Resource Allocation with Persistent and Transient Flows

Supratim Deb[1], Ayalvadi Ganesh[2], and Peter Key[2]

[1] Coordinated Science Lab., University of Illinois at Urbana-Champaign, 1308 W. Main Street, Urbana, IL 61801, USA deb@uiuc.edu
[2] Microsoft Research, 7 J J Thomson Ave., Cambridge CB3 0FB, UK
ajg,peterkey@microsoft.com

Abstract. The flow control algorithms currently used in the Internet have been tailored to share bandwidth between users on the basis of the physical characteristics of the network links they use rather than the characteristics of their applications. This can result in a perception of poor quality of service by some users even when adequate bandwidth is potentially available, and is the motivation for seeking to provide differentiated services. In this paper, stimulated by current discussion on Web mice and elephants, we explore service differentiation between persistent and short-lived flows, and between file transfers of different sizes. In particular, we seek to achieve this using decentralized algorithms that can be implemented by end-systems without requiring the support of a complex network architecture. The algorithms we propose correspond to a form of weighted processor sharing and can be tailored to approximate the shortest remaining processing time service discipline.

Keywords: Service differentiation, bandwidth allocation, decentralized control, weighted processor sharing, shortest remaining processing time.

1 Introduction

Most data in the current Internet is transferred using TCP. This protocol has two phases: a slow start phase which probes for available bandwidth up to a certain threshold, and a subsequent congestion avoidance phase that attempts to stabilize around a fair share. Despite having an aggressive ramp up phase, the throughput during slow start is typically much less than in the congestion avoidance mode due to the small size of the initial window, time-outs triggered by packet loss, etc. Moreover, the fair shares reached in the congestion avoidance phase allocate equal bandwidth to all file transfers having the same round-trip time and access bandwidth, irrespective of the sizes of the files being transferred. This results in a poor response time for short file transfers and raises the question of whether it is possible to improve performance for short file transfers without significantly degrading it for long file transfers. This question assumes particular importance in the context of the finding by a number of researchers that file sizes on the Web have a heavy-tailed distribution [6]: when file sizes vary over several

E. Gregori et al. (Eds.): NETWORKING 2002, LNCS 2345, pp. 455–466, 2002.

orders of magnitude, treating all file transfers identically may not be appropriate. This has led to research on improving the throughput of short flows, either by altering the slow-start behavior [2,11,3], or by putting short flows into a different class [19,10], or by providing a predictive service to long flows [5].

A related problem is that of sharing bandwidth between file transfers and real-time traffic such as Internet telephony or video conferencing. Real-time flows are usually long-lived and can be treated as persistent sources for purposes of analysis. They have very different quality of service requirements from file transfers. Whereas what matters for file transfers is usually the transfer time, or equivalently, average bandwidth over the entire transfer period, real-time flows typically care about the bandwidth they receive at each instant in time (or, more precisely, averages over time periods much shorter than the lifetime of the connection). The value of bandwidth to a user can be described mathematically by a utility function which captures elements of the quality of service perceived by the user. Utility functions are commonly used in economics to represent individual preferences and to address questions of fair allocation. The resource allocation problem can be cast as one of maximizing the aggregate utility of all users.

We model the utility for a file transfer as the negative of the time taken to complete the transfer. For real-time traffic, we assume that the total utility obtained is the integral of an instantaneous utility over the lifetime of the connection; the instantaneous utility, in turn, is modeled as an increasing and concave function of the bandwidth received by the flow at that instant. Such a concave function reflects diminishing marginal utility to the user as the allocated bandwidth increases. Equivalently, concavity models a preference on the part of the user for a fixed constant bandwidth over a fluctuating bandwidth allocation with the same mean. Sources with such a utility are referred to as *elastic* sources in the literature. There has recently been considerable work on bandwidth sharing between persistent elastic users [14,17]. However, the problem of combining such sources with transient sessions such as file transfers has received little attention. One recent study [15] suggests that, when the two traffic types share a network, file transfers should receive priority.

Our main results and the organization of the paper are as follows. In Section 2, we consider persistent elastic sources sharing a link with transient sessions transferring a fixed volume of data. We pose the bandwidth allocation problem as an optimization problem and solve it numerically. We then derive practical flow-control schemes that can be easily implemented in a decentralized manner, and show that these are close to optimal. In Section 3, we consider a scenario where the transient sessions have different amounts of data to transfer. The shortest remaining processing time (SRPT) policy yields the optimal bandwidth allocation. We propose a practical scheme that approximates SRPT and study its performance through simulation. We show that there is an advantage to increasing the throughput given to short flows, and that this can be done without appreciably penalizing long flows. We present our conclusions and discuss directions for future research in Section 4.

2 Bandwidth Sharing between Persistent and Transient Flows

Consider a single link of capacity C shared by a fixed number, K, of persistent flows and a variable number of short-lived flows (also called Mice). The persistent flows are modeled in aggregate as having an increasing and strictly concave instantaneous utility function $KU_e(x_e)$, where x_e is the aggregate bandwidth allocated to these flows at a specified time. The utility over a time period is given by the integral of the instantaneous utility over that period. To simplify technicalities in the analysis below, we assume in addition that U_e is differentiable. Short flows correspond to file transfers. They arrive into the system at the points of a Poisson process of rate λ and leave when the file transfer is complete. The file sizes are assumed to be exponentially distributed with mean f. Let $\rho = \lambda f/C$ denote the load offered by the short flows. We shall assume that $\rho < 1$.

There is a unit holding cost per unit time for each short flow in the system. The goal is to maximize the time average of $KU_e(x_e(t)) - N(t)$, where $N(t)$ denotes the number of short flows in the system at time t. To this end, we introduce the performance objective,

$$J_\pi(n) = \lim_{T\to\infty} \frac{1}{T} E\left[\int_0^T [KU_e(x_e(t)) \ - \ N(t)]dt \ \Big| \ N(0) = n\right] \tag{1}$$

and seek a policy π that maximizes this objective. By Little's law, the time average of $N(t)$ is the same as λ times the mean sojourn time of a file transfer, so the objective is to maximize the utility of long flows, subject to a bound on the mean sojourn time of file transfers. The objective function above is precisely the Lagrangian for this optimization problem.

We seek stationary optimal policies for the optimization problem described above. By the assumption of exponential file sizes, the state of the system is fully described by the number of short flows in progress and we have a semi-Markov decision problem. If the number of short flows is restricted to some $n_{\max}$, and non-zero capacity is allocated to short flows whenever any are present, then the Markov process is irreducible and has a finite state space. Under these conditions, it can be shown that there is a stationary optimal policy, and that it can be computed numerically using value iteration. The proof is omitted due to lack of space, but can be found in [8] along with a discussion of structural properties of the optimal control policy.

In order to compare the optimal policy with sub-optimal policies that we shall consider below, we need the following elementary bound on the performance of the optimal policy.

Lemma 1. *Suppose the state space is not truncated, i.e., $n_{\max} = \infty$. Then, for any policy π and any initial state n, we have $J_\pi(n) \le KU_e((1-\rho)C)$.*

Proof. Since the load offered by the short flows is ρ, any policy π that allocates capacity less than ρC to these flows on average will be unstable in the sense that

$N(t) \to \infty$ as $t \to \infty$. Thus, for any such policy, $J_\pi(n) = -\infty$, starting from any n. Therefore, we can restrict attention to policies π that, on average, allocate capacity no more than $(1-\rho)C$ to the persistent flows. Since U_e was assumed to be concave, we now obtain from Jensen's inequality and the non-negativity of $N(t)$ that

$$\frac{1}{T}\int_0^T [KU_e(x_e(t)) - N(t)] \le KU_e\left(\frac{1}{T}\int_0^T x_e(t)\right) .$$

Taking expectations and using Jensen's inequality once more, we get

$$J_\pi(n) \le KU_e\left(E\left[\frac{1}{T}\int_0^T x_e(t)|N(0)=n\right]\right) \le KU_e((1-\rho)C) , \qquad (2)$$

since $E[\frac{1}{T}\int_0^T x_e(t)] \le (1-\rho)C$ for all T sufficiently large, and U_e is an increasing function. □

Implementing the optimal policy requires knowledge of the number of short flows in progress and may not be practical. This leads us to consider simpler policies that are practically realizable. We show for two such policies below that they are close to optimal. In the rest of the paper, we will work with utility functions of the form

$$U_e(x_e) = \frac{1}{1-\beta}\left(\frac{x_e}{C}\right)^{1-\beta} , \quad \beta > 0. \qquad (3)$$

If $\beta = 1$, we take $U_e(x_e) = \log(x_e/C)$. These constitute a fairly general class of utility functions and have been considered by a number of authors; see, for example, [18]. The bandwidth shares assigned by TCP approximately maximize a utility function of this form with $\beta = 2$.

Static Policy. A fixed amount of bandwidth $\tilde{C} < C$ is reserved for the persistent sources and the remainder is shared equally among file transfers. This can be implemented by logically partitioning the link between persistent and short flows and using TCP for the short flows, for example.

Now, irrespective of the file size distribution, the number of short flows in progress evolves like the queue size in an $M/G/1-PS$ queue, with load $\alpha = \lambda f/(C-\tilde{C})$. The equilibrium queue length distribution is geometric with parameter α (see [13], for example), and so the mean number of short flows in progress is $E_\pi[n] = \alpha/(1-\alpha)$. The bandwidth allocated to persistent flows, $\tilde{C}$, can be expressed as $\tilde{C} = C - (\lambda f/\alpha) = (\alpha-\rho)C/\alpha$. Hence,

$$E_\pi[KU_e(x_e(n)) - n] = KU_e\Big(\frac{\alpha-\rho}{\alpha}C\Big) - \frac{\alpha}{1-\alpha} . \qquad (4)$$

Taking $\alpha = 1 - a/\sqrt{K}$ and using (3), we obtain

$$E_\pi[KU_e(x_e(n)) - n] = \frac{a\sqrt{K}}{1-\beta} + \frac{K - a\sqrt{K}}{1-\beta}\left(\frac{1-\rho-(a/\sqrt{K})}{1-(a/\sqrt{K})}\right)^{1-\beta} - \frac{\sqrt{K}}{a} + 1$$

$$= K\frac{(1-\rho)^{1-\beta}}{1-\beta} + O(\sqrt{K}) \;=\; KU_e((1-\rho)C) + O(\sqrt{K})\ . \tag{5}$$

Recall that ρC is the rate at which work is brought in by short flows, and αC is the capacity allocated to them. The choice $\alpha = 1 - a/\sqrt{K}$ corresponds to allocating most of the available capacity to short flows, reserving only a small fraction $a/\sqrt{K}$ for persistent sources.

How much worse than optimal is the static policy? One way to quantify this is to ask how large a capacity $\hat{C}$ is needed, so that the total utility achieved using the static policy on a link of capacity C is the same as the utility achieved using the optimal policy on a link of capacity $\hat{C}$. Recall that, by (2),

$$E_\pi[KU_e(x_e(n)) - n] \le \frac{K}{1-\beta}\left[\frac{(1-\rho)\hat{C}}{C}\right]^{1-\beta}$$

for a link of capacity $\hat{C}$, using any policy. Comparing this with (5), we see that $\hat{C} = C(1 - O(1/\sqrt{K}))$. In so far as K is large in the typical operating regime of interest, this shows that the static policy is close to optimal.

Implementation of the static policy requires that bandwidth partitioning be carried out by network routers. In contrast, the weighted processor sharing policy discussed next can be implemented at end systems.

Weighted Processor Sharing. Suppose each persistent source has weight 1 and each file transfer in progress has weight w, and that capacity is shared between users in proportion to their weights. In particular, each file transfer in progress gets the same share of capacity. Thus, irrespective of the file size distribution, the number of file transfers in progress can be modeled by a symmetric queue (see Lemma 3.9 in Kelly [13]), and has the invariant distribution of a birth-death process with constant birth rate λ, and state-dependent death rate $\mu_n = (C - x_e(n))/f$. Here $x_e(n)$ is the capacity allocated to persistent sources when n short flows are in progress, and f is the mean file size. If we assume further that $k := K/w$ is an integer, then it can be shown that the invariant distribution is given by

$$\pi(n) = \binom{k+n}{n}\rho^n(1-\rho)^{k+1}\ , \tag{6}$$

and a simple calculation yields $E_\pi[n] = (k+1)\rho/(1-\rho)$ for the mean number of short flows in the system. Details are omitted for brevity, but can be found in [8]. It is not possible to obtain a closed-form expression for $E_\pi[U_e(x_e(n))]$ in general, but we can obtain approximations using a Taylor expansion for U_e when k is large. After routine calculations detailed in [8], we obtain

$$E_\pi[KU_e(x_e(n)) - n] \approx$$
$$\frac{K(1-\rho)^{1-\beta}}{1-\beta} + \frac{K\rho(1-\rho)^{1-\beta}}{k} - \frac{K\beta\rho(1-\rho)^{1-\beta}}{2k} - \frac{(k+1)\rho}{1-\rho}\ . \tag{7}$$

Recall that $k = K/w$, where w is the weight given to short-lived flows. It follows from the above that, by choosing $w = \sqrt{K}$, we get $E_\pi[KU_e(x_e(n)) - n] \geq KU_e((1-\rho)C) - O(\sqrt{K})$. Since no policy can achieve a total utility greater than $KU_e((1-\rho)C)$, we conclude that the minimum bandwidth, $\hat{C}$, required by an optimal policy to achieve the same utility as achieved by the weighted processor sharing policy is given by $\hat{C} = C(1 - O(1/\sqrt{K}))$.

Thus, the weighted processor sharing policy is nearly optimal, in the same sense as the static policy. Moreover, it can be implemented by the end systems rather than the network, for example by having end systems use a weighted analogue of TCP with weights chosen as above. An alternative implementation would be to use a willingness-to-pay scheme, as described in [9], with a willingness-to-pay parameter proportional to the weights above.

Numerical Results. We now derive explicit formulas in the special case $\beta = 2$, i.e., $U_e(x) = -C/x$. Recall that this is the utility function implicitly maximized by TCP. The static policy allocates fixed capacity $\rho C/\alpha$ to the short flows; when $\beta = 2$, we obtain from (4) that the optimal value of α is $\alpha = 1 - \frac{1-\rho}{1+\sqrt{K\rho}}$, and that

$$E_\pi[KU_e(x_e(n))] \;=\; -\,\frac{K+\sqrt{K\rho}}{1-\rho}\;, \quad E_\pi[n] \;=\; \frac{\sqrt{K\rho}+\rho}{1-\rho}\;.$$

When $\beta = 2$, we can also explicitly calculate $E_\pi[U_e(x_e(n))]$ for the weighted-PS policy. We obtain

$$E_\pi[U_e(x_e(n))] = E_\pi\left[U_e\left(\frac{k}{k+n}C\right)\right] = -E_\pi\left[\frac{k+n}{k}\right] = -\frac{k+\rho}{k(1-\rho)}\;.$$

A simple calculation now yields that the optimal value of k is $\sqrt{K}$, i.e., each transient flow should be given a weight $w = \sqrt{K}$ relative to each persistent flow. With this choice of k, we get

$$E_\pi[KU_e(x_e(n))] \;=\; -\,\frac{K+\sqrt{K}\rho}{1-\rho}, \quad E_\pi[n] \;=\; \frac{\sqrt{K}\rho+\rho}{1-\rho}.$$

We compare the mean utility and number in system for the static and weighted-PS policies with those for the optimal policy, obtained numerically. For this purpose, we choose the system parameters $C = 1000$, $K = 25$, $f = 100$, and vary λ so that $\rho = \lambda f/C$ spans the interval $[0.1, 0.7]$. We truncate the state space at $n_{\max} = 100$ for the value iterations. The results are plotted below. Figure 1 shows the mean utility for the optimal, static and weighted policies, while Figure 2 shows the mean number of short flows in progress for each policy. The figures show that neither the persistent nor the transient flows suffer much by using the sub-optimal policies considered. Figure 3 shows the additional capacity required by the static policy if it is to achieve the same total utility as the optimal policy; 3(a) corresponds to $K = 25$ and 3(b) to $K = 5$. We see that the loss incurred by the sub-optimal policies is small, even for small values of K.

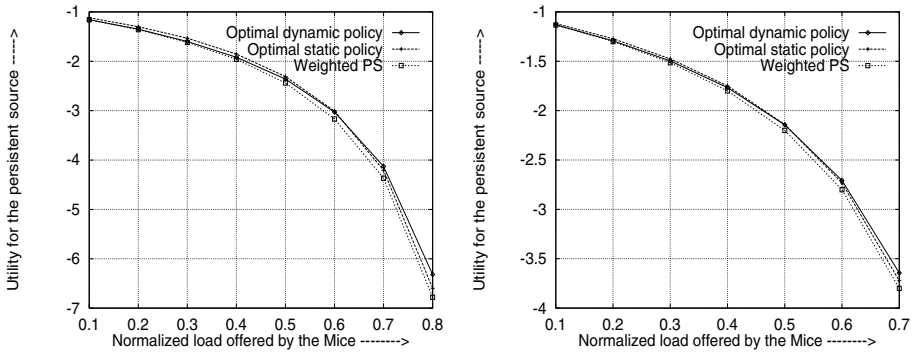

Fig. 1. Average utility of the long flow under three different allocation strategies. $C = 1000$, Mean file size=100, $U_e(x) = \frac{-C}{x}$, $n_{max} = 100$. $K = 5$ in the *left panel* and $K = 25$ in the *right panel*. The arrival rate is varied along the x-axis

3 Bandwidth Sharing between Transient Flows

We now consider how capacity should be shared between file transfers when the sizes of the files being transferred might vary over several orders of magnitude. If the objective is to minimize the number of file transfers in progress (equivalently, the mean holding cost or mean sojourn time) and the amount remaining to be transferred is known, then a simple interchange argument shows that the optimal policy is to give priority to the file with shortest remaining processing time (SRPT). This policy has been proposed in the context of Web servers [12, 1]. However, it is not suited to our problem for a couple of reasons. First, it needs a centralized controller to assign priority (or a distributed leader election protocol, which imposes a high overhead). Second, while the concept is clear for a single bottleneck link or resource, it does not generalize easily to multiple bottlenecks. This motivates us to consider a generalization of the weighted PS policy introduced in the previous section and show that it can be made to approximate SRPT. Though the analysis and simulations in this paper pertain to a single link, the algorithms we propose generalize easily to networks.

We also note that, in networks, the stability region of priority policies such as SRPT is not easily obtained; it is known that the "$\rho < 1$" condition that the offered load on each link be smaller than its capacity is not sufficient for stability. On the other hand, this condition does guarantee stability for the algorithms we consider, as shown in [4]. That is another advantage of the proposed algorithms in the network context.

We continue to work with the optimization problem posed in the previous section. There, we considered how to split capacity between persistent and transient flows but did not consider further how the capacity allocated to transient flows should be shared between them. If file sizes are exponentially distributed and the allocation decision has to be made without knowing the sizes of all file transfers in progress, then it does not matter how this allocation is made; any

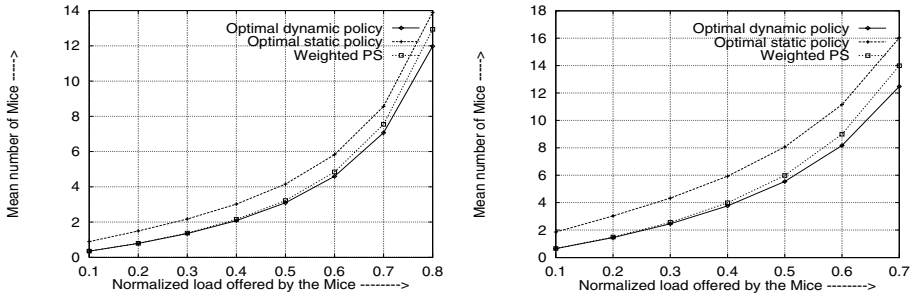

Fig. 2. Average number of short flow under three different allocation strategies. $C = 1000$, Mean file size=100, $U_e(x) = \frac{-C}{x}$, $n_{max} = 100$. $K = 5$ in the *left panel* and $K = 25$ in the *right panel*. The arrival rate is varied along the x-axis

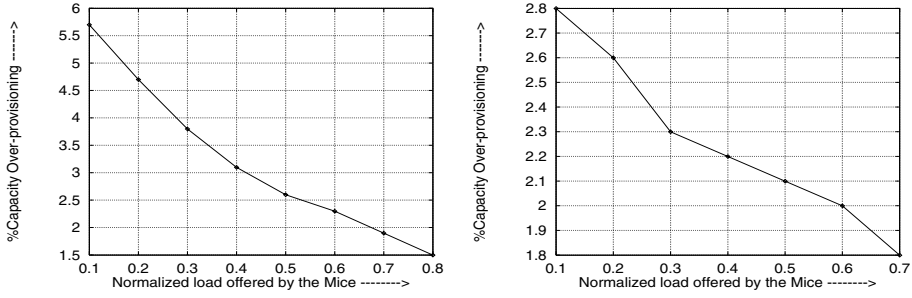

Fig. 3. Capacity over-provisioning sufficient for optimal static allocation to outperform optimal dynamic allocation. $C = 1000$, $f = 100$, $U_e(x) = \frac{-C}{x}$, $n_{max} = 100$. $K = 5$ in the *left panel* and $K = 25$ in the *right panel*

allocation that doesn't leave capacity idle achieves the same mean number in system. If file sizes are known or if they aren't exponentially distributed, then this is no longer true; for example, if file sizes are heavy-tailed, the first-come-first-served policy performs worse than processor-sharing. We noted above that, if file sizes are known, then SRPT is optimal.

We now consider a weighted processor sharing policy where each transient flow chooses its own weight based on its residual file size. Suppose the weights are chosen according to

$$w_i = w_{min} \ + \ (w_{max} - w_{min}) \exp(-a f_i^r) \ , \tag{8}$$

where w_i and f_i^r denote the weight assigned to the i^{th} flow and its residual file size, and w_{min}, w_{max} and a are system parameters. The link capacity C is shared between flows in proportion to their weights, i.e., flow i receives capacity $w_i C/W$, where W denotes the sum of w_i over all flows in the system. A similar policy has been proposed recently in [7].

We shall assume that W is constant over time. Such an assumption is plausible in a large system operating in a steady-state regime. In particular, if the system carries a large number of persistent flows, then the fluctuation in W is only due to short flows entering and leaving the system, and can be neglected to a first approximation. With this assumption, we can calculate the sojourn time of a file transfer as a function of the initial file size. Letting f_i denote the size of file i, we have

$$f_i^r(0) = f_i \ , \quad \frac{d}{dt} f_i^r(t) \ = \ - \ \frac{w_i(t)}{W} C \ , \tag{9}$$

where $w_i(t)$ is specified in terms of $f_i^r(t)$ via (8). Here, t denotes the time since the arrival of flow i into the system. Let $T_i = \inf\{t > 0 : f_i^r(t) = 0\}$ denote the sojourn time of flow i. A straightforward calculation using (8) and (9) yields

$$T_i = \frac{W}{aCw_{min}} \log \left[1 + \frac{w_{min}}{w_{max}} \left(e^{af_i} - 1 \right) \right] \ .$$

The (unweighted) processor sharing policy is recovered in the limit $a \to 0$, in which case $T_i = Wf_i/w_{max}$. The sojourn time of a file is thus proportional to its size, which is desirable in terms of fairness but has the disadvantage that small files see poor performance.

In order to quantify the extent to which the proposed service discipline favors short flows, we compute the ratio of sojourn times for two different files, of sizes f_1 and f_2. With plain sharing, this ratio is $T(f_1)/T(f_2) = f_1/f_2$. Denoting the ratio w_{min}/w_{max} by α, we obtain for the scheme proposed above that

$$\frac{T(f_1)}{T(f_2)} = \frac{\log \left[1 + \alpha (e^{af_1} - 1) \right]}{\log \left[1 + \alpha (e^{af_2} - 1) \right]} \ . \tag{10}$$

We observe that if f_1 and f_2 are both large relative to $1/a$ and if, moreover, αe^{af_i} is much bigger than 1 for $i = 1, 2$, then $T(f_1)/T(f_2) \approx f_1/f_2$. In other words, the *stretch*, defined as the ratio of sojourn time to file size, is roughly constant for large files, meaning that the scheme approximates processor sharing at large file sizes. In particular, it avoids starvation of very large file transfers. On the other hand, if f_1 and f_2 are both small relative to $1/a$, then again $T(f_1)/T(f_2) \approx f_1/f_2$. Finally, suppose f_1 is large and f_2 is small relative to $1/a$. Then, by (10),

$$\frac{T(f_1)}{T(f_2)} \approx \frac{af_1 + \log \alpha}{\alpha a f_2} \approx \frac{1}{\alpha} \frac{f_1}{f_2} = \frac{w_{max}}{w_{min}} \frac{f_1}{f_2} \ .$$

In other words, the large file has a stretch approximately $1/\alpha$ times greater, or receives a bandwidth share approximately $\alpha = w_{min}/w_{max}$ as much as a small file. Loosely speaking, files much smaller than $1/a$ are "mice", files much larger than $1/a$ are "elephants", all mice are treated roughly equally, as are all elephants, but mice are favored over elephants. Note that this is achieved without explicitly splitting files into classes, but simply by having them choose individual weights based on their residual file sizes.

The degree to which mice are favored is determined by the ratio $1/\alpha = w_{max}/w_{min}$. This can be seen clearly in Figure 4, where we have plotted the stretch, $T(f)/f$, as a function of the normalized file size af over the range $[0, 20]$. We take $W/(Cw_{max}) = 1$ for convenience. From top to bottom, the 3 curves on the plot correspond to $1/\alpha = 5, 10$ and 20 respectively.

The plots suggest that large files receive much less capacity on average than do short files. It needs to be kept in mind that this is under the assumption that W is constant, which is not valid if there are no persistent flows. A model with no persistent flows and with an SRPT service discipline has been studied in [1], where it is shown that the stretch of long flows remains bounded. The intuition is that there will be epochs when the long flow is competing with very few or no short flows, at which times it is not handicapped by its small weight. A similar intuition applies to our model, and in fact the plots of stretch in Figure 4 correspond to "worst-case" values.

Simulation Results. We simulate a system with capacity $C = 1000$ carrying $K = 25$ persistent flows, each of which has weight 1 and has the utility function $U_e(x) = -C/x$. File transfers arrive at rate λ, and file sizes have the Pareto distribution, $P(\text{file-size} > x) = 1/(1 + (x/f))^2$, $x \geq 0$, with mean file size $f = 100$. We take $a = 1/f$, $w_{max} = 50$ and $w_{min} = 10$. Performance measures for processor sharing with the scheme described above, and processor sharing with constant (file-size independent) weights $\overline{w}$ for three different weights, 10, 50 and 25, are shown in Figure 5. The left panel shows the utility received by the persistent flows under each policy. The panel on the right shows the mean number of transient flows in the system. The simulation results are based on 12,000 events (file arrivals) with a burn-in period of 1000 time units for the system to reach stationarity, and, are averaged over multiple runs. Clearly, when $\overline{w} = 50$, the average stretch of the transient flows go down but at the cost of a reduced utility for the persistent flow. When $\overline{w} = 10$, the persistent flows perform better but the average stretch of the short flows increases a lot. However, by using the processor sharing described in this section when the weights of the transient flows are varied in a dynamic manner, the transient flows can achieve a small stretch without starving the persistent flow much. We have also shown the plots for the case when the weights are kept constant at $\overline{w} = 25$. The proposed processor sharing scheme still does better.

4 Concluding Remarks

We considered the problem of optimal bandwidth allocation in a system consisting of both persistent and transient flows. Treating all transient flows as identical, we first described simple algorithms that achieve a nearly optimal partitioning of the available bandwidth between the persistent and transient sources. We then studied the problem of how to share the bandwidth allocated to transient flows among file transfers of different sizes. We described a distributed scheme

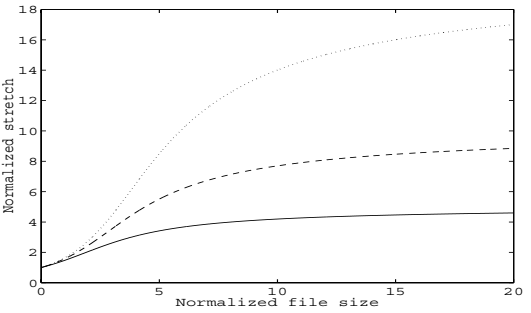

Fig. 4. Stretch vs. file size for $w_{max}/w_{min} = 20(top)$, $10(middle)$ and $5(bottom)$

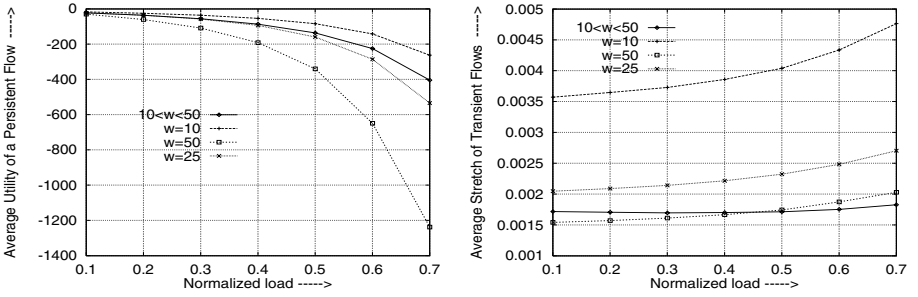

Fig. 5. Plots showing comparison of of average utility of a persistent flow (*left*), and, average stretch of short flows (*right*) with four schemes: in one the weights of the short flows are varied between 10 (w_{min}) and 50 (w_{max}) according to the scheme discussed in this section, and the other three are with fixed weights ($\overline{w}$) as 10, 50 and 25 respectively. The different parameters are, $C = 1000$, $f = 100$, $U_e(x) = \frac{-C}{x}$, $K = 25$ (the number of persistent flows). The arrival rate λ is varied along the x-axis

that can be viewed as approximating SRPT or, equivalently, as discriminating in favor of "mice" over "elephants".

The analysis in this paper pertains to idealized systems in which bandwidth is shared perfectly between users in proportion to their weights. In fact, such an allocation can be approximately achieved by decentralized adaptive mechanisms described in [14,9,16] etc., which can be implemented by end systems with minimal support from the network. Second, the optimal choice of parameters for the policies described in Section 2 requires knowledge of system parameters, which is often unrealistic. We believe that comparable performance can be achieved by adaptive policies that tune their parameters based on measurements, but this remains a topic for future research.

The policies studied here for a single link can be extended easily to networks. The extensions have the desirable property that the network is stable under the natural condition that the offered load at each resource is smaller than its capacity. This is in contrast to priority schemes which can be unstable even

when this condition is satisfied. A detailed investigation of the performance of the algorithms described here in a network context is a subject for future work.

References

1. N. Bansal and M. Harchol-Balter, "Analysis of SRPT scheduling: investigating unfairness", *Proc. ACM Sigmetrics*, 2001.
2. C. Barakat and E. Altman, "Performance of Short TCP Transfers", *Proc. Networking*, Paris, 2000.
3. Y. Bhumralkar, J. Lung and P. Varaiya, "Network Adaptive TCP Slow Start", 2000. http://www.path.berkeley.edu/∼varaiya/papers_ps.dir/jeng.pdf
4. T. Bonald and L. Massoulié "Impact of fairness on Internet performance", *Proc. ACM Sigmetrics*, 2001.
5. D. D. Clark and W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service", *IEEE/ACM Trans. Networking*, 6(4): 362–373, 1998.
6. M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes", *IEEE/ACM Trans. Networking*, 5(6): 835–846, 1997.
7. G. de Veciana, "Enhancing Both Network and User Performance Metrics for Networks Supporting Best Effort Traffic", *Thirty-Ninth Annual Allerton Conference on Communication, Control, and Computing*, Allerton House, Illinois, 2001.
8. S. Deb, A. Ganesh and P. Key, "Resource allocation with persistent and transient flows", Microsoft Research Technical Report, 2001. http://research.microsoft.com/scripts/pubs/view.asp?TR_ID=MSR–TR–2001–114
9. R. J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control", *Automatica*, 35: 1969–1985, 1999.
10. L. Guo and I. Matta, "The war between mice and elephants", Technical Report, Boston University, BU-CS-2001-0005, 2001.
11. T. Hammann and J. Walrand "A new fair algorithm for ECN-capable TCP", *Proc. Infocom*, 2000.
12. M. Harchol-Balter, M. Crovella and S. Park, "The case for SRPT scheduling in Web servers", Technical Report, MIT-LCS-TR-767, 1998.
13. F. P. Kelly, *Reversibility and Stochastic Networks*, John Wiley and Sons, New York, 1979.
14. F. P. Kelly, A. Maulloo and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability", *J. Oper. Res. Soc.*, 49: 237–252, 1998.
15. P. Key and L. Massoulié "User policies in a network implementing congestion pricing", Workshop on Internet Service Quality Economics (ISQE), 1999.
16. R. J. La and V. Anantharam, "Charge-sensitive TCP and rate control in the Internet", *Proc. Infocom*, 2000.
17. S. H. Low and D.E. Lapsley, "Optimization flow control – I: Basic algorithm and convergence", *IEEE/ACM Transactions on Networking*, 7: 861–875, 1999.
18. J. Mo and J. Walrand, "Fair end-to-end window-based congestion control", *IEEE/ACM Trans. Networking*, 8(5): 556–567, 2000.
19. S. Yilmaz and I. Matta, "On class based isolation of UDP, short lived and long lived flows", *Proc. Ninth Intl. Symp. Modeling, Analysis And Simulation of Computer And Telecommunication Systems,* Cincinnati, 2001.