Bandwidth Sharing: The Role of User Impatience

Shanchieh Yang, Gustavo de Veciana
Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712

***Abstract*— Empirical work has shown that up to 20% of the volume transferred on data networks might correspond to 'aborted' connections, *i.e.,* badput. With this in mind, we propose two generic models that capture a variety of user impatience behaviors, and investigate their impact on user perceived and system performance achieved by various bandwidth sharing schemes. Our study suggests that differentiating bandwidth allocation based on job size, rather than using traditional fair share allocations, results in a more 'graceful' performance degradation, and particularly in the presence of impatient users, leads to better network efficiency as well as user perceived performance.**

I. Introduction

As evidenced by traffic loads on the Internet, exchanging data files has been, and will continue to be, a major fraction of the volume carried by data networks. Typically such transfers correspond to sending known amounts of data with a wide tolerance to changes in the transmission rate during the transfer. Thus such flows can adapt their rates to dynamically share congested links with contending transfers. However, as the congestion level goes up, depending on the way bandwidth is shared, some or all flows may see poor performance, possibly leading to aborted transfers, *i.e.,* stopped before completion, due to *user impatience*. Empirical evidence collected from representative servers [1], [2], [3] suggests that non-negligible amounts of data may correspond to aborted transfers, *e.g.,* [1] found that 11% of all transfers were interrupted, corresponding to 20% of the transfered volume. Users may abort their transfers, *e.g.,* push the stop button on the a browser, for various reasons, such as incorrect document address, long connection setup time, or poor performance during transfer. Our focus herein is on user impatience with respect to *transfer dynamics* once a connection is established. Of particular interest will be the interaction between user impatience characteristics and bandwidth sharing policies.

Bandwidth in today's Internet is shared in a dynamic fashion, *i.e.,* flows come and go and allocations are mediated through TCP's congestion control mechanisms. Recently researchers have focused on the 'fairness' of bandwidth allocation mechanisms among contending flows, *e.g.,* [4], [5], [6], by defining abstract notions of network utility as a function of the current allocations. Alternatively, one may consider optimizing user perceived performance. For example, we propose in [7] a simple modification to TCP by incorporating size-based differentiation which significantly enhances the perceived average bit transmission delay, *i.e.,* delay/job size - perhaps a more representative measure of network utility for data transfers[1]. Re-examining the design objectives underlying bandwidth sharing on today's networks leads to further questions. What happens to user perceived performance when the system is overloaded? Is a graceful degradation achieved? How do various bandwidth sharing mechanisms fare when users are impatient?

This paper is organized as follows. In §II we briefly present two bandwidth sharing policies, *i.e.,* fair sharing and size-based differentiation, which we evaluate over a range of loads when users are 'not' impatient. We intend to exhibit performance degradation incurred by the two policies when the system is moving from an underloaded to overloaded regime. The rationale for doing so is that given the difficulties in traffic modeling, planning, and dimensioning of links for data transfer networks, resources will see 'temporarily' congestion albeit infrequently. In §III we present two generic models capturing a wide diversity of plausible user impatience behaviors. Note that in practice user behavior can be very complex. Our goal is to achieve a better understanding of how the characteristics of user impatience impact the performance achieved by the two types of bandwidth sharing schemes. After identifying several performance metrics for evaluating systems with aborted transfers, we will provide a detailed discussion accompanied by simulation results in §V. Concluding remarks are given in §VI, including an extension of our models to the case where each user's perception of performance is associated with transferring a 'cluster' of jobs, as might correspond to a single web page access.

II. Bandwidth Sharing: Moving from Underloaded to Overloaded Regime

We model each data transfer as a fluid flow with a known finite volume when the transfer is initiated. To capture the performance during transfer, we assume all requests are granted, *i.e.,* no incorrect address or denied access, and the response time from initiation to the time the server starts transferring is zero[2]. Once a transfer is initiated, it contends with on-going flows on a fixed set of network resources throughout its lifetime, *i.e.,* fixed routing until completion or the user aborts his transfer. To capture the fact that users with larger files are likely to be less sensitive to transfer delays, we choose the *'bit transmission delay (BTD)'* as the primary performance measure of interest for data transfers, where the BTD for a transfer is given by delay/file size [7], [8].

For analysis purposes, we will consider the fair sharing (FS) and size-based differentiation (SD) policies on a single bottleneck link in this paper, *i.e.,* all data transfers are contending for a single resource capacity. Note that this is a reasonable assumption considering that the bottleneck of data transfers typ-

This work is supported by NSF Career Award NCR 96-24230, a grant from Tivoli Corp. Austin, and an Intel Technology for Education 2000 equipment grant.

[1]We will discuss this further in §II.

[2]In practice the connection setup time also impacts user impatience.

ically is at the edge of the network, *e.g.,* access routers, and assuming that one can neglect other factors, such as the heterogeneous round-trip delay, impacting the bandwidth allocation. For the single link case, the various FS policies considered in the literature essentially correspond to providing equal share of the link capacity to all ongoing flows. By contrast, the SD policy gives priority to the flows that have small 'residual work' to complete. The key of SD is to exploit the range of user tolerances to bandwidth or delay in order to benefit the whole. By speeding up small transfers, large ones only see a negligible performance degradation resulting in a better overall performance. We model SD type of policies as a weighted processor sharing discipline with the weight of a flow proportional to $\exp(-p(t))$ where $p(t)$ is the remaining size to transfer at time t, as suggested in [7]. Note that the extreme case of SD is so called Shortest Remaining Processing Time first (SRPT) discipline where only the file with the smallest remaining size among all ongoing ones will be served at any given time.

Before examining the interaction of user impatience with the two types of policies, we consider how FS and SD perform when moving from an underloaded to heavy or even overloaded regime assuming no aborted transfers. We conduct simulations for the case of a single link with capacity 1 Mbps under a range of traffic loads. We assume that the transfer flows arrive according to Poisson processes and the file size distribution of the transfers is bounded Pareto with mean 5 Kbytes. Fig.1 shows the average BTD (on a logarithmic scale) perceived by the sets of jobs with increasing size. We divided job sizes into 4 bins, where the first bin are the jobs that have size in the interval $[10^3, 10^4)$ bits, the second in $[10^4, 10^5)$, and so on. Note that the results shown for the overloaded cases are 'transient' in the sense that they are collected on a finite event simulation for an unstable system. As seen, when moving from the underloaded to the overloaded regime, SD maintains good performance for small to medium size jobs (graph at bottom), while FS degrades performance 'uniformly' for all jobs (graph at top). Notice that since most transfers are small in size, as suggested in by, *e.g.,* [2], [9] SD will benefit the majority of demands (98% of jobs fall into the first three bins in our simulated case) while incurring comparable performance degradation to FS for the very few large jobs. Furthermore, if users were allowed to abort transfers due to poor performance, the 'effective' traffic load might be 'gracefully' reduced and thus produce an even better performance for the remaining users.

III. Modeling User Impatience

In this section we will propose two generic models that capture a range of plausible user impatience behaviors. There are two types of user sensitivity to transfer performance: (1) a concern with the received cumulative service, *i.e.,* how much work has been completed since the transfer is initiated, and (2) a concern with marginal progress, *i.e.,* how much work is completed over the past u time units. Without loss of generality, suppose a transfer is initiated at time 0. Let $w(s,t]$ denote the cumulative

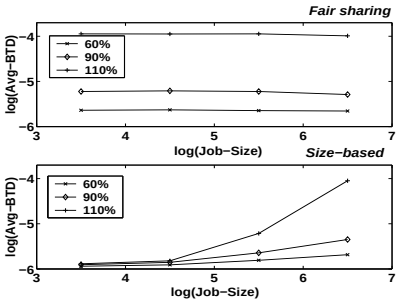

Fig. 1. Average BTD for different size flows under FS (top) and SD (bottom) in the underload (60%), heavy-load (90%), and overload (110%) regime.

work that is completed for a transfer during the time interval $(s,t]$. We define the following two models of user impatience.

Definition 1: We call $e_c(t)$ a *minimum cumulative service* (MCS) curve for a user if it represents the minimum amount of work that needs to be completed after t units of time since a transfer was initiated. That is, such a user will abort his transfer at time $t > 0$ if and only if $w(0,t] < e_c(t)$.

Definition 2: We call $e_p(u)$ a *minimum progress service* (MPS) curve for a user if it represents the minimum amount of work that needs to be done during any time interval of length u during the transfer. That is, such user will abort his transfer at time $t > 0$ if and only if $w(s,t] < e_p(t-s)$, for some $0 < s < t$.

The key difference between the two models is that the MPS curve captures a user's 'time-invariant' expectation of perceived performance, and thus can be used to evaluate the transfer progress at shorter time scales from the current time to the past, while only the largest time window $(0,t]$ is used by the MCS users. For convenience we refer $e_c(t)/t$ as the minimum expected 'cumulative throughput' at time t, and $e_p(u)/u$ as the minimum expected 'transfer rate' at time scale u. Fig.2 shows an example of how a user might evaluate his transfer performance based on MCS and MPS curves. Let p denote the total size of the transfer. We consider two service curves which have the exact same shape but different meanings. We let $e_c(t) = rt$ and $e_p(u) = ru$. This means that the user is expected to have a constant minimum cumulative throughput for the MCS case and a constant minimum transfer rate (the slope of $w(0,t]$) for the MPS case. Observe that the transfer completes for the MCS case since the cumulative work $w(0,t]$ always stays above $e_c(t) = rt$. By contrast, the MPS curve imposes a more stringent constraint and thus the transfer will be aborted when the user perceives a slower transfer rate than r.

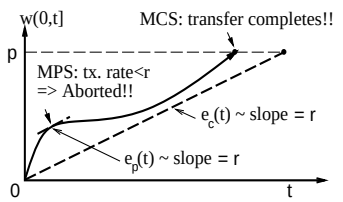

Fig. 2. Ex: evaluate transfer performance based on MCS and MPS curves.

In general $e_c(t)$ and $e_p(u)$ can be any non-decreasing function with respect to the 'elapsed time' t and the 'evaluation time window' u, respectively. Fig.3 shows several characteristic MCS (top: (a)-(c)) and MPS (bottom: (d)-(f)) curves. The service curves shown in Fig.3 (a) and (d) are the ones we considered in the previous example. Note that for these two cases, users evaluate performance from the very beginning of the transfer. Furthermore, for the case shown in (d), it is assumed that the user can monitor the 'instantaneous' transmission rate. Typically, however, users may be patient at the very beginning of a transfer, *i.e.*, there is some 'grace period' before users start to evaluate performance. Drawing on an analogy from the leaky bucket constraint [10], we can introduce such grace period by letting $e_c(t) = rt - \sigma$ (MCS) and $e_c(u) = ru - \sigma$ (MPS) - see Fig.3 (b) and (e). Note that for the MPS curves, the introduction of σ not only provides an 'initial' grace period but also a 'time scale' over which users evaluate the transfer rate. This is a more reasonable assumption than that used for the user behavior exhibited in (d). One can further generalize the functions to the case where a user wishes to evaluate his transfer rate at multiple time scales, see, *e.g.*, Fig.3 (f).

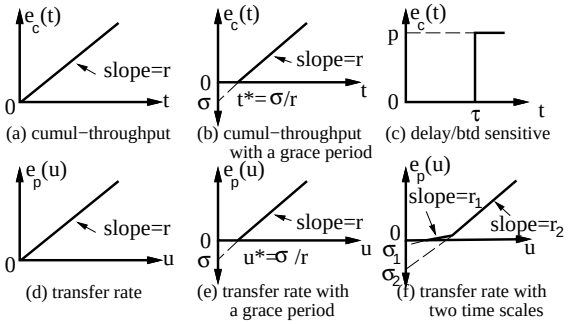

Fig. 3. Examples of MCS (top:(a)-(c)) and MPS (bottom:(d)-(f)) curves.

Another type of user may not be concerned with how his transfer progresses. Instead, he may only wait for at most τ units of time for his transfer to complete, as shown in Fig.3 (c). Note that a user who exhibits such impatience behavior is more 'elastic' than those shown in (a) or (b), given that $\tau = p/r$ where p is the size of the transfer, in the sense that he has higher flexibility in allocating bandwidth during the transfer. When the value of τ is independent of the size of the transfer, we call such users (fixed) delay sensitive. Alternatively, the user may be aware of that how long he might be waiting depends on the size of his transfer. To model such cases, one can set $\tau = p/r$ with a fixed r, which implies that the user expects a maximum BTD of $1/r$ regardless of the transfer size. We call such users BTD sensitive. Similarly, other parameters, such as σ, can also depend on the file size to reflect that the user may evaluate his cumulative throughput or transfer rate less frequently if the size is larger.

As a final note, we emphasize that a user's impatience is complex and could be a combination of the behaviors discussed above. Furthermore, it may change over time, based on the type of document that is being transferred, etc. Our attempt is to characterize a broad collection of impatience behaviors so as to assess their impact on system performance.

IV. Performance Metrics with Aborted Transfers

Before we discuss the interaction between user impatience behavior and bandwidth allocation policies, we shall first identify the metrics one may use to evaluate system as well as user perceived performance when users exhibit impatience behavior. The fact that users may abort their transfers due to unsatisfactory performance may result in mixed consequences. On the one hand, the aborted transfers may be translated to denied service, and the work that was done for those aborted transfers is wasted[3] and thus contributes to what one may call 'badput'. On the other hand, with some transfers leaving the system prior to completion, the effective traffic load is reduced and thus the rest of the transfers may see a reasonable performance even when the system is overloaded. Table I exhibits several metrics that we will use to evaluate both user perceived performance and system efficiency.

TABLE I

Metrics when users abort transfers

Metrics	Description
completion rate	number of completed transfers per second
incomplete rate	number of aborted transfers per second
goodput	rate of completed work in bits per second
badput	rate of transferred work for incomplete transfers
residual work	rate of transferred work for incomplete transfers
AvgBTD completed	Average BTD perceived by completed transfers

V. Bandwidth Sharing vs. User Impatience

This section investigates, via simulation, how FS and SD perform when users exhibit different impatience behaviors, *i.e.*, the ones we discussed in §III. We examine various scenarios wherein all users have the same impatience behavior. We will once again consider the single link case described in §II.

We will begin by considering users who are sensitive to cumulative service. Fig.4 shows the system performance achieved by SD and FS under four MCS type of behaviors. The parameters used for each case are shown on top of the figures. We plot the average completion and incomplete rate on the left and the goodput, badput, and residual work per second on the right for each behavior. The results for the FS and SD cases are shown in adjacent bars (FS: left, SD: right).

Observe first that for these behaviors, SD performs mostly better than FS except for case (a), where users are sensitive to the cumulative throughput of 50 Kbps with a zero grace period, and in terms of goodput. The reason is that without the initial grace period, large transfers may be discontinued early on, *e.g.*, right after initiation, under SD when small ones are also present. These large files, although only few in number, contribute a large portion of the total work, hence a reduction in goodput. Note however that when the large flows stay in the

[3]Researchers have proposed ways to re-use such partially transferred work by, *e.g.*, caching schemes. We however assume such work will be discarded.

system, they may see a similar performance as they would have seen under FS - recall our results shown in Fig.1. In fact, the zero grace period is not only unreasonable for users to evaluate throughput but also limits the SD's flexibility in allocating bandwidth to various size transfers. Now if we add a 1 second grace period before users start evaluating throughput, SD not only catches up but further outperforms FS upon system overloads - see Fig.4 (b). Meanwhile, with the inclusion of a grace period, both FS and SD allow almost every job to complete, except some very large ones in the overloaded regime to reduce the actual traffic load (to be below 1 Mbps). The difference in case (b) between FS and SD is that upon aborting large transfers, a larger portion of those files has been transferred under FS, resulting in a more significant badput. In fact this larger badput phenomenon under FS applies to all cases we explored below. This suggests that for most cases, SD achieves a more efficient utilization of resources.

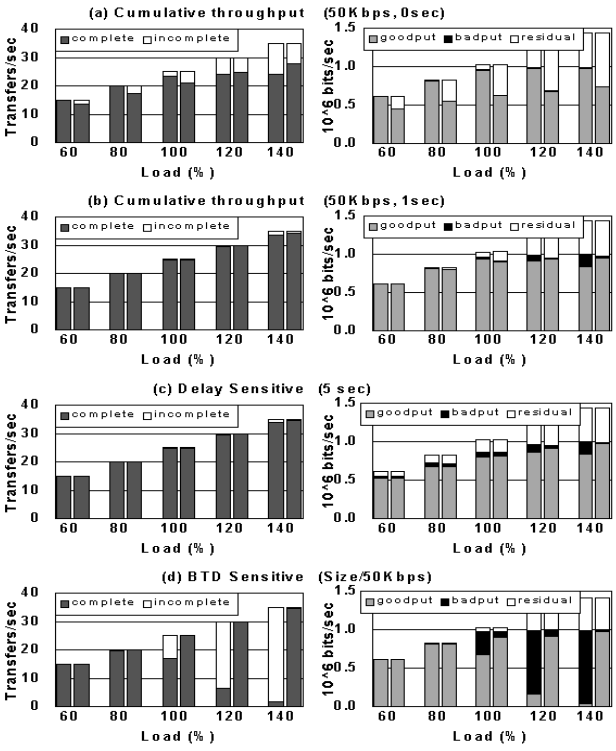

Fig. 4. Performance under FS and SD with MCS curve sensitive users.

Similar to having a grace period, users who are sensitive to delays also allow SD to be flexible in allocating bandwidth. As seen, results shown in in Fig.4 (c) are similar to those in (b), except less goodput is incurred for the case of delay sensitive users. This is due to the 5 second limit and 1 Mbps capacity which makes it impossible to complete transferring files of size larger than 5 Mbits. Note that the choice of a 1 second grace period for case (b), which allows most small transfers to complete as the 5 second delay constraint does, reflects that users who are transferring small files are not in a position to assess the 'cumulative throughput'. Fig.4 (d) further shows an example for the case when users' delay constraint depends on file size - constraint equals to file-size/50 Kbps. One can easily see that while SD maintains a very good system performance, FS performs poorly in the overload regime. This is because by providing an equal share of resources to the transfers, almost every flow fails to complete before its expected size-dependent delay, and, furthermore, upon abortion a good portion of the file has been transferred, resulting in a very poor badput. In other words, under FS the transfers will suffer from a 'uniform' degradation of performance.

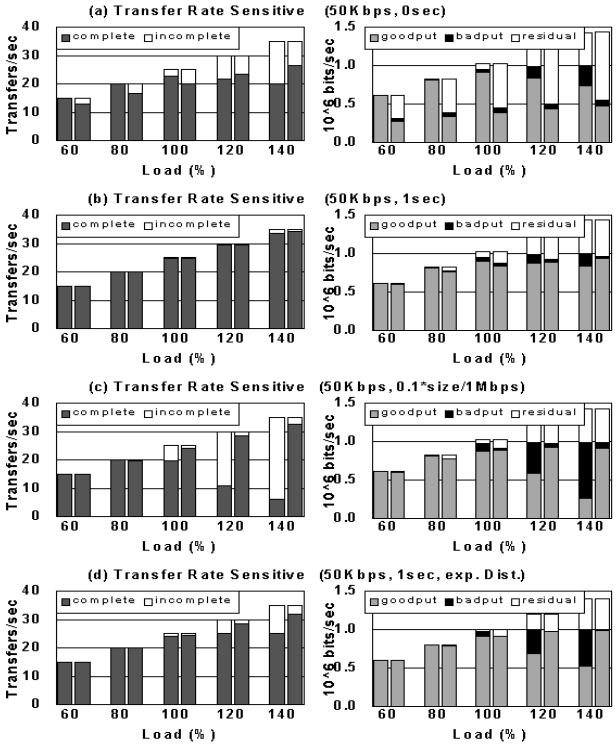

Fig. 5. Performance under FS and SD with MPS curve sensitive users.

Next we turn our focus to users who are sensitive to the marginal progress being made, *i.e.*, those modeled by MPS curves. We consider users whose minimum expected transfer rate is 50 Kbps, but at different time scales and under different circumstances. Paralleling the previous comparisons, we plot the system performance achieved by FS and SD in Fig.5. The results shown in Fig.5 (a) are for the case where users have a minimum 'instantaneous' transfer rate requirement, *i.e.*, zero grace period for evaluating the transfer rate. These results are similar to those in Fig.4 (a), but with less goodput and more severe badput. This is due to the fact that the transfer rate constraint is more stringent than the cumulative throughput. Again as we introduce a 1 second time scale for users to evaluate the transfer rate, the system performance can be brought back to a reasonable level, as shown in Fig.4 (b). We further consider the case where the time scale to evaluate performance is, instead of fixed for all transfers, proportional to the file size, *i.e.*,

users who transfer larger files will evaluate their transfer rates less frequently. The results for this case are shown in Fig.5 (c). Interestingly, they are similar to those of BTD sensitive users, *i.e.*, the system performance degrades dramatically in terms of all metrics under FS, but not for the SD case.

The last scenario we explored for the case of transfer rate sensitive users is the impact of file size distributions on performance. The results shown in Fig.5 (d) were obtained with the same parameters used for those in (b), but instead of bounded Pareto size distribution an exponential distribution with the same mean size is used. As seen the performance degrades under FS. We believe this is due to the fact that exponential distribution results in more similar mid-size file sizes than the bounded Pareto case. In turn, more transfers suffer a uniform performance degradation and thus are unable to achieve the minimum transfer rate.

Overall these results exhibit the performance impacts of various user impatience behaviors from the system's point of view. From the users' point of view, it may be important to maintain a good average BTD for completed transfers. Fig.6 shows the average BTD performance of completed transfers achieved under FS and SD for all impatience behaviors discussed above. As predicted, SD reduces the average BTD ranging from 1/3rd to less than 1/10th of that under FS when traffic load increases. Note that although for few impatience behaviors SD performs worse than FS in terms of goodput, the completed transfers indeed see an order of magnitude better performance in terms of BTD when the system is heavily loaded.

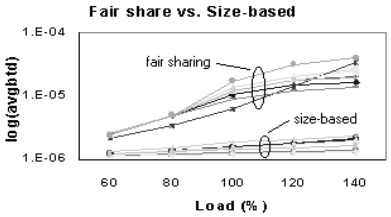

Fig. 6. Average BTD for completed transfers under FS and SD.

VI. Concluding Remarks

Our study addressed an important yet usually neglected question: how users' reaction to transfer performance impacts the design of bandwidth sharing schemes. We found that with most characteristic user impatience behaviors, SD is more effective at reducing the traffic load than FS when the system is heavily or overloaded, and thus leads to a better network efficiency as well as user perceived performance. As network resources can be periodically, but temporarily, overloaded with data transfers, our approach ensures that users will not perceive such phenomenon too badly.

Our approach can be further extend to the case where each user's perception of performance is associated with transferring 'a cluster of files', as opposed to 'individual' files. This models certain data transfer applications, such as web browsing, where each user access, *e.g.*, a web page, may contain several simultaneous[4] transfers of files. One possible user perception of performance for an access, or a cluster of files, may be based on the 'completely transferred work' associated with the whole cluster, *i.e.*, the total size of files within the cluster that have been completed so far. Simulations were also conducted for this scenario with various user impatience behaviors and the two bandwidth sharing schemes as in §V. Due to the space limit, we present a representative set of results in Fig.7. As seen, SD performs better than FS for all traffic loads and for all performance metrics even for the case of zero grace period, recall Fig.4 (a). This observation extends to other user behaviors considered previously, and suggests that SD is more beneficial when users evaluate performance on a higher level where transfers are correlated.

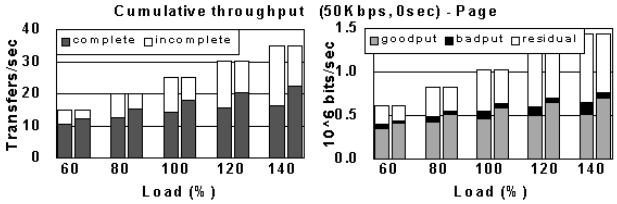

Fig. 7. Performance achieved by FS and SD for the batch arrival case.

Our final discussion above suggests that a possible research direction is to consider and validate experimentally the existence of high level or application specific user impatience models. Moreover, one may view user impatience models as criteria for user self-admission control [4], [8]. Thus the results shown in this paper exemplify the possible impact such mechanisms might have on system performance.

References

[1] A. Feldmann et.al., "Performance of web proxy caching in heterogeneous bandwidth environment," *IEEE INFOCOM'99*, vol. 1, pp. 107–116.

[2] M. Arlitt and C. L. Williamson, "Web server workload characterization: the search for invariants," *ACM SIGMETRICS*, 1996.

[3] William E. Moen, "An evaluation of the federal government's implementation of the government information locator service (gils)," 1997, available via http://www.unt.edu/wmoen/publications/gilseval/titpag.htm.

[4] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan, "Rate control in communication network: shadow prices, proportional fairness, and stability," *JORS*, vol. 49, pp. 237–255, 1998.

[5] L. Massoulié and J. Roberts, "Bandwidth sharing: objectives and algorithms," *IEEE INFOCOM'99*, vol. 3, pp. 1395–1403.

[6] J. Mo and J. Walrand, "Fair end-to-end window based congestion control," *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 556–567, 2000.

[7] S.-C. Yang and G. de Veciana, "Size-based adaptive bandwidth allocation: Optimizing the average QoS for elastic flows," *submitted to INFOCOM'02*.

[8] J. Roberts and L. Massoulié, "Bandwidth sharing and admission control for elastic traffic," *ITC Seminar, Yokohama*, 1998.

[9] M. Crovella, M. Taqqu, and A. Bestavros, "Heavy-tailed probability distributions in the world wide web," *A Practical Guide To Heavy Tails, Chapman & Hall, New York*, pp. 3–26, 1998.

[10] M. Butto, E. Cavallero, and A. Tonietti, "Effectiveness of the leaky bucket policing mechanism in ATM networks," *IEEE JSAC*, vol. 9, no. 3, pp. 335–342, 1991.

[4]Modern web browser initiates 'threads' to download remote files while reading a main page. One may view those threads as simultaneous transfers.