case, and in [15] [13] and [14] for more general cases. None of such works consider the heavy traffic case, a finite buffer and closed systems, as we do in this work. Only [16] considered the heavy traffic regime for the linear time-dependent priority scheme (i.e., $r = 1$) and infinite buffer, so our results for the proportional scheme represent a generalization of the results in [16].

The heavy traffic regime for the linear time-dependent priority (i.e., $r = 1$), and some of the properties related to the proportional scheme, has been also studied within the *Proportional Delay Differentiation* (PDD) framework [10, 17]. In [10] the authors consider the specific case with $\mu_i = \mu$, $\forall i \in P$, while [17] derives the conditions under which the proportional delay differentiation is feasible. As previously pointed out, we consider the general case with any value of $r > 0$ and different μ_i.

Also the authors in [10] study the properties of the additive scheme under heavy traffic, in the specific case with $b_i = b = 1$: however, they do so using a simulation-based approach. Instead, we consider the general additive scheme with $b_i = b$ and we provide analytical results of its properties in heavy traffic.

Finally, all the above works consider systems and applications with *no preemption*, *i.e.*, a single server queue in which, once a request has been scheduled, it will be served before any other request will be considered for scheduling. In contrast, we provide results also for the pre-emptive work conserving case.

3 Time-dependent disciplines

In this Section we analyze in detail a range of time-dependent priority disciplines under the single server queue model. Building upon the formulation described in Section 2, we consider both a service *with* and *without* preemption and for a generic buffer size k.

In summary, our analysis indicates that it is possible to derive a simple expression to characterize the average waiting time for requests that arrive at the single

server queue. Our finding shows that there exists invariants that depend on the coefficients of the scheme used (proportional or additive), but not on the traffic composition.

As such, our results indicate a simple method to tune the relative performance of each priority class, which consists in setting appropriately the coefficients of each priority group.

Furthermore, we show that the *proportional* priority discipline for service differentiation, that has been widely explored in the context of IP networking [10,17], is very sensitive to the absolute values of the coefficients attributed to each priority class. The distribution of the request waiting times exhibits *heavy tails* which imply that requests from low priority groups may require a very long time before they are served. These effects disappears in case of the *additive* scheme, which is also less sensitive to the parameters of each priority group.

In the following, we derive the mean waiting time for customer requests for the two types of priority disciplines we hinted above: the proportional and the additive schemes.

3.1 Proportional Scheme

We consider the time-dependent priority discipline defined in Eq. 2 and assume $a_i = 0, \forall i \in P$.

3.1.1 Service without Preemption

As described in Sec. 2, the scheduler of the single server queue selects the next request to serve by choosing the request with the highest instantaneous priority $q_i(t)$. Once such request has been scheduled, if the server operates in a *non-preemptive* way, the next request will be scheduled only once the current request has been fully served.

The following result holds for systems with both an infinite and a finite buffer, and for closed systems.

Theorem 1. *Given any two priority groups i and j, the mean waiting times W_i and W_j, in case of non pre-emptive service, in the heavy traffic regime, satisfies the following condition:*

$$\frac{W_i}{W_j} \rightarrow \left(\frac{b_j}{b_i}\right)^{1/r} \tag{3}$$

Proof. See Appendix A.1. □

In other words, Theorem 1 indicates that, independently from the traffic composition (*i.e.*, the values of ρ_i), a time-dependent priority discipline provides a *proportional differentiated service* that depends on r and the coefficients b_i and b_j. While this results has been already found in case of $r = 1$ and infinite buffer, Theorem 1 states that the result holds also in case of finite buffer and closed system, and for any r.

In the following Corollary, we show a simple way to compute the absolute values of the waiting times.

Corollary 1. *Under the same hypothesis of Theorem 1, the mean waiting times can be computed as:*

$$W_i = \begin{cases} \frac{1}{b_i}\frac{\rho}{1-\rho}W_0\frac{1}{\sum_{i=1}^{P}\frac{\rho_i}{{b_i}^{1/r}}} & \textit{infinite buffer} \\ \\ \frac{1}{b_i}\frac{k}{\mu}\frac{1}{\sum_{i=1}^{P}\frac{p_i}{{b_i}^{1/r}}} & \textit{closed system (buffer = } k\textit{)} \end{cases} \tag{4}$$

Proof. In case of infinite buffer, by using Kleinrock's conservation law, we have a relation between $\sum \rho_i W_i$ and W_0, which can be used to derive the result. For the closed system, the number of requests in the queue is constant (k) and equal to the sum of requests belonging to each group, which can be derived from W_i using Little's theorem. □

Note that, in case of infinite buffer, under heavy traffic regime $W_i \to \infty$; nevertheless, Theorem 1 still holds, i.e., the ratio of the waiting times of two different classes is constant. Corollary 1 is interesting because it shows the relation between the mean waiting times and the parameters of the system (k, μ, and b_i) that can be tuned by the system administrator. Actually, the absolute values of the waiting times (in case of infinite buffer) can be computed using the formulas provided in [12] [15]: nevertheless such formulas imply recursive computations and do not show the direct influence of the system parameters on the waiting times.

As previously said, in case of finite buffer and heavy traffic regime, the system tends to behave as the closed system, therefore the results of Corollary 1 can be used also for approximating the finite buffer case.

3.1.2 Service with Preemption

We now consider a single server queueing system in which the service to any request can be interrupted by a new request that, as time progresses, has gained a higher priority than the currently scheduled one. The interrupted request can be resumed if its priority is the highest one. The *preemptive* work conserving case for a system operating in the heavy traffic regime has been rarely considered in the literature: however, as we will show in Sec. 5, there exist an important class of applications that operate in the preemptive mode.

In the specific case of proportional scheme with pre-emption, we add a constraint, *i.e.*, $\mu_i = \mu$, $\forall i$. We note that the restrictive assumption of a unique service rate μ reflects a system in which the requests arriving from different priority classes concern the same set of "objects", and thus the service rate is the same, independently of class i.

Let T_i be the mean time spent in the single server queuing system by a request belonging to priority class i, *i.e.*, $T_i = E[T_{\text{leave}} - T_{\text{arrival}}]$. Clearly, we have that $T_i = W_i + 1/\mu_i$, where W_i is the mean waiting time for a request in the class i.

The following result holds for systems with both an infinite and a finite buffer, and for closed systems (in all cases with $\mu_i = \mu, \forall i$).

Theorem 2. *Given any two priority groups i and j, the mean times spent in the system T_i and T_j, in case of pre-emptive service, in the heavy traffic regime, satisfies the following condition:*

$$\frac{T_i}{T_j} \to \left(\frac{b_j}{b_i}\right)^{1/r} \tag{5}$$

Proof. See Appendix A.2. □

To the best of our knowledge, this results has been never found before, not even in the infinite buffer case. The following Corollary complements Theorem 2, and indicates a simple way to compute the absolute values of $T_i, \forall i$.

Corollary 2. *Under the same hypothesis of Theorem 2, the mean time spent in the system can be computed as:*

$$T_i = \begin{cases} \dfrac{1}{b_i}\dfrac{1}{1-\rho}\dfrac{1}{\mu \sum_{i=1}^{P} \frac{p_i}{{b_i}^{1/r}}} & \textit{infinite buffer} \\ \\ \dfrac{1}{b_i}\dfrac{k+1}{\mu}\dfrac{1}{\sum_{i=1}^{P} \frac{p_i}{{b_i}^{1/r}}} & \textit{closed system (buffer = } k\textit{)} \end{cases} \tag{6}$$

Proof. The mean number of requests inside the system is equal to the sum of requests belonging to each group, which can be derived using Little's theorem: for the infinite case, we use the result from the $M/M/1$ queue with FCFS discipline, while for the closed system case the number of requests inside the system is constant and equal to $k + 1$. □

For the finite buffer case, considerations similar to the ones provided in the non pre-emptive case hold.

3.2 Additive Scheme

We now study an alternative time-dependent priority discipline that can be obtained from Eq. 2 when we set the coefficients $b_i = b, \forall i \in P$ and $r = 1$.

For the additive scheme, we are able to find general results which is valid for both the non pre-emptive and the pre-emptive cases. The result holds for systems with both an infinite and a finite buffer, and for closed systems.

Theorem 3. *Given any two priority groups i and j, the mean waiting times W_i and W_j, for both the non pre-emptive and the pre-emptive cases, in the heavy traffic regime, satisfies the following condition:*

$$(W_i - W_j) \rightarrow \frac{a_i - a_j}{b} \tag{7}$$

Proof. See Appendix A.3. □

As for the proportional case, Theorem 3 provides a relation between the mean waiting times independently from the traffic composition (i.e., the values of ρ_i). The absolute values of W_i, $\forall i$, can be easily computed using the following Corollary.

Corollary 3. *Under the same hypothesis of Theorem 3, the mean waiting times can be computed as:*

$$W_i = \begin{cases} \dfrac{a_i}{b} + \dfrac{1}{1-\rho} W_0 - \dfrac{1}{b\rho} \displaystyle\sum_{i=1}^{P} \rho_i a_i & \textit{infinite buffer} \\ \\ \dfrac{a_i}{b} + \dfrac{k}{\mu} - \dfrac{1}{b} \displaystyle\sum_{i=1}^{P} p_i a_i & \textit{closed system} \end{cases} \tag{8}$$

Proof. The expressions can be obtained following through with almost identical arguments used in Corollary 1. □

As for the proportional case, with Corollary 3 we can easily evaluate the impact of the system parameters on the waiting times.

3.3 Distribution of the Waiting Times: Results From a Numerical Analysis

The theoretical results provided in the previous sections consider the mean waiting time. In this section we are interested in understanding some basic properties of the complete probability distributions of the request waiting times. Since it is hard to derive such distributions analytically, we take a numerical approach which is similar to that developed in [18]. Here we focus on the *non-preemptive* case only, assume a finite buffer of size $k = 5000$ and a *heavy traffic* regime, and set $\mu_i = \mu = 1s^{-1}$.

We compare the distribution obtained by three service disciplines: (i) the basic First Come First Serve (FCFS) discipline, (ii) the time-dependent proportional scheme (Sec. 3.1) and (iii) the time-dependent additive scheme (Sec. 3.2).

Specifically, for the proportional scheme we generate a large set of requests whose priority class is uniformly distributed in the interval $b_i \in \{1, 50\}$, with $r = 1$ and $a_i = 0 \, \forall i \in P$. Similarly, we evaluate the additive scheme for a set

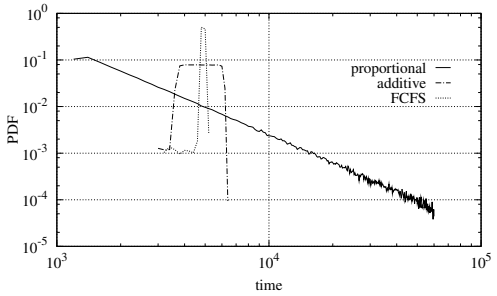

Figure 2: PDF of the waiting times for different priority schemes.

of requests whose priority class is identified by coefficients chosen uniformly at random in the set $a_i \in \{1, 2500\}$, with $r = 1$ and $b_i = 1 \, \forall i \in P$. The results of our experiments consist in the empirical probability density function (PDF) of the request waiting times in the system, and are depicted in Figure 2.

Our results indicate that, for the FCFS scheme, the PDF of the waiting times exhibit a peak around the mean waiting time, as expected. Figure 2 illustrates that, for the proportional case, the PDF exhibits *heavy tails*, a result also observed in [19]. We performed another experiment with $b_i \in \{1, 10\}$ to study the sensitivity of the proportional scheme to the range in which the coefficient b_i can take value: also in this case, the results (that we do not report here for the sake of clarity) show a PDF with heavy tails.

In [19], the authors consider also the additive scheme: they show that the additive scheme exhibits heavy tails if the coefficients a_i are selected from a probability distribution that has heavy tails. This means that, if the coefficients are bounded, i.e., $a_i < a_{max}$, $\forall i \in P$, the waiting time distribution does not have heavy tails, as our numerical results confirm (see Fig. 2). We note that the PDF is centered around the mean waiting time of the FCFS scheme, which is due to the uniformity of the distribution of the coefficients a_i, and has a support that is correlated to the difference between the maximum and minimum values of the coefficients a_i.

In summary, proportional and additive differentiation represent a powerful way to control the resources dedicated to the different priority classes, and thus their relative performance in terms of waiting times. However, a system based on the proportional scheme exhibit heavy tails in the distribution of the waiting times. Instead, the additive scheme, independently from the coefficients a_i, does not exhibit heavy tails.

In Section 5 we study applications that adopt the proportional and the additive scheme, and discuss the rationale of the particular choice of the different time-dependent priority disciplines.

4 Distributed Systems

In this Section we extend the single server queue model that we discussed in Sec. 2 and describe a *distributed system* involving a number of homogeneous (and possibly unreliable) entities, each modeled as a single server queue. In our model, each entity also acts as a customer that issues service requests to the other servers in the system. In summary, here we study the impact of system dynamics and show that variations of the system size, due to entities leaving and joining the system, can introduce a distortion that affect the relative performance differentiation of the priority classes.

The extended model outlined above, clearly targets a class of applications known as peer-to-peer (P2P) applications. In the following, our distributed system materializes as a P2P file-sharing application in which nodes implement a time-dependent priority discipline to award upload bandwidth resources, *i.e.*, nodes apply priority scheduling to the requests they receive for uploading the content they store. In Sec. 5 we study an instance of our model and delve into the details of the eMule [20] file-sharing application. There we show that eMule uses a proportional scheme for service differentiation, and we discuss the particular choice of the coefficients that govern its priority discipline.

Usually each server holds objects that may receive a large number of requests, while the available bandwidth resources are limited. Consequently, such applications work in the *heavy traffic* regime. Note that the model presented here is not limited to file-sharing applications, and it can be easily extended, for example, to distributed storage applications.

We now describe in detail our distributed system model and focus in particular on the unreliability of the servers in the system. In practice, our model accounts for what is generally known as node *churn*, i.e. the dynamic arrival and departure of nodes. Our goal is to study the impact of churn on the effective service rate and on the priority discipline. We suppose that when a node fails, the single server queue maintained by that node is flushed. As soon as the node come back online, the queue is initialized into an empty queue.

We now make a number of assumptions that can help in studying the effects of node churn. Each node in the system tries to download a particular content (that is, the node behaves as a customer) by sending a request to a *single* server node at a time. The client node remains online until the request has been fulfilled. If the server goes offline before the request has been served, the client node issues a request to another server uniformly picked at random from the available servers in the system that hold the desired content.

Let T_s be the session time[1] of the server nodes: T_s is a random variable with cumulative distribution function (CDF), that we label $F_{T_s}(t)$. We assume $F_{T_s}(t)$

[1] We talk about *session time* of a node i implicitly considering the session time as seen by another generic node j that contacts node i at a random instant. In the literature, this is usually referred to as the *residual* session time. The residual session time can be derived from the session time using Smith's formula [21].

to be known – see for instance the session time distribution in [22].

The time spent in the system by a request is also a random variable, but in this case, since we do not know its complete distribution, we consider its mean value, labeled as T_d, to be a deterministic value. Even if this represents a strong approximation (since the distribution may have heavy tails, as shown in Sec. 3.3), it is necessary for analytical tractability. In Sec. 6 we study the impact of this assumption with numerical simulations. The value of T_d is derived using the analysis presented in Sec. 3.

Now, assume the client to randomly choose a first server to issue the request to. The probability that the server session time is larger than the time necessary to serve the client request writes as:

$$\Pr[T_s > T_d] = 1 - F_{T_s}(T_d) = p.$$

In this case the total time a request spends in the system is exactly T_d.

Instead, the probability that the server session time is smaller than the time necessary to serve the client request is given by:

$$\Pr[T_s < T_d] = F_{T_s}(T_d) = 1 - p$$

In this case, the client node is required to find another server in the system and issue again a request to access the desired content. As such, the mean time that elapses for a client node that waits to be served is given by the conditional mean server session time, which writes as $\overline{T}_s^c = E[T_s|T_s < T_d]$. Hence, the total time the client request spends in the system, that we label T_d^{tot}, can be computed with the following expression:

$$\begin{aligned} T_d^{\text{tot}} &= pT_d + (1-p)p(T_d+\overline{T}_s^c) + (1-p)^2p(T_d+2\overline{T}_s^c) + ... \\ &= \sum_{i=0}^{\infty}(1-p)^i p(T_d + i\overline{T}_s^c) \\ &= T_d + \frac{1-p}{p}\overline{T}_s^c. \end{aligned} \tag{9}$$

The conditional mean server session time, $\overline{T}_s^c$, can be computed as follows. Let $F_{T_s}^c(t) = \Pr[T_s < t|T_s < T_d]$ be the CDF of the conditional server session time. Then, the following expression holds:

$$F_{T_s}^c(t) = \begin{cases} \dfrac{F_{T_s}(t)}{F_{T_s}(T_d)} & t \leq T_d \\ 1 & t > T_d \end{cases}$$

As a consequence, the conditional mean server session time writes as:

$$\overline{T}_s^c = E[T_s|T_s < T_d] = \int_0^{T_d}\left(1 - \frac{F_{T_s}(t)}{F_{T_s}(T_d)}\right)\,dt. \tag{10}$$

Finally, we have that the total time a request spends in the system is given by the following expression:

$$\begin{aligned} T_d^{\text{tot}} &= T_d + \frac{F_{T_s}(T_d)}{1 - F_{T_s}(T_d)} \int_0^{T_d} \left(1 - \frac{F_{T_s}(t)}{F_{T_s}(T_d)}\right) dt \\ &= \frac{1}{1 - F_{T_s}(T_d)} \left(T_d - \int_0^{T_d} F_{T_s}(t)\, dt\right). \end{aligned} \tag{11}$$

Now, assuming the CDF of the server session time to be known, it is possible to compute T_d^{tot} as a function of T_d. For the specific case where the server session time is exponentially distributed, it is possible to express T_d^{tot} in a closed form, using Eq. 11. Thus, we have that:

$$T_d^{\text{tot}} = T_s \left(e^{T_d/T_s} - 1\right). \tag{12}$$

As an illustrative example, consider the following setting. Let's assume server nodes operate with four priority classes. Also, assume client requests to fall in each priority group with probability $p_i = p = 1/4$. Let the service rate be uniform and equal to $\mu_i = \mu = 1\text{minutes}^{-1}, \forall i \in P$, and $r = 1$.

Since we focus on servers implementing the time-dependent priority discipline, we now define the coefficients in case of proportional or additive scheme. For the proportional scheme, let $b_i \in \{1, 2, 4, 10\}$. For the additive scheme, let $a_i \in \{1, 15, 30, 50\}$ minutes, and $b_i = b = 1$. In both cases, we consider the closed system scenario, with $\lambda = \mu$. Finally, let the CDF of the server session time to be exponentially distributed with mean 500 minutes.

Now, for a given buffer size k, we can use Theorems 1 and 3 to compute the values of the mean time T_i spent by client requests in the system (recall that $T_i = W_i + 1/\mu_i$), for each priority class i.

Fig. 3 illustrates, for different values k of the buffer size, the mean time spent in the system for the lowest priority requests and for the highest priority requests. For comparison, we also show the mean waiting time of such requests in case of FCFS discipline.

The results for the particular numeric example we discuss here are useful to illustrate the impact of node churn on the mean time spent in the system by client requests. For example, when $k = 250$, the ratio between the download time of the lowest and the highest priority groups is approximately 17, while the ratio of the coefficients b_i is 10. This indicates that node churn introduces a *distortion* ($+70\%$ in this case) that would not be present in a stable system. This distortion is present also in the FCFS case: since we are considering a closed system, where the number of customers is constant and equal to k, the mean waiting time with FCFS discipline is simply k/μ, i.e., linear with k. We note instead a super-linear growth, with a 30% distortion. The additive scheme is not immune to the distortion due to churn: however, the effects of churn ($+32\%$) are closer to those experienced for servers implementing the FCFS discipline.

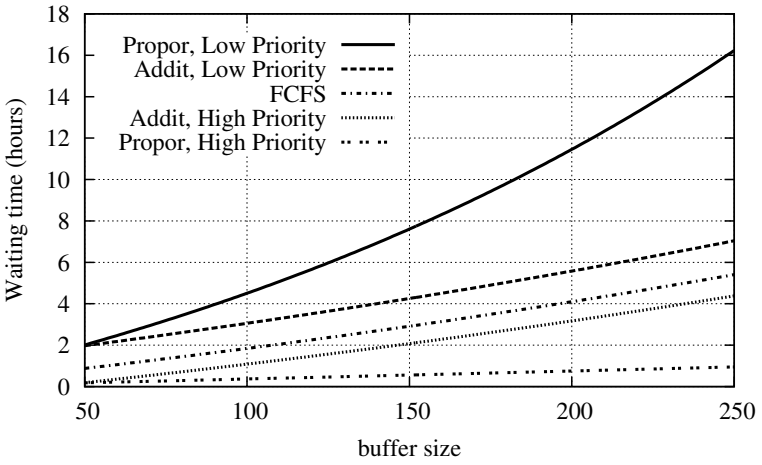

Figure 3: Mean download time for different buffer sizes, scheduling policies and priority groups.

5 Applications

In this Section we overview two applications[2] that use time-dependent priority disciplines akin to the ones we analyzed in Sec. 3. Service differentiation has been traditionally studied in the context of IP networking [10, 17]: our aim is to show that our results may be used to analyze a variety of applications that operate in the heavy traffic regime.

5.1 File-sharing Applications

In this Section we focus on the eMule file-sharing application [20,23] and show the inner principles of the priority discipline implemented in each eMule peer.

The motivation for eMule peers to use a priority scheme for awarding upload slots to remote peers stems from the fact that peers may behave selfishly and *free-ride* on the system resources. As such, the priority scheme is effectively an incentive mechanism that aims at fostering peer cooperation. However, unlike other popular file-sharing applications such as BitTorrent [24], which implements an instantaneous mechanism akin to the tit-for-tat scheme, in eMule time plays an important role.

Before proceeding any further, we shall note that the eMule system operates in the heavy traffic regime: the request rate to access content approaches or is larger than the service rate a peer can offer[3]. In the following, we will gloss over several

[2]The heterogeneity of the applications we consider and the implementation details cannot be studied in depth within the space constraints of this paper.

[3]We were not able to find in the literature a work to support this statement (which is based on user experiences): for this reason we are doing a measurement campaign. The preliminary results confirm the statement.

implementation details of the eMule application and focus only on the elements that are related to the priority discipline we are studying.

Each peer in eMule records the volume of data exchanged (download and upload) with every other peer it has interacted with in the past, for a finite amount of time. The combination of these two values is referred to as *credits*. Such credits are used to assign the priority that remote peers will be granted for each content request. Note that credits are content oblivious, *i.e.*, they are accumulated by each peer independently of the requested or served content. Furthermore, it should be noted that credit associated to a peer are never stored on the peer itself. For example, if peer A exchanged data with peer B and C, both peer B and C will maintain a distinct value for the credits of peer A. Credits are "sealed" such that the credit that peer B holds for peer A cannot be forwarded to peer C.

A peer in eMule implements a variant of the *proportional* priority discipline *with preemption* described in Sec. 3. The time-dependent priority scheme is governed by an expression similar to that defined in Eq. 2. For a generic request j received from a remote peer, its priority over time is computed as follows:

$$q_j(t) = \left(t - T_{\text{arrival}} + T_{30}\mathbb{I}_s(t)\right) \cdot f_p \cdot C_j(t)$$

where T_{arrival} is the arrival time of the request, $t \geq T_{\text{arrival}}$, T_{30} is a constant equal to 30 minutes, $\mathbb{I}_s(t)$ is the indicator function for the service – which takes the value 1 if the request is in service, and 0 otherwise – f_p is a constant value associated to each file, and $C_j(t)$ is the priority coefficient for that specific request (derived from the credits), which varies over time.

It is crucial to note here that pending requests may change priority class while they are waiting to be served (or even while they are being served). This is an important extension to the model presented in Sec. 3.1.2. Indeed, the coefficient $C_j(t)$ is computed as follows:

$$C_j(t) = \max\left(1, \min\left(\frac{2D(t)}{U(t)}, \sqrt{D(t)+2}, 10\right)\right)$$

where $D(t)$ and $U(t)$ is the total volume of data (expressed in MBytes) respectively downloaded and uploaded at time t by the peer that issued the request j, as tracked by the peer currently acting as a single server queue for that particular request j. In eMule, the constant f_p can take one of the following values: 0.2, 0.6, 0.7, 0.9, 1.8. As a result of the "min" and "max" operations, the value of $C_j(t)$ is lower bounded to 1 and upper bounded to 10.

The entire distributed system that involves all peers interacting to exchange some content can be studied through the lenses of our results. Besides the complicated nature of the expression that defines the credit system, it is possible to analyze the behavior of the proportional priority discipline using the analysis presented in Sec. 3.1.2.

Let assume, for the sake of clarity, that the term $T_{30}\mathbb{I}_s(t)$ can be neglected and the constant f_p associated to each file is set to its default value (0.7), which will be

equal for all files. In this case, if the mean download time for a request with the highest possible priority is T_H, then the mean download time for a request for the lowest possible priority will be $T_L = 10T_H$ (since the ratio between the maximum possible value and the minimum possible value of $C_j(t)$ is 10).

In summary, in this Section we have delved into the details of the service differentiation mechanism implemented in eMule. The time-dependent, proportional priority scheme adopted by the system designers introduces the notion of the "history" of past interactions among peers to compute the coefficients that govern the generic priority law we analyzed in this work.

Despite the additional complexity due to time-dependent tuning of such coefficients, we showed that the mathematical tools we developed in this work can be effectively used to understand the mean download times of requests belonging to different classes of priority. We are currently performing some real-life measurements to verify the precision of our model.

Additionally, our analysis hints at a possible drawback of the current proportional scheme currently implemented in eMule: the distribution of the download times for content requests may exhibit a heavy tail, as discussed in Sec. 3.3. Hence, a possible direction to improve the performance of the eMule application would be to adopt an additive scheme based on the lines of what we have presented in this work.

5.2 Operating Systems Scheduling

The design of an Operating System (OS) scheduler is meant to allocate hardware resources (e.g. CPU) appropriately to all applications. In this Section we focus on the Linux OS scheduler, which has received great attention from the open source community.

An OS scheduler (especially for a desktop environment) caters low-latency and interactive applications. Furthermore, an OS scheduler strives at achieving fairness in terms of access to hardware resources.

A recent scheduler, that has been deployed on all Linux OSs, is the $O(1)$ scheduler, which is the focus of this section. $O(1)$ has been designed with the ultimate goal of allocating hardware resources to multi-threaded applications. In particular, the system operating point matches that of the *heavy traffic* scenario we have described and studied in Sec. 2.

In the $O(1)$ scheduler, each processor in the system maintains a queue of tasks that requires to be served. Each queue keeps track of all runnable tasks using 2 arrays: an *active* array and an *expired* array. It is outside the scope of this Section to detail the internals of the scheduler. Here we focus on the way active tasks are scheduled, and refer the reader to [25] for an in depth analysis of $O(1)$.

The scheduler selects a runnable task with the highest priority from the active array to execute on the CPU for a pre-determined time slice; ties are broken with a round robin approach. The $O(1)$ scheduler follows a time-dependent priority discipline akin to the additive scheme described in Sec. 3. Since in $O(1)$ lower values

of $q_i(t)$ imply higher priorities – whereas in this paper we consider the opposite –, we invert the sign of the priority groups, i.e., instead of priorities ranging from 100 to 139, we let the priority range be $\{-139, -100\}$. With this notation, we can express $q_i(t)$ as

$$q_i(t) = \min\Big(\big(b(t - T_{arrival}) - a_i\big), \big(-a_i + 10\big), \big(-100\big)\Big) \tag{13}$$

where $b = 10$, $a_i \in \{101, 139\}$.

For the sake of clarity, we omit several technicalities that are used to combine strict priority and time-dependent disciplines, and focus on the latter. Note also that the dynamic value of the priority associated is bounded to -100. In Eq. 13 we can see that part of the law is determined by the additive scheme we studied in Sec. 3.2.

In summary, time-dependent priority scheduling has been applied to scheduling tasks in recent OSs, and the ultimate goal of providing fairness in accessing CPU resources has guided developers to what we have defined in this work as additive schemes. To the best of our knowledge, no work has provided a model for the $O(1)$ scheduler. Our results can be used to analyze the relative performance of tasks, as scheduled by the OS, when an additive priority scheme is used. Furthermore, our findings can help in determining a more fine-grained tuning of the coefficients that govern the priority scheme.

6 Numerical Results

The purpose of this Section is to validate the analysis we developed in Sec. 3 and 4. To do so, we use a custom, event-driven simulator[4] that implements the distributed model described in Sec. 4 and study a *generic* file-sharing application, both with a static set of peers and with churn. Due to space constraints, here we present results for the *non-preemptive* case, but similar findings can be obtained when considering the preemptive case.

Our results are helpful to assess the accuracy of our model for predicting the mean waiting time of the requests issued by the peers in the system, for a given set of parameters that define the behavior of each peer, and for a simple choice of the coefficients that govern the time-based priority disciplines we analyzed in this work.

In the following, we first describe in detail the main traits of our simulator and then move on to illustrate the good match between the experiments we performed and the theoretic results obtained by applying Theorems 1 and 3, together with Corollaries 1 and 3.

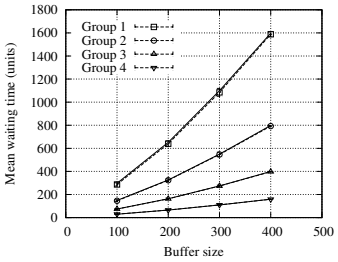

Figure 4: Mean waiting time for different buffer sizes, proportional priority scheme (no churn).

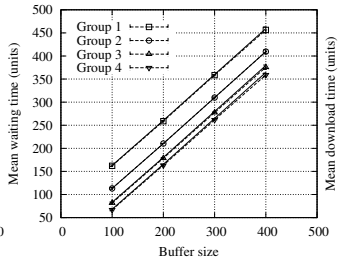

Figure 5: Mean waiting time for different buffer sizes, additive priority scheme (no churn).

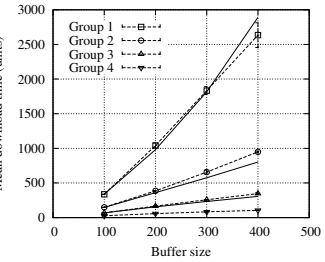

Figure 6: Mean download time for different buffer sizes, proportional priority scheme (with churn).

6.1 Methodology

The goal of our custom simulator is to be sufficiently general to describe any distributed system in which each entity holds and requests a set of scarce resources for which there is a high demand. In particular, our work can be easily described when the system materializes as a generic P2P file-sharing application in which each peer holds a set of files that other peers are eager to download.

Each peer in the system implements a time-dependent priority queue of fixed size k, which we regard as a simulation parameter. Our results cover both the *proportional* and the *additive* schemes described in Sec. 3. However, to simplify our validation, we do not implement a complex method to set the coefficients that govern the priority mechanisms we study, such as the one described in Sec. 5.1. In the following we assume each node to belong to a pre-set priority group, each characterized by a fixed coefficient, i.e. the b_i and a_i are time-invariant. In our simulations, the probability for a peer (and hence its requests) to belong to a priority group and the coefficients of the priority scheme constitute an input of the simulator.

We simulate a file-sharing application composed of N peers that are supported by a centralized lookup service keeping track of the files that exist in the system and their location. We assume there exist F different files, and that each files is initially replicated C times in distinct peers of the system.

Upon joining the system, a peer selects uniformly at random a subset of size $S < F$ of the available files (excluding those files owned locally by the peer) and, with the help of the lookup service, locates the whole set of remote peers currently holding a copy of each of them. Then, the peer selects, for each file, a random remote peer that holds a replica of that file and issues a request.

Upon receiving a request to download a file from a remote peer, the local peer can be in any of the following two states. If the local peer is idle, then it simply serves the remote peer. Otherwise, the local peer pushes the received request in its

[4]The code of our simulator is available for download at [26].

queue. When the service of the current request terminates (that is, a particular file has been completely uploaded to a remote peer), the local peer updates the priority of all current pending requests using the proportional or additive variant of Eq. 2, and it selects the next request to serve that holds the highest priority value.

When a request has been served, the peer that has issued the request selects a new file (not owned) and a peer that holds such file, and it sends the request to this peer. In this way, there are at most S requests in the whole system for each peer.

We simulate the effects of peer *churn* as follows. Each peer remains online for a randomly chosen session time, with a system-wide mean session time equals to T_s. When a peer goes offline, it removes all the requests from its priority queue. As a consequence, a peer whose request has been dropped due to churn, immediately issues the request to another remote peer that may hold a copy of the desired file. Note that this operation is performed upon receiving an update from the lookup service on the peers currently holding a copy of the required file. Additionally, if a peer is serving (or it is supposed to serve) a remote peer that has left, then the service is terminated (or the request is dropped from the queue).

We note that by properly setting the parameters of the simulator that describe the system size and resources (N,F,S) it is possible to ensure the *heavy traffic* regime, which constitutes the operational setting we consider in this work.

6.2 Settings

We consider four different groups of priority. In the proportional scheme we have $r = 1$ and $b_i = \{1, 2, 4, 10\}$, while in the additive scheme we have $b = 1$ and $a_i = \{5, 20, 50, 100\}$. We set the number of peers, the number of files and the number of requests issued by a peer to $N = 10^4$, $F = 40$ and $S = 10$ respectively.

We study different buffer sizes k: for each buffer size, we perform 5 runs and compute the confidence interval for a confidence level of 95%. The duration of each run has been set to reach the steady state for a sufficiently long time [27].

Note that, due to the constraints on the maximum number of requests issued by a peer, we can not control the arrival process and impose a specific arrival rate: we can only observe it during the simulation.

The service rate is the same for all groups, and it is exponentially distributed with rate 1 unit of time, so that the mean waiting time that we measure are expressed as a multiple of such unit.

In case of churn, we have considered two cases for the session times: exponentially and Weibull distributed. We use the exponential distribution (with mean 500 units) due to the simplicity of the expression of Eq. 12. We use the Weibull distribution as a representative session time distribution of real systems [22]. Since in both cases we have obtained a good match between theoretic results and simulations, in the following we show the results only for the exponential case, to facilitate reproducibility by the community.

6.3 Results

We now present the results of our experiments in terms of the mean waiting times for requests issued by customers. In our application scenario, this clearly translates into the mean download times achieved by peers, as imposed by the two priority schemes we study in this work. As such, in the following we show the mean waiting time per priority class, and use as a simulation parameter the buffer size k.

In case of no churn (infinite session time), we consider a generic peer inside the system and we record all the traffic through such a peer. Our goal here is to compare experimental to theoretic results. In order to do so, we need to compute the constants defined in Corollaries 1 and 3. Such constants are a function of both the coefficients of the priority scheme and the traffic composition, i.e., the probability that a request belongs to the different priority groups. As previously noted, we can not impose this probability as input, but we can only observe it during the simulation. As such, the following plots are computed "a-posteriori", *i.e.*, we measure the effective request arrival rate for each group and use it to compute the constants defined in Corollaries 1 and 3, and the theoretic mean waiting times using Theorems 1 and 3.

Figures 4 and 5 illustrate the mean waiting time for the requests of the four priority groups, for a variable buffer size k, for the proportional and additive schemes respectively. In the figures, the continuous lines indicate the theoretic values of the mean waiting times, while the dashed lines with error bars indicates experimental results. The confidence intervals (indicated by the error bars) represents 1-2% of the estimated value, thus it is difficult to see them on the figure.

For the proportional priority scheme, shown in Figure 4, we note a good match between theoretic and numerical results. Given the constant ϕ defined in Corollary 1, one would expect that, as k varies, ϕ should grow linearly. We have a different behaviour, since ϕ depends also on the traffic composition, which, we observed, changes with different buffer sizes.

In case of the additive priority scheme, shown in Figure 5, we have also a good match between theory and simulations. The traffic composition seems to be less sensitive to the buffer size, thus the mean waiting times varies linearly with k.

We now turn to the case where peers arrive and leave the system. Since the requests issued by peers pass through multiple server peers, we can not simply observe a generic peer and the request it serves. For this reason, we have selected four different peers (one for each priority group), imposing an infinite session time for such peers, and we record all the requests issued by these peers.

When we compute the theoretic results using Eq. 12, we need to know the mean session time of the peers and T_d, the mean waiting time for a specific priority group in case of no churn. The problem is that we can not compute the absolute value of T_d, since it depends on the traffic composition which is not known by the peer issuing the request.

Our approach to solve this problem is to use the properties given by Theorems 1 and 3. For instance, consider the proportional case. If we know T_i (recall that $T_i = W_i + 1/\mu_i$, then we can compute $T_j = \frac{b_i}{b_j}\left(T_i - \frac{1}{\mu_i}\right) + \frac{1}{\mu_j}$. In case of churn, we can measure T_i^{tot}, the mean total download time. Inverting Eq. 12 we obtain T_i from T_i^{tot}; from T_i we compute T_j and, using Eq. 12, we can obtain T_j^{tot}, $\forall j$.

In Figure 6 we show the results for the proportional scheme. For each buffer size k, we consider the highest priority group as the reference group (T_i^{tot}), and we compute the theoretic value of the download time T_j^{tot} for all the other groups. Also in this case, there is a good match between simulation results (dashed lines with error bars) and theoretic (continuous lines).

In summary, our results indicate a very good match between experimental and theoretic results: our work represent an effective tool that can help designing systems (both following the single server queue or the distributed models) that require service differentiation among classes of requests. Moreover, in the particular case of a distributed system in which the priority class assigned to a request depends on the past behavior of the peer issuing the request (e.g. as discussed in Sec. 5.1), our work provides users with the means to guide their behavior based on the desired quality of service and their own objective function.

7 Conclusion

The ability to predict the performance of widespread applications, deployed in scenarios ranging from desktop environments to large scale distributed systems, is of crucial importance, especially when the resources that come into play are scarce and the demand to access them is high.

In such context, it is often necessary to differentiate classes of customers by assigning a priority to the requests they generate when trying to access resources. Armed with the realization that, to obtain service differentiation, a strict priority scheme is not sufficiently flexible, in this paper we set off to provide a theoretic foundation for the analysis of a range of time-dependent priority disciplines.

Our work focused explicitly on the analysis of real systems that could accommodate a *finite* number of customer requests, while operating in an heavy traffic regime. Our analysis showed that it is possible to derive simple laws that govern the service differentiation achieved by a range of priority mechanisms. As such, our tools allow system designers to focus on the configuration of a handful set of parameters and enable them to predict the relative performance achieved by customer requests that compete to access the services of a variety of applications.

Finally, we described a number of popular applications that adopt service differentiation, including P2P file-sharing applications and OS scheduling: our results may help in understanding how to operate and tune their priority mechanisms. As a consequence, our future research agenda will include a performance evaluation, complemented by a measurement study, of the described applications with the help of the results described in this paper.

References

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

[2] P. White, "Rsvp and integrated services in the internet: A tutorial," *IEEE Communication Magazine*, pp. 100–106, 1997.

[3] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding phb," *IETF RFC 2598*, 1999.

[4] D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Trans. on Networking*, vol. 6, pp. 362–373, 1998.

[5] A. Charny and J. Le Boudec, "Delay bounds in a network with aggregate scheduling," in *QOFIS*, 2000.

[6] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu, "On achievable service differentiation with token bucket marking for TCP," in *ACM SIGMETRICS*, 2000.

[7] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *ACM SIGCOMM*, 1999.

[8] C. Cetinkaya and E. W. Knightly, "Egress admission control," in *IEEE INFOCOM*, 2000.

[9] A. Odlyzko, "Paris metro pricing: The minimalist differentiated services solution," in *IEEE/IFIP Int. Workshop Quality of Service*, 1999.

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

[11] A. Cobham, "Priority assignment in waiting line problems," *Operations Research*, vol. 2, pp. 70–76, 1954.

[12] L. Kleinrock, "A delay dependent queue discipline," *Naval Research Logistics Quarterly*, vol. 11, pp. 329–341, 1964.

[13] A. Netterman and I. Adiri, "A dynamic priority queue with general concave priority functions," *Ops. Res.*, vol. 27, pp. 1088–1100, 1979.

[14] U. Bagchi and R. Sullivan, "Dynamic, non-preemptive priority queues with general, linearly increasing priority function," *Operations Research*, vol. 33, pp. 1278–1298, 1985.

[15] L. Kleinrock and R. Finkelstein, "Time dependent priority queues," *Operations Research*, vol. 15, pp. 104–116, 1967.

[16] R. Nelson, "Heavy traffic response times for a priority queue with linear priorities," *Operations Research*, vol. 38, pp. 560–563, 1990.

[17] M. Leung, J. Lui, and D. Yau, "Adaptive proportional delay differentiated services: Characterization and performance evaluation," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 801–817, 2001.

[18] A.-L. Barabási, "The origin of bursts and heavy tails in humans dynamics," *Nature*, vol. 435, pp. 207–211, 2005.

[19] P. Blanchard and M. Hongler, "Modeling human activity in the spirit of barabasi's queueing systems," *Phys. Review E*, vol. 75, p. 026102, 2007.

[20] "The emule project," http://www.emule-project.net/.

[21] N. Prabhu, *Stochastic processes: basic theory and its applications*. New York: Macmillan, 1965.

[22] M. Steiner, T. En-Najjary, and E. Biersack, "Long term study of peer behavior in the kad dht," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1371–1384, 2009.

[23] Y. Kulbak and D. Bickson, "The emule protocol specification," Leibniz Center TR-2005-03, School of Computer Science and Engineering, The Hebrew University, Tech. Rep., 2005.

[24] B. Cohen, "Incentives build robustness in BitTorrent," in *First Workshop on Economics of Peer-to-Peer Systems*, 2003.

[25] C. Wong, I. Tan, R. Kumari, J. Lam, and W. Fun, "Fairness and interactive performance of o(1) and cfs linux kernel schedulers," in *International Symposium on Information Technology (ITSim2008)*, 2008.

[26] "PRISM: PRIority SiMulator," http://www.megaupload.com/?d=LZ2E0ZOG.

[27] J. Banks, J. Carson, B. Nelson, and D. Nicol, *Discrete-Event System Simulation*. Prentice Hall, 2004.

A Proofs

A.1 Proof of Theorem 1

We consider a generic request coming from group p, and its mean waiting time, W_p. We start from the P simultaneous equations used to derive the time spent in the system defined in [12]. Let N_i be the mean number of requests of group i in the queue, and let f_{ip} be the expected fraction of such requests which receive service before the newly arrived request from group p.

Let M_i be the mean number of requests of group i which arrive during W_p, and let g_{ip} be the expected fraction of such requests which receive service before the generic request of group p we are considering.

Given these definitions, for a generic class p we have:

$$W_p = W_0 + \sum_{i=1}^{P} \frac{N_i f_{ip}}{\mu_i} + \sum_{i=1}^{P} \frac{M_i g_{ip}}{\mu_i}. \tag{14}$$

We need to compute the different parameters. In case of N_i, we can use Little's theorem, obtaining $N_i = E[\lambda_i]T_i$, where $E[\lambda_i]$ is the mean arrival rate for class i. In case of M_i, when observing the system for W_p seconds, we see $M_i = E[\lambda_i]W_p$ arrivals. In both cases, N_i and M_i, the mean arrival rate for class i has different meanings depending on the system characteristics: infinite buffer, finite buffer and closed system. With infinite buffer, λ is the external arrival rate and $E[\lambda_i] = p_i\lambda$. For the closed system, we have $E[\lambda_i] = p_i\mu$. With finite buffer, for notational simplicity, we use the symbol λ to denote the traffic that has entered in the system, and $E[\lambda_i] = p_i\lambda$. Accordingly, in the following we will use the symbol ρ_i to indicate the ratio between the mean arrival rate and the service rate of group i, independently from the case we consider (infinite buffer, finite buffer or closed system).

For the parameters f_{ip} and g_{ip}, we note that the derivation obtained in [12] and [15] are based only on the Little theorem, which is valid for the finite buffer and closed system cases[5]. Therefore, we can use those results and arrive at the following expressions:

$$f_{ip} = \begin{cases} (b_i/b_p)^{1/r} & i < p \\ 1 & i \geq p \end{cases}$$

$$g_{ip} = \begin{cases} 0 & i \leq p \\ 1 - (b_p/b_i)^{1/r} & i > p \end{cases}$$

Combining all the information, we obtain

$$W_p = \frac{W_0 + \sum_{i=1}^{p-1} \rho_i W_i \left(\frac{b_i}{b_p}\right)^{1/r} + \sum_{i=p}^{P} \rho_i W_i}{1 - \sum_{i=p+1}^{P} \rho_i \left(1 - \left(\frac{b_p}{b_i}\right)^{1/r}\right)}. \tag{15}$$

At this point, [12] invokes the Kleinrock's conservation law to simplify the expression. Since we are considering not only the infinite buffer case, but also the finite

[5]This is true under the hypothesis used in this paper: in particular, for the finite buffer case, λ represents the rate of arrival inside the system, and for the closed system case the service rate is exponentially distributed.

buffer and the closed system cases, we analyze Eq. 15 without using the Kleinrock's conservation law.

For the lowest priority group ($p = 1$), noting that $\sum_{i=2}^{P} \rho_i = \rho - \rho_1$, in the heavy traffic case ($\rho \to 1$), from Eq. 15 we obtain

$$W_0 + \sum_{i=1}^{P} \rho_i W_i = b_1^{1/r} W_1 \sum_{i=1}^{P} \frac{\rho_i}{b_i^{1/r}}. \tag{16}$$

For the group with $p = 2$, Eq. 15 becomes, after some manipulation (always assuming $\rho \to 1$),

$$W_2 = \frac{W_0 + \sum_{i=1}^{P} \rho_i W_i - \rho_1 W_1 \left(1 - \left(\frac{b_1}{b_2}\right)^{1/r}\right)}{1 - \sum_{i=3}^{P} \rho_i + b_2^{1/r} \sum_{i=3}^{P} \frac{\rho_i}{b_i^{1/r}}}. \tag{17}$$

The numerator of the fraction, with the help of Eq. 16, can be transformed in

$$b_1^{1/r} W_1 \sum_{i=1}^{P} \frac{\rho_i}{b_i^{1/r}} - \rho_1 W_1 + \rho_1 W_1 \left(\frac{b_1}{b_2}\right)^{1/r} =$$
$$b_1^{1/r} W_1 \left(\frac{\rho_1}{b_2^{1/r}} + \sum_{i=2}^{P} \frac{\rho_i}{b_i^{1/r}}\right).$$

The denominator of the fraction can be transformed in

$$\rho_1 + \rho_2 + b_2^{1/r} \sum_{i=3}^{P} \frac{\rho_i}{b_i^{1/r}} = b_2^{1/r} \left(\frac{\rho_1}{b_2^{1/r}} + \sum_{i=2}^{P} \frac{\rho_i}{b_i^{1/r}}\right).$$

Equation 17 then becomes

$$b_2^{1/r} W_2 = b_1^{1/r} W_1. \tag{18}$$

With the help of Eqs. 18 and 16 we can compute W_3; repeating this process for all groups we obtain the desired result.

A.2 Proof of Theorem 2

In case of service with pre-emption, we consider the mean time spent in the system by a generic request coming from group p, T_p. With similar arguments used in Appendix A.1 we have the following relation:

$$T_p = \frac{1}{\mu_p} + \sum_{i=1}^{P} \frac{N_i f_{ip}}{\mu_i} + \sum_{i=1}^{P} \frac{M_i g_{ip}}{\mu_i}. \tag{19}$$

It is easy to show that the values of N_i, M_i, f_{ip} and g_{ip} remain the same as in Appendix A.1. Assuming a homogeneous system, with $\mu_i = \mu, \forall i \in P$, we obtain:

$$T_p = \frac{\frac{1}{\mu} + \sum_{i=1}^{p-1} \rho_i T_i \left(\frac{b_i}{b_p}\right)^{1/r} + \sum_{i=p}^{P} \rho_i T_i}{1 - \sum_{i=p+1}^{P} \rho_i \left(1 - \left(\frac{b_p}{b_i}\right)^{1/r}\right)}. \tag{20}$$

Comparing Eqs. 20 and 15 we notice that they have the same structure, with $1/\mu$ instead of W_0, and T_i instead of W_i. Thus the proof follows exactly the same scheme used in Appendix A.1.

A.3 Proof of Theorem 3

We consider first the non-preemptive case: the starting point remains Eq. 14, and the value of N_i, M_i are the same, while f_{ip} and g_{ip} change.

Let's assume that the newly arrived request (which we call the tagged request) belongs to group p. As said before, f_{ip} represents the expected fraction of group i requests (already in the queue at the arrival of the tagged request) which receive service before the tagged request. Clearly, if $i \geq p$, then $f_{ip} = 1$. If $i < p$, the request arrived at time Y_i seconds before the tagged one, with $W_i > Y_i$ such that

$$bY_i - a_i = -a_p$$

will receive service before the tagged request. So, the group i request should arrive at most $Y_i = (a_i - a_p)/b$ seconds before the tagged one. Let $P[w_i > t]$ be the probability that the waiting time w_i (whose mean is W_i) is greater than t, we obtain

$$f_{ip} = \begin{cases} \int_{(a_i - a_p)/b}^{\infty} \lambda_i P[w_i > t]\, dt & i < p \\ 1 & i \geq p \end{cases}$$

The parameter g_{ip} represents the expected fraction of group i requests which arrive during W_p and receive service before the tagged request. If $i \leq p$, then $g_{ip} = 0$. If $i > p$, the request will receive service if it arrives before w_p, and V_i seconds after the tagged request, with:

$$bV_i - a_p = -a_i.$$

Therefore, $g_{ip} = \lambda_i \min\big((a_p - a_i)/b, w_p\big)$. Following the same approach used in [14] it is possible to show that

$$\min\big((a_p - a_i)/b, w_p\big) = \int_0^{(a_p - a_i)/b} P[w_p > t]\, dt.$$

We then obtain

$$g_{ip} = \begin{cases} 0 & i \leq p \\ \lambda_i \int_0^{(a_p - a_i)/b} P[w_p > t]\, dt & i > p \end{cases}$$

Combining all the information, we obtain

$$W_p = W_0 + \sum_{i=1}^{p-1} \rho_i \int_{(a_i - a_p)/b}^{\infty} P[w_i > t]\, dt + \sum_{i=p}^{P} \rho_i W_i + \sum_{i=p+1}^{P} \rho_i \int_0^{(a_p - a_i)/b} P[w_p > t]\, dt. \tag{21}$$

Note that

$$\int_x^{\infty} P[w_i > t]\, dt = W_i - \int_0^x P[w_i > t]\, dt.$$

In case of heavy traffic, as done in [14], we can assume that $w_i > (a_i - a_j)/b$, for any j, and approximate the integrals by $\int_0^x P[w_i > t]\, dt \approx x$. Equation 21 becomes

$$W_p = W_0 + \sum_{i=1}^{P} \rho_i W_i - \sum_{i=1}^{p-1} \rho_i \frac{a_i - a_p}{b} + \sum_{i=p+1}^{P} \rho_i \frac{a_p - a_i}{b}. \tag{22}$$

Since $\rho \to 1$, we obtain

$$W_p - \frac{a_p}{b} = W_0 + \sum_{i=1}^{P} \rho_i W_i + \sum_{i=1}^{P} \rho_i \frac{a_i}{b} = \text{constant}. \tag{23}$$

In case of service with pre-emption, we consider the mean time spent in the system by a generic request coming from group p, T_p. With similar arguments used for the non pre-emptive case, we arrive at the following relation:

$$T_p = \frac{1}{\mu_p} + \sum_{i=1}^{P} \rho_i T_i - \sum_{i=1}^{p-1} \rho_i \frac{a_i - a_p}{b} + \sum_{i=p+1}^{P} \rho_i \frac{a_p - a_i}{b}, \tag{24}$$

which leads to:

$$T_p - \frac{1}{\mu_p} - \frac{a_p}{b} = \sum_{i=1}^{P} \rho_i T_i + \sum_{i=1}^{P} \rho_i \frac{a_i}{b} = \text{constant}. \tag{25}$$

Recalling that $T_p - \frac{1}{\mu_p} = W_p$, we have completed the proof.