Enhancing Both Network and User Performance for Networks Supporting Best Effort Traffic

Shanchieh Jay Yang, Gustavo de Veciana

Abstract—**With a view on improving *user perceived* performance on networks supporting best effort flows, *e.g.,* multimedia/data file transfers, we propose a family of bandwidth allocation criteria that depends on the residual work of on-going transfers. Analysis and simulations show that allocating bandwidth in this fashion can significantly improve the user perceived delay, bit transmission delay, and throughput over traditional approaches, *e.g.,* by 58% on an 80% loaded linear network. A simple implementation based on TCP Reno, exemplifies how one might approach practically realizing such gains. We discuss several other advantages of incorporating such differentiation at the transport level. In particular we make the case that favoring small transfers combined with user impatience or peak rate constraints, both of which are natural mechanisms for users to express the utility of completing transfers, offers a lightweight approach to achieving good overall network goodput and/or utility for best effort networks.**

I. Introduction

In this paper we consider the question of how to optimize both network and user perceived performance for 'best effort' services, *e.g.,* file transfers mediated by TCP. Given that a large fraction (*e.g.,* 90% [1]) of the volume on today's Internet corresponds to this type of traffic, it is indeed surprising that this problem has not been more prominently addressed. At first glance the question of enhancing the performance seen by best effort flows may seem paradoxical, in fact it is not. A key property of a typical best effort transfer, is that it is malleable in the sense that 'instantaneous' throughput variations throughout the transfer are tolerable, as long as the *flow level* performance is adequate, *e.g.,* delay, bit transmission delay (BTD) (delay/size), or perceived throughput (size/delay). This tolerance to instantaneous throughput variations, underlies the traditional notion of 'best effort' where the emphasis is placed on adapting the user's instantaneous transmission rate (window) to share, possibly time varying, available resources 'fairly'. However a more natural design objective would be to optimize for the users' perceived performance rather than an artificial notion of fairness. Keeping with the spirit of 'best effort' performance it makes sense to design networks so as to optimize the *average* performance seen by users rather than making individual guarantees. Thus both from a network's and users' points of view, designing mechanisms that achieve enhancements in the average delay, BTD, etc., for best effort networks makes good sense.

A. Is overprovisioning a reasonable answer to performance problems?

A common, not unreasonable argument is often made that one can enhance user perceived performance by simply overprovisioning the network. However, 'efficient' overprovisioning of heterogeneous IP-based networks is itself not a simple task, requiring, among other things, fairly predictable traffic and growth models. Unfortunately recent experience has shown that even aggregated data traffic may exhibit long-range dependence, non-stationarity, and bursty growth on longer time scales, *i.e.,* is hard to model. Moreover, even if a single carrier could overcome this problem, the heterogeneity of the current inter-networking infrastructure would make coordinating inter-carrier agreements to avoid mismatches exceedingly difficult. Just as "nature abhors a vacuum", one should recognize that highly overprovisioned resources will quickly be filled, with possibly unwanted traffic. Thus even in the *unlikely* scenario where an institution/provider is willing to pay to overprovision their network infrastructure, this would need to be supplemented by fairly sophisticated monitoring tools to ensure that the benefits are indeed seen by their most valued customers. Moreover, given that resources are likely to become overloaded, even occasionally, questions concerning user behavior upon overload need to be faced. For example, a recent study [2] claims that 11% of the TCP mediated files transfers (to a given server) were interrupted, corresponding to an excess of 20% of the total traffic volume, *i.e.,* 20% of the transmitted data went to waste. Together, these arguments suggest that the costly capacity, infrastructure, and technology commitments required to achieve better performance for best effort flows are not likely to be based on overprovisioning alone.

B. Enhancing performance through better network protocols

In order to achieve a more efficient use of network resources, we propose to exploit the malleability of best effort traffic. In particular, we suggest allocating more 'instantaneous' bandwidth to flows with smaller residual size. Such differentiation, although seemingly 'unfair' at a given time instance, in fact benefits the whole over time, *i.e.,* enhances the *average* 'flow level' quality of service (QoS), *e.g.,* delay, BTD, or perceived throughput. In fact as will be discussed in the sequel, for heavy tailed flow size distributions, this will not only improve the average, but also tend to improve the performance seen by the majority of flow sizes except for the very largest [3], [4]. The key idea is that by speeding up flows with small residual work to be done, one expedites short transfers without necessarily compromising the large ones. On a general network, in addition to differentiating based on the residual flow sizes, one needs to account for the degree of service parallelism that can be achieved, *e.g.,* giving priority to multiple transfers on short routes versus 'small' transfers on longer routes. The question then is how to make the most out of limited network resources while achieving high concurrency. In this paper, we will start by examining the single (bottleneck) link case and then progress to study the network case, where we propose a general class of dynamic bandwidth allocation schemes - see §III. The proposed scheme aims at exploring tradeoffs between favoring small flows and achieving high degrees of parallelism in allocating bandwidth to flows. We then briefly illustrate a simple implementation at the transport layer, *i.e.,* a modification of TCP, and discuss representative

This work is supported by NSF Career Award NCR 96-24230, by a grant from Tivoli Corp. Austin, and an Intel Technology for Education 2000 equipment grant. This paper is an extended version of work published in the Proceedings of IEEE INFOCOM 2002.

packet level simulations in §IV.

While our analysis in §III focuses on optimizing the average BTD[1], we take a step further to investigate the performance impact of our proposed size-based differentiation scheme from both the user's and network/service provider's perspective in §V. In particular, we consider network provisioning, user impatience behavior upon system overloads, and scenarios where flows are peak rate constrained. Note that both users' impatience and access peak rate constraints are natural mechanisms for users (of best effort service) to express the utility of completing transfers. We will make the case that favoring smaller transfers combined with these two natural mechanisms offers a lightweight approach to achieving good overall network utility.

There are various approaches to achieving size-based differentiation. The focus of this paper is on bandwidth allocation, as realized by transport level protocols. This provides a robust mechanism to achieve differentiation among flows sharing bottlenecked resources, wherever those might be in the network. In §VI we conclude this paper by offering some remarks on other opportunities for size based differentiation, *e.g.,* at the server side [5], within web browsers, within edge and core routers [6], or even at the network level through routing [7], [8], [10], [9]. In the next section we provide a formal description of the dynamic bandwidth allocation problem and review some of the related work in this area.

II. Problem Description and Related Work

Traditional research on bandwidth allocation for networks supporting best effort transport, *e.g.,* TCP or ABR service, has for the most part focused on considering a *static regime, i.e.,* the bandwidth allocation depends on the *fixed* number of flows on each route. The focus has been on possible bandwidth allocation criteria, *e.g.,* max-min, proportionally, or potential delay fairness, and devising stable decentralized mechanisms to achieve such allocations, see *e.g.,* [11], [12], [13], [14], [15], [16]. We shall refer to these as the 'traditional' fairness criteria. The basic insight underlying this body of work is to associate a (concave) utility function (of the allocated bandwidth) to each user and to devise mechanisms which optimize the overall network utility subject to capacity constraints. Interestingly, one can attempt to reverse engineer the Internet, by inferring the *implied utility* functions associated with current transport mechanisms, see *e.g.,* [17]. Overall this research lacks a clear articulation of the impact that the bandwidth allocation criterion, *i.e.,* the choice of utility function, has on flow-level performance in a *dynamic regime* where flows come and go.

This paper focuses on the dynamic bandwidth allocation problem. That is, determining the possibly 'time varying' bandwidth to allocate to a set of flows subsequent to their arrival to the network such that the overall performance is optimized. We consider a network consisting of a set of links L where link $l \in L$ has capacity c_l bps. Each file transfer $j \in J$ is modeled as a fluid flow with a known initial volume of p_j bits of data to send. Upon arrival/initiation at time a_j, flow j is assigned a route denoted by $r_j \in R$ and contends for bandwidth on the links along its route.[2] The set of links traversed by route r is captured by a 0-1 matrix A where A_{lr} is 1 if route r traverses link l and 0 otherwise. We let $x_j = (x_j(t),\ t \geq 0)$ denote the bandwidth allocated to flow j as a function of time. We assume without loss of generality that it is zero prior to a flow's arrival and after it completes. The time to complete flow j, *i.e.,* its transfer delay, is denoted by d_j and depends on its size p_j and the possibly time varying bandwidth the flow is allocated. For the most part in this paper, we will focus on the bit transmission delay (BTD) as the performance measure of interest where the BTD for flow j is given by $b_j = d_j/p_j$. Notice that from a flow's point of view, minimizing the BTD is equivalent to maximizing the throughput. We summarize the notation in Table I and formally define the problem of interest as follows.

Problem 1 (Dynamic Bandwidth Allocation)

$$\begin{array}{ll}
\text{min} & \displaystyle\frac{1}{|J|}\sum_{j\in J} b_j = \frac{1}{|J|}\sum_{j\in J}\frac{d_j}{p_j}, \\
\text{over} & \mathbf{x} = (x_j : \mathbf{R}_+ \to \mathbf{R}_+,\ j \in J), \\
\text{s.t.} & \displaystyle p_j = \int_{a_j}^{a_j+d_j} x_j(\tau)d\tau,\ \ \forall j \in J, \\
 & \displaystyle\sum_{j\in J} A_{lr_j} x_j(t) \leq c_l,\ \ \forall t \geq 0,\ l \in L.
\end{array}$$

TABLE I

Summary of Notation

Set of Links: L	capacities $c_l, l \in L$
Set of Routes: R	link-route incidence matrix A_{lr}
Set of Jobs: J	(a_j, p_j, r_j) for flow j
	(arrival time, size, route)
Bandwidth Allocation	$\mathbf{x} = (x_j(t),\ t \geq 0,\ j \in J)$
Performance Metrics	d_j, $b_j = d_j/p_j$, $\frac{1}{|J|}\sum_{j\in J} d_j/p_j$
	delay, BTD, avg BTD

Notice that the problem stated above aims at minimizing the average BTD for a 'finite' set of jobs J with *known* arrival times - this is the so called *off-line* regime, which serves to identify the best one could do. In practice future arrival times would not be known whence only allocation policies that depend on past events, *i.e., on-line* policies are permissible. If further the arrivals and flow sizes were modeled by stationary stochastic processes, one can consider optimizing the customer average BTD over stationary causal policies, *i.e.,* minimize $\mathbb{E}[B] = \lim_{|J|\to\infty} \frac{1}{|J|}\sum_{j\in J} B_j$, where B denotes the typical BTD experienced under a stationary dynamic bandwidth allocation policy. We refer to bandwidth allocation policies that minimize the average BTD as BTD-optimal.

The authors of [18], [19], [20], [21], [22] have considered stochastic models to capture the dynamic behavior of *existing* network mechanisms, *e.g.,* TCP and traditional fairness criteria. One lesson from this body of work is that, even for a given bandwidth allocation policy, it is difficult to analytically model the performance seen by users in a dynamic network setting, except for specially structured topologies. To our knowledge,

[1]We focus on the BTD metric since it captures a reasonable user expectation that larger files takes longer to complete.

[2]For simplicity we assume routes are stable, *i.e.,* for the most part flow associated with a given transfer follows the same route.

the only existing work that attempts to find a BTD-optimal policy at the network level was conducted in [23]. Their results however suggest that the problem is NP-hard unless one allows 'resource augmentation'. In fact, one can show that even for flows sharing a single link, there is *no* online algorithm that minimizes the average BTD [24]. One key idea drawn from the single link case is that the Shortest Remaining Processing Time first (SRPT) scheduling discipline performs well for the average delay as well as BTD [22], [24], [3]. However, to our knowledge, no systematic allocation criterion and associated transport mechanism have been proposed to enhance user perceived performance on a *network*.

Due to the inherent difficulty of the online version of this problem, in the sequel we will consider the 'transient' regime which is tractable. More specifically, we consider the set of ongoing flows $J(t)$ at time t where some of them may have been partially transferred, and the goal is to minimize the overall 'residual BTD', where the residual BTD of flow j at time t is defined as $b_j(t) = (f_j - t)/p_j$, and f_j denotes the time at which the transfer completes. Our investigation of the transient regime provides an avenue for determining *greedy* policies for the online case, where, at each point in time, bandwidth is allocated so as to complete the 'current' set of ongoing flows in a BTD-optimal manner.

III. Residual size based Differentiation

A. Sharing bandwidth on a single link

We begin by considering the case where flows contend for bandwidth on a single link. Despite its simplicity this model captures the scenario where a set of flows are constrained by the same bottleneck, *e.g.,* an access gateway, and offers basic insights that apply to the general network case.

A.1 Fair sharing

In the context of sharing a single link, traditional fairness criteria, *e.g.,* max-min and proportional fair, reduce to *fair sharing, i.e.,* each ongoing flow gets an equal share of the available bandwidth. A collection of TCP flows with the same round trip time would approximately realize an equal share of the capacity. For simplicity, we consider an idealization where each ongoing flow $j \in J(t)$ at time t is assigned a bandwidth $x_j(t) = c/n(t)$ where c is the link capacity and $n(t) = |J(t)|$ is the total number of ongoing flows at time t. We shall consider this to be our baseline bandwidth allocation policy for the single link case.

While 'fairness' has been discussed at length, policies that achieve fair shares do *not* necessarily achieve good user perceived performance. In particular one can prove that in the case of a single link, the average BTD (and delay) achieved by policies that share non-trivial amounts of capacity among multiple flows can always be improved.

Lemma 1: Consider a set of flows contending for capacity on a single link. Any bandwidth allocation policy that allocates positive bandwidths to more than one flow at a time is not BTD-optimal.

The proof, given in the appendix, relies on showing that one can always improve upon policies that share bandwidth among ongoing flows by 'speeding' up those that would complete earlier, *i.e.,* giving them the full link capacity, without penalizing any of the others. Note that the flows that will complete earlier are those with smaller sizes or residual work to be done. This suggests one might consider alternative bandwidth allocation policies that differentiate based on flows' (residual) sizes.

A.2 Size-dependent differentiation

A well known size-dependent scheduling, here viewed as a bandwidth allocation policy, is the Shortest Remaining Processing Time first (SRPT) discipline. Let $p_j(t)$ denote the residual work associated with flow j at time t. The SRPT policy assigns the full link capacity to a flow $j^* \in J(t)$ with the smallest residual work at time t, *i.e.,* $j^* \in \text{argmin}_{j\in J(t)}(p_j(t))$. SRPT is known to minimize the average *delay* for flows sharing a single link [25], and was recently shown to be 2-competitive for the average BTD metric [24]. With a view on further enhancing performance and developing allocation policies that can be implemented, below we propose two novel policies to realize size-dependent differentiation.

First we shall consider a policy that allocates the full capacity to the flow j^* having the smallest product of original and residual size, *i.e.,* $j^* = \text{argmin}_{i\in J(t)}(p_i \cdot p_i(t))$. We refer to this as the Shortest Processing Time Product first (SPTP) policy. The rationale for this can be easily seen by considering the case where two flows have the same residual size. In this case it should be clear that to minimize the BTD, one should favor the flow with the smallest original size. In fact one can show that SPTP is BTD-optimal in the transient regime.

Theorem 1: At each point in time the SPTP bandwidth allocation policy will minimize the overall residual BTD for ongoing flows sharing a single fixed capacity link if there are no additional arrivals.

A proof of this result is given in the appendix. One can think of SPTP as a greedy online policy in the sense that it always seeks to do the best for the flows that are currently active, *i.e.,* empty the system in a manner that incurs a minimum overall residual BTD. We will show via simulation that SPTP marginally outperforms SRPT with respect to the average BTD. We later revisit this policy when we consider a network scenario.

Recognizing that allocating bandwidth based on SRPT and SPTP will be difficult in a decentralized framework, we propose a second class of policies whereby each active flow j has an associated size-dependent weight $w(p_j, p_j(t))$, and bandwidth is allocated in proportion to these weights. Thus by appropriately selecting weight functions that are decreasing in the residual size, *e.g.,* $w_j(t) = \exp(-\alpha p_j(t))$ where $\alpha > 0$, one may approximate the SRPT discipline as $\alpha \to \infty$. Similarly, SPTP can be approximated by using $w_j(t) = \exp(-\alpha\sqrt{p_j \cdot p_j(t)})$. We refer to these two size-dependent weighted fair sharing policies as Remaining Processing Time Weighted Sharing (RPT-WS) and Processing Time Product Weighted Sharing (PTP-WS). Although Lemma 1 suggests that one can always improve performance over policies that share bandwidth, such as RPT-WS and PTP-WS, we will show that they already achieve significant performance improvement over fair sharing while allowing flows to simultaneously make progress towards completion.

A.3 Performance gains of size-dependent differentiation

Recently [3] showed analytical bounds for the performance gains that can be achieved by SRPT versus fair sharing for heavy tailed flows. In this section we shall revisit this and briefly evaluate the three policies proposed above, *i.e.*, SPTP, RPT-WS, and PTP-WS, versus fair sharing via simulation.

Simulations were conducted for a 10 Mbps link shared by flows arriving according to Poisson processes with sizes selected from a bounded Pareto distribution with mean 5 KBytes. We will for the most part use this 'heavy-tailed' distribution to model flow sizes throughout the paper. Such distributions have been widely studied and empirically found to be representative of the file size on the Internet [26], [27], [28], [29]. Our results were however found to be robust to other size distributions. Fig.1 shows the average BTD performance *improvement* achieved by size-dependent policies over fair sharing. As can be seen the four size-dependent policies significantly outperform fair sharing with SPTP exhibiting the best average BTD. In these simulations we use a moderate value $\alpha = 1$ in RPT-WS and PTP-WS, but they already exhibit 30-60% improvements over fair sharing. Based on our experiments, the average BTD performance achieved by RPT-WS and PTP-WS improves quickly as one increases the value of α.

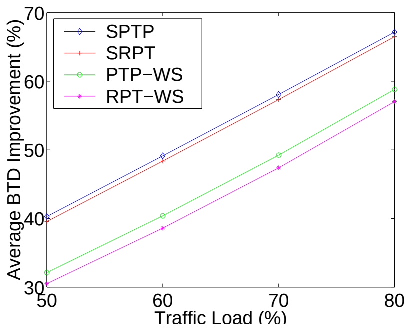

Fig. 1. Average BTD improvement for SRPT, SPTP, RPT-WS, and PTP-WS over fair sharing as traffic load increases.

In summary there are various bandwidth allocation approaches that favor small flows, and they tend to significantly reduce the average BTD for the single link case. A criticism brought against these SRPT-like polices is their potential to induce starvation for large flows. However [3] and [4] have argued and shown that this conclusion does not apply in the case where the flow sizes have a large variance - which is typical of files transferred over the Internet [26]. Our own experiments also confirm that starvation is indeed not a concern, particularly as compared to fair sharing. Refer to [7] for a more detailed discussion.

B. *Sharing bandwidth on networks*

Although the single link case suggests one should 'always' favor small flows in order to minimize the average BTD, this turns out not to be true in general. In the network case a flow with a small residual size may contend for bandwidth with multiple sets of flows on disjoint routes which could be served in parallel. This leads to a trade off between giving preferential treatment to flows with smaller residual size versus maximizing the service parallelism that can be achieved. Consider the symmetric linear network shown in Fig.2 including m links with the same capacity c, and m short and 1 long route. Even if there

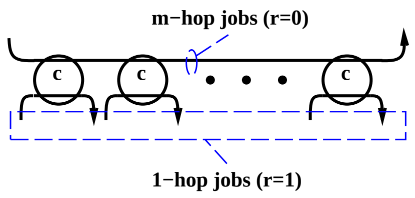

Fig. 2. Linear network: m equal capacity links.

were a flow with a small residual size on the long route, one may wish not to allocate the full capacity on all the links to it, since this would temporarily 'block' the concurrent service of flows on various short routes. In the sequel we will consider this linear network in more detail as a means to identify the characteristics that a 'good' bandwidth allocation might have.

B.1 Example: Symmetric Linear Network

To simplify our analysis, suppose that bandwidth allocation among flows sharing the same route is independent of the aggregate bandwidth allocated to that route, and follows the SPTP policy introduced in §III-A. That is, at any given time t, the aggregate route bandwidth $y_r(t)$ will be allocated to the flow that has the smallest product of original and residual size among all flows on route r. We shall refer to the SPTP discipline as our 'intra-route' bandwidth allocation policy, *i.e.*, dictating how bandwidth is allocated among flows sharing the same route. The question then is to identify the aggregate route bandwidths $(y_r(t),\ t \geq 0,\ r \in R)$ to allocate to each route, *i.e.*, a good 'inter-route' bandwidth allocation.

For succinctness we refer to the long route as the m-hop route and give it route index $r = 0$, while the set of short routes are referred to as 1-hop routes and indexed by $r = 1, 2, \cdots, m$. Since link capacities are equal, it should be clear that each 1-hop route can be allocated the same aggregate route bandwidth without compromising optimality, *i.e.*, c minus that allocated to the m-hop route. This means that we need only consider bandwidth allocations which give the same bandwidth to all 1-hop routes. In the sequel we let $y_1(t)$ denote the aggregate bandwidth allocated to any of the 1-hop routes, and $y_0(t)$ be that allocated to the m-hop route. This symmetry in the topology significantly simplifies the state space, and thus the analysis of interactions among routes. In particular, the dynamics on this network correspond to m-hop flows contending for a 'single bottleneck' resource of capacity c with 'all' flows on 1-hop routes, but where some of the 1-hop flows can be served in parallel. By analogy with Lemma 1 for flows sharing a single link, one can show a 'no-sharing' result for the symmetric linear network. The proof is similar to that of Lemma 1, and thus omitted in this paper due to limited space.

Lemma 2: A BTD-optimal inter-route bandwidth allocation $\mathbf{y}^*$ for flows on a symmetric linear network (see Fig.2) is such that at any time t either $y_0^*(t) = 0$ or $y_0^*(t) = c$.

Combining the necessary condition in Lemma 2 with the assumption that flows on the same route are allocated bandwidth according to the SPTP policy, below we determine a BTD-optimal inter-route bandwidth allocation policy for the linear network in the transient regime. More specifically, the policy determines whether to allocate the full capacity c to the m-hop route (or 1-hop route otherwise) at any time t such that the overall residual BTD is minimized assuming no arrivals after time t. Before presenting this last result we introduce some further notation. Without loss of generality, we shall separately index the $n_0(t)$ m-hop flows and all $n_1(t)$ 1-hop flows in the network at time t according to their finishing order assuming that they are served according to SPTP among flows on the same route. To distinguish the flows that belong to different route types, we use p_j^0 and p_j^1 to denote the original size of the jth m-hop and 1-hop flow to complete, respectively. Similar notation applies to the residual size. Furthermore, we define the cumulative residual work, $\tilde{p}_j^s(t) = \sum_{i \le j \ \& \ r_i = r_j} p_i^s(t)$, $s = 0, 1$, as the total residual work that needs to complete on the route associated with flow j prior to its completion. We assume ties are broken arbitrarily. Now we may present our policy.

Algorithm 1 (Greedy Algorithm for Linear Network) At any time t, $y_0^*(t) = c$ and $y_1^*(t) = 0$ if

$$\underbrace{\frac{1}{p_1^0} \cdot \frac{c}{\tilde{p}_1^0(t)}}_{\substack{\text{max-wgt-thruput} \\ (m\text{-hop route})}} \quad > \quad \underbrace{\max_{k=1,2,\cdots,n_1(t)} \left[\left(\frac{1}{k} \sum_{l=1}^{k} \frac{1}{p_l^1}\right) \cdot \left(\frac{k \cdot c}{\tilde{p}_k^1(t)}\right) \right]}_{\substack{\text{max-weighted-throughput} \\ (\text{1-hop routes})}} \qquad (1)$$

and $y_0^*(t) = 0$ and $y_1^*(t) = c$ otherwise.

Theorem 2: At each point in time Algorithm 1 minimizes the overall residual BTD for flows on the linear network shown in Fig.2, assuming that SPTP is the intra-route policy and there are no future arrivals.

In other words, Algorithm 1 is BTD-optimal for the symmetric linear network in the transient regime. Despite its lengthy proof (given in the appendix), (1) has a fairly simple interpretation. In deciding whether to allocate bandwidth to the long or short routes, one needs to assess which option will lead to the highest 'weighted throughput' considering various finite time windows into the future. More specifically, the throughput over a time window, measured in flows/sec, is given by the number of flows that complete service in that window. The weight is given by the average of the reciprocal sizes for the flows that complete service in the window under consideration. Intuitively, this weighting factor accounts for the impact that completing these flows has on BTD. Because of the possibility that one might achieve a higher weighted throughput by serving flows in parallel, one should not put excessive emphasis on favoring small flows traversing long routes when deciding inter-route bandwidth allocation.

B.2 Size-Based Adaptive Bandwidth Allocation

The above example shows the potential complexity of a BTD-optimal policy for the transient regime. Even for a simple toy network, one may need to account for the sizes of almost all flows (the 'smallest' m-hop flow and all 1-hop flows) to determine a bandwidth allocation that minimizes the overall residual BTD. We will show later via simulation that this policy exhibits excellent performance as a greedy online strategy for allocating bandwidth on a symmetric linear network. However it does require a centralized agent to coordinate across flows and routes to determine dynamic changes in the bandwidth allocation. As a step towards a more practical realization, below we propose a general class of bandwidth allocation criteria where per-user weights that depend on residual sizes are considered. Following [11], [13], [15], we define a class of size-dependent adaptive bandwidth allocations (SABA).

Definition 1: Let $J(t)$ denote the set of active flows at time t, and $p_j(t)$ be the residual size of flow $j \in J(t)$. A bandwidth allocation, $(\mathbf{x}^*(t), t \ge 0)$, is said to satisfy Size-based Adaptive Bandwidth Allocation (SABA) criterion if and only if at each time t,

$$\mathbf{x}^*(t) \;=\; \arg\max_{\mathbf{x}(t)} \sum_{j \in J(t)} w_j(t) \cdot U_\beta(x_j(t))$$

such that

$$\sum_{j \in J(t)} A_{lr_j} x_j(t) \le c, \;\; \forall\, l \in L,$$

where $w_j(t)$ is flow j's weighting function depending on the residual flow sizes at time t, and

$$U_\beta(x) \;=\; \begin{cases} \log x & \beta = 1, \\ (1-\beta)^{-1} \cdot x^{1-\beta} & \beta \ge 0 \text{ and } \beta \ne 1, \end{cases} \qquad (2)$$

is a utility function characterizing the sensitivity of a flow to its bandwidth allocation. The first order optimality condition can be written as follows: at any time t, $\mathbf{x}^*(t)$ is optimal if for any other *feasible* allocation $\mathbf{x}(t)$ we have that

$$\sum_{j \in J(t)} w_j(t) \frac{x_j(t) - x_j^*(t)}{(x_j^*(t))^\beta} \;\le\; 0.$$

Bandwidth allocations associated with maximizing the utility functions defined in (2) with *fixed* weights have been considered in [11], [13], [14], [15]. Notice that maximizing the overall user utility subject to resource constraints naturally favors flows that use fewer resources, therefore achieving higher service parallelism. Our premise here is to introduce per-flow weights that increase as the residual size of flow goes down. By appropriately selecting a dynamic residual-size dependent weight function and a concave utility function, SABA can trade off favoring small flows versus achieving high parallelism. In [7] we discussed in detail how the choices of weight functions impact SABA's performance. In this paper we follow the guidelines provided in [7] and consider a version of SABA employing reciprocal weight functions, *i.e.*, $w(p_j(t)) = (p_j(t))^{-\alpha}$.

We have conducted a variety of simulations to demonstrate the benefits of using SABA, see [7]. A representative result is exhibited in Fig.3, where we examine various bandwidth allocation policies on the afore-mentioned symmetric linear network. As expected, traditional fairness criteria achieve similar average BTD.[3] By contrast, SABA and the greedy optimal policy (Algorithm 1) significantly outperform those based on the traditional

[3]Note these results correspond to 'idealized' fluid-flow simulations where upon flow arrivals and departures bandwidth allocations are *re-computed* based on the appropriate criterion and then *frozen* during inter-event periods. This cuts down simulation time significantly.

fairness criteria - achieving improvements of up to 58% when the links are 80% loaded. The 'dynamic' residual size based weight function contributes to the success of SABA. With such weights even if discrimination among flows is small at a given point in time the dynamics are such that bandwidth allocation will be increasingly biased as flows progress towards completion. Hence if a given bandwidth allocation favors a small flow or achieves good parallelism, then this will be reflected in the relative increase in weights for the associated flows - letting these flows gain momentum and finish.

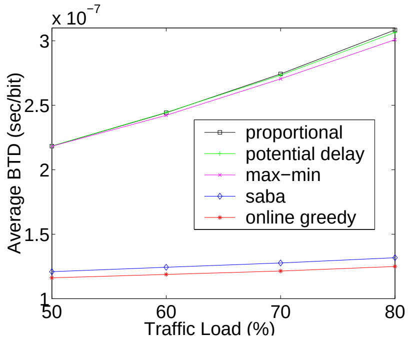

Fig. 3. The average BTD achieved by SABA, the greedy algorithm, and traditional fairness criteria on a 5-link linear network where each link has capacity 10 Mbps, the loads are symmetric, and the flow size distribution is bounded Pareto with mean 5 KBytes.

Note that, for the reciprocal weight shown above, increasing α increases the bias towards flows with a small residual flow size. By contrast increasing β, *i.e.*, changing the utility function, decreases both the extent to which parallelism is emphasized, *i.e.*, discrimination against flows which traverse multiple links, and reduces the impact of the weights[4]. The coupling between these parameters may be complex, but we have found performance be fairly robust with moderate choices of α and β. Other choices for the weight functions (*e.g.*, depending on all flow sizes sharing a route) permit a decoupling in the bandwidth allocation among routes versus how it is shared among flows sharing the same route. This decoupling can in turn lead to further performance gains - see [7].

IV. Realizing SABA

For the simple case proposed above where the weight function associated with a flow depends only on its own residual size, it is fairly straightforward to implement a decentralized mechanism that approximates SABA. Indeed, the end-system need only track the amount of data that remains to be transmitted, and modulate the 'aggressiveness' of TCP's *congestion avoidance* mechanism based on the residual flow size. A simple proof-of-concept implementation, TCP SAReno, is proposed in [7]. Let us briefly summarize how SAReno might be implemented by contrasting it with Reno. It requires a few changes on the sender side. An 'initial size' parameter is passed from the application layer to the SAReno transport layer, which will thereafter keeps track of the residual size of the transfer by tracking byte-wise acknowledgments. In each SAReno flow the 'additive increase rate' and 'multiplicative decrease ratio' is modulated based on the residual size of the transfer. To reduce frequent changes in these parameters we have quantized flow sizes into regions [5]. Note that if the initial flow size is not known, as would be the case for dynamic web contents, then a crude estimate would suffice.

TCP SAReno is compared against TCP Reno in [7]. In this case, packet-level simulations, including the intricacies of the real world, *e.g.*, round-trip times, the slow start phase, etc., give remarkable performance gains in terms of the average BTD, on the order of 30-40% on various networks. An interesting question is how SAReno and Reno flows might fare if they coexist on a network. Note that in such a scenario, Reno flows have essentially a 'constant aggressiveness' with a fixed linear increase rate and 50% multiplicative decrease ratio, while SAReno flows exhibit increasing aggressiveness as the residual flow size decreases. We conducted simulations to examine how the performance benefits would accrue as the penetration of SAReno flows increases. The simulations presented below concern a 6-branch star network.

We first consider a random penetration scenario, where the transfers mediated via SAReno rather than Reno were selected at random according to the penetration level being simulated. Fig.4 shows the normalized average BTD over all flows, for Reno flows only, and for SAReno flows, as the percentage of SAReno flows increases. They are normalized by the average BTD that would be achieved when all flows are mediated via Reno, *i.e.*, 0% SAReno flows. As seen, on average SAReno flows will see better performance (the normalized BTD is less than one) for all penetration levels. Moreover Reno flows will also see improved performance once the penetration of SAReno flows exceeds 20%. The fact that SAReno flows consistently see better performance than Reno flows suggests that users will have proper incentives to upgrade from Reno to SAReno.

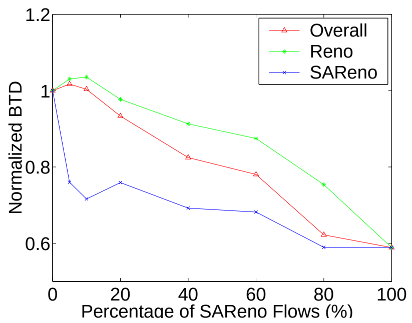

Fig. 4. Normalized BTD (over the case where all flows are Reno) as the percentage of (random) SAReno flows increases.

Alternatively SAReno might not be deployed in the homogeneous manner discussed above, but in a clustered manner corresponding to, say, access domains that adopt the new transport service. For example, one might have an increasing number of access domains that use SAReno. We examined the average BTD performance as one increases the number of access nodes on the star network that mediate transfers via SAReno. Fig.5 shows the normalized average BTD over all flows, Reno flows, and SAReno flows as the number of SAReno domain increases from 0 (all Reno) to 6 (all SAReno). Again the average BTDs

[4]For details see [15], [18], [7]

[5]Reader may refer to [7] for detailed values used for each quantized category.

are normalized by that achieved when all flows are mediated via Reno. One can observe that when one deploys SAReno on a per-domain basis, it has an even quicker impact than the homogeneous random deployment scenario - see Fig.4. This is to be expected since intra-route discrimination can be more effective when SAReno flows originate from the same access domain.

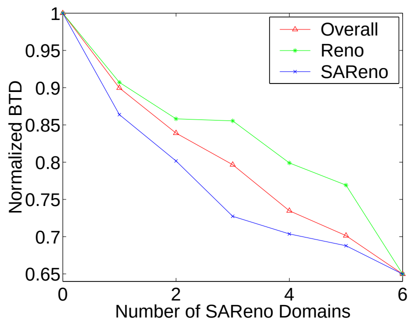

Fig. 5. Normalized BTD (over the case where all flows are Reno) when one increases the number of domains that sends SAReno flows.

V. Benefits beyond improving the average BTD

We have proposed a fairly 'straightforward' mechanism for residual size based differentiation (SD) at the transport level that achieves a significant average BTD improvement over traditional mechanisms, *i.e.*, fair sharing (FS). In this section we shall discuss various practical aspects that may limit, or enhance, the potential gains of SD over FS bandwidth allocation.

A. Network dimensioning

Suppose a network is provisioned to meet a given *average* QoS target, *e.g.*, BTD, then what would be the value of implementing SD over FS? One way to measure the gains is to evaluate the capacity savings, *i.e.*, the reduction in provisioned capacity to meet the same QoS target. However assuming a network provider has prior estimates for flow-level traffic loads and QoS targets for network dimensioning may be unreasonable. A complementary question would be: given a *fixed* capacity, how much more traffic load one can support on a network using SD over FS? We answer this question by examining two queues, M/GI/1-PS and M/GI/1-SRPT which model FS and SD, respectively. The difference in the traffic load the two queues can support while achieving the same average BTDs are shown in Fig.6. We plot two lines that connect points indicating the maximum load one can support under FS (y-axis) and SD (x-axis) when the target average BTD decreases. The two lines are associated with the flow size distributions being bounded Pareto and exponential, respectively. By comparing with the $x = y$ line, one can see the increase in the range of loads achieved by using SD versus FS. For example, in the case of the bounded Pareto distribution, SD offers a 30% increase in the traffic load that can be supported using FS for a given average BTD target. One can interpret this gain as revenue increase for a network service provider by either admitting more traffic or offering a more robust network that can tolerate larger traffic fluctuations.

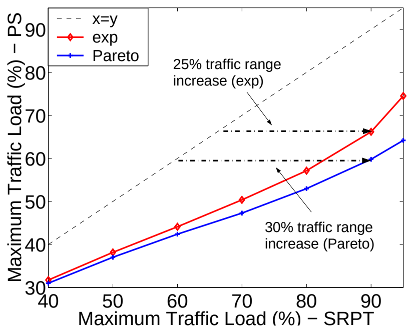

Fig. 6. Maximum traffic load one can support under M/GI/1-PS (*y*-axis) and M/GI/1-SRPT (*x*-axis) for same average BTD requirements.

B. What size flows benefit?

Given that size based differentiation is geared at speeding up smaller flows at the possible detriment of larger flows one might ask whether an improvement in the *average* user perceived performance, BTD in this case, is representative. In particular our results have focused on flow sizes following a heavy tail distribution – as currently seen on the Internet [26], [27], [28]. Such distributions have a large fraction of flows which are small and some exceedingly large ones. This is, in a sense, ideal for SD, as the performance for a large number of small flows can be improved at a small marginal performance cost to large flows, possibly leading to a dramatic improvement in the average performance. Moreover speeding up a collection of already short-lived flows may not really provide a great value from the user's point of view. [6]

Thus it is of interest to see how the performance benefits of SD are seen by flows corresponding to different size categories. Fig.7 shows the average BTD (on a logarithmic scale) seen by sets of flows with increasing sizes. Flow sizes were divided into 6 bins, where the first bin corresponds to flows of size $[10^3, 10^4)$ bits, the second $[10^4, 10^5)$, and so on. The results are shown for various loads, including an overloaded case, corresponding to a 'transient' finite time simulation of the unstable system, see [7] for details. As seen, when the load increases SD maintains good performance for small to medium size flows, while FS 'uniformly' degrades performance for all flows. In fact SD benefits the majority of flows (99% of flows fall into the first 5 bins in our simulated case) while incurring 'comparable' performance degradation to FS for the few very large flows. Refer to [3] for an interesting analysis of this phenomenon.

The *graceful* performance degradation achieved by SD as the system becomes increasingly loaded might be viewed as beneficial in that it 'masks' congestion for smaller flows. When networks/systems in the real world can be occasionally congested for a short period of time, it might be wise to let the majority of flows, *i.e.*, the small to medium ones, continue to go through quickly, without slowing down too much for the large flows as compared to that under FS. Note further that in reality users sharing a congested resource may become impatient

[6]In practice small flows are likely to be bottlenecked by the slow start phase and/or possibly large round trip times, whence there is not much of a speed-up to be realized. Moreover if the performance associated with the flow is already acceptable, *e.g.*, the total response time is within a second, there may not be much to gain here.

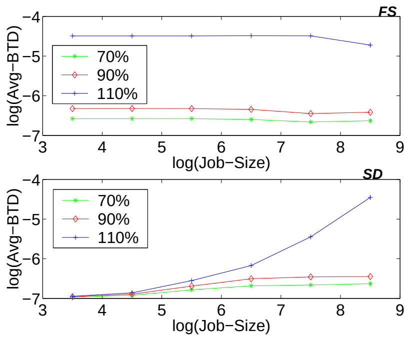

Fig. 7. Average BTD for different size flows under FS (top) and SD (bottom) in the under-load (70%), heavy-load (90%), and overload (110%) regime on a 10 Mbps link shared by Poisson arrivals with bounded Pareto flow size distribution of mean 5 Kbytes.

and interrupt their transfers prior to completion. Thus a new question arises as to how the interaction between the bandwidth sharing policy and impatient users impacts the overall resource utilization.

C. *Sharing bandwidth among impatient users*

Impatient users would presumably interrupt a transfer due to perceived poor performance, *e.g.*, delay, throughput, or the progress of the transfer. This may in turn cause poor network utilization or goodput, *i.e.*, the total number of bits per second associated with completed transfers. User impatience behavior can be very complex and change under various circumstances. In [30] we propose several impatience behavior models and investigate their interactions with the FS and SD bandwidth sharing policies. In this paper, we will focus on users that are sensitive to a delay constraint $\bar{d}$, *i.e.*, a user will abort a transfer if $\bar{d}$ is exceeded. The authors of [31] propose a simple analytical approximation for the goodput achieved on a single link that serves such delay sensitive users based on FS. Below we propose a similar 'crude' model for links using SD, based on the M/GI/1-SRPT queue.

Let c denote the link capacity and F the flow size distribution function for flows arriving to the link as a Poisson process. Following [3], the steady state delay on a possibly overloaded M/GI/1-SRPT queue for a flow of size $p < p^*$ can be expressed as $\mathbb{E}[D(p)] =$

$$\left(\int_0^p \frac{ds}{1-\rho(s)} + \frac{\lambda}{2} \cdot \frac{\int_0^p s^2 dF(s) + (F(p^*) - F(p))p^2}{(1-\rho(p))^2} \right) \frac{1}{c},$$

where p^* is such that $\rho(p^*) = 1$ and $\rho(p) = \lambda \int_0^p s dF(s)/c$. Since $\mathbb{E}[D(p)]$ is monotonically increasing in p, one can determine a unique $\bar{p}$ such that

$$\bar{p} = \text{argmax}\{p \mid \mathbb{E}[D(p)] \leq \bar{d}\}.$$

Assuming a flow of size p sees *exactly* its expected delay $\mathbb{E}[D(p)]$, then all flows with size exceeding $\bar{p}$ would interrupt their transfers. Thus one can approximate the goodput (bits/sec) and throughput (transfers/sec) associated with a system using SD and supporting impatient users by $\rho(\bar{p})$ and $\lambda \int_0^{\bar{p}} f(s) ds$, respectively. These approximations should be conservative since the above expressions do not account for the reduced load resulting from users actually aborting their transfers, *i.e.*, leaving the system.

Fig.8 and 9 show the analytical approximations and simulations for the goodput and throughput achieved on overloaded FS and SD links supporting impatient users with increasing tolerance to delay and increasing load, respectively. We draw the goodput and throughput as in the percentage of the link capacity and the offered load (in transfers/sec), respectively. As can be seen both the goodput and throughput achieved by SD are significantly larger than FS under various conditions. Moreover these measures degrade dramatically under FS as the traffic load increases. By contrast, SD maintains a fairly good resource utilization, *i.e.*, goodput, and lets most flows go through even the system is loaded beyond 200%. Thus not only does SD provide improved performance, *e.g.*, reduced BTD, to users over FS, but it ensures the network resources are effectively used even upon overload, making the network robust to traffic variations.

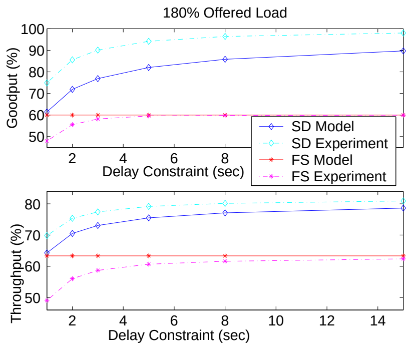

Fig. 8. Goodput (top) and throughput (bottom) achieved under M/GI/1-FS and M/GI/1-SRPT queues shared by impatient users with a homogeneous delay constraint as one increases (relaxes) the delay constraint.

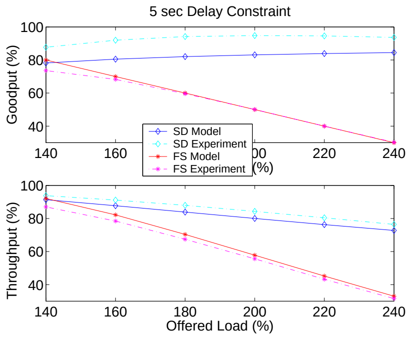

Fig. 9. Goodput (top) and throughput (bottom) achieved under M/GI/1-FS and M/GI/1-SRPT queues shared by impatient users with a homogeneous delay constraint as one increases the offered load.

The above analysis assumes the users have homogeneous impatience behavior, *i.e.*, all users have the same delay constraint. We further conduct simulations to examine systems in which users have heterogeneous impatience. For simplicity, we consider a 200% loaded single link serving two types of delay sensitive users. Both types offer traffic load that equals the link capacity and have delay constraints of 1 and 5 seconds, respectively. Table II shows the goodput and throughput (again as a percentage) for the two user types under FS and SD. These results suggest that SD can achieve a higher overall utility by letting more users with a delay constraint of 1-sec go through, without compromising those with the 5-sec constraint. In fact, by examining other scenarios with mixed heterogeneous users, we find that under FS the users with a smaller delay constraint are increasingly penalized when the difference in delay constraints among users increases, while SD consistently achieves good but still differentiable performance for users with different level of impatience.

TABLE II
GOODPUT AND THROUGHPUT ACHIEVED FOR HETEROGENEOUS DELAY SENSITIVE USERS (1-SEC,5-SEC) UNDER FS AND SD.

	FS	SD
Goodput (%)	(26.8%,48.0%)	(45.9%,51.9%)
Success Rate (%)	(77.3%,95.8%)	(94.9%,96.5%)

As evidenced by the use of today's Internet, it seems to be unavoidable for users to experience congestion, possibly only occasionally, and thus interrupt or continue their transfers based on how much they value each transfer. In fact, without any additional traffic differentiation scheme, each user's (im)patience may be already a good expression of his utility to complete a transfer. The above findings suggest that using SD (versus FS) one may achieve both differentiation among users with different patience/utility, and high throughput for all types of users upon overloads. Furthermore, SD may offer an additional incentive for users to stick with their transfer if congestion arises, rather than repeatedly interrupting and re-trying. This will lead to yet another user self-differentiation and hence a more efficient use of resources.

D. Do peak rate constraints limit the benefits ?

As argued by [31] today's best effort data networks may look much more like circuit switched networks, in that flows are 'individually' peak rate constrained, either by their access links, *e.g.*, modem/wireless users, or intrinsic limitations of TCP, *e.g.*, maximum window size divided by round trip time. Thus from the point of view of core network links/routers, only a very high traffic load would cause such flows not to see their peak rate constraints. Thus a natural question arises as to whether SD will realize the gains we have discussed above in practice.

Clearly if users are peak rate constrained, the degree to which SD can differentiate allocations among users with different residual flow size decreases, and thus the improvement in the performance over FS will decrease. This of course depends on the degree to which the link is loaded, *i.e.*, differentiation can achieve higher benefits at higher loads. Fig.10 shows how close the performance improvement achieved by SD is to the optimal (*i.e.*, performance associated with having access to the peak rate) when one compares both with FS, under various peak rate constraints and traffic loads. As can be seen SD achieves almost optimal performance. The improvement is significant if flows (1) share heavy or overloaded resources and (2) are not significantly constrained relative to the bottleneck capacity. Note that in practice, user flows are 'concentrated' into increasingly large capacity links, *e.g.*, data grooming gateways, prior to reaching the backbone, and then go to the destination access network. Hence effectively the peak rate of a (possibly aggregate) flow from a concentration link's point of view will be the capacity of the previous concentration point, *i.e.*, not that far away from it's own capacity. Furthermore, it is also unlikely that all such data grooming points are overprovisioned. Our experiments [7] find that for such typical hierarchical networks, SD consistently achieves high performance improvements.

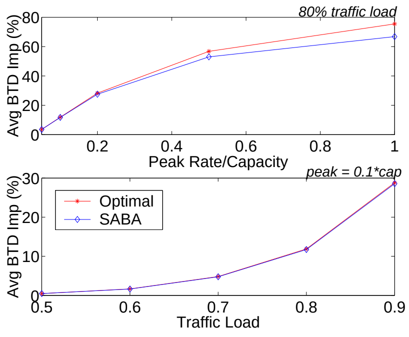

Fig. 10. Improvements in average BTD achieved by SD over FS as compared to the optimal performance (*i.e.*, peak rate) over FS on a single 10 Mbps link with Poisson arrivals of bounded Pareto flow sizes, as one increases the homogeneous peak rate constraints and loads.

As with the case of user impatience, we can view an individual peak rate constraint as a user's tacit indication of the utility of his/her transfers. Thus an appropriate metric for system performance, or utility, is the fraction of the transfers that achieve throughput close to their individual peak rate constraint. Our simulations show that under SD about 95% of the flows see a throughput exceeding 90% of their peak rate, while only 42% of the flows see this under FS. In Fig.11 the afore-mentioned percent of users perceiving 'good' throughput are broken out by flow size. As expected in order to achieve these benefits SD favors shorter to medium sized flows, compromising the larger ones, where the differences for the large ones are relatively small as compared to those for the small to medium ones.

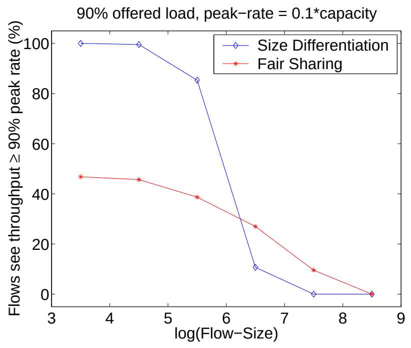

Fig. 11. The proportion of flows within various size bins that realize 90% of their peak rate (1Mbps) on a 10 Mbps link supporting a 90% load.

In a system with heterogeneous peak rates, we [7] find that flows with reduced constraints (larger peak rates) would see significant performance improvements under SD over FS, while those with tight constraints will mostly see their peak rates under both SD and FS. This is an arguably desirable characteristic since users with larger access rates typically pay more, so the system may achieve a higher utility for them and for the network/service provider.

VI. Conclusion

In this paper, we propose to enhance user perceived performance by differentiating based on the residual flow sizes. In particular, we propose to realize the size-based differentiation at the transport layer by a small alteration of TCP Reno. A common concern with size-based differentiation is the possibility that malicious users may cut their files into small pieces and transfer them in parallel, or even cheat on their file sizes. This is an inherent problem with end-to-end flow control and the Internet where a cooperative environment is assumed. To resolve such matters, either an appropriate policing scheme needs to put into place or a pricing scheme that makes such behavior expensive would be required. This is however outside the scope of this paper. We shall simply note that many cooperative environments do exist where our scheme would provide many advantages.

As mentioned in the introduction, other researchers have considered benefits that size based differentiation might provide on today's best effort networks. For example, the work of [5] makes a convincing argument for implementing SRPT-like policies at the 'server' side in order to enhance server performance. From a system performance point of view this approach recognizes that the server may be a bottleneck, and proposes differentiation at the 'application/os' level rather than the transport level. This approach achieves similar performance gains for the servers as those for the single link case discussed in this paper, but ignores the possibility that bottlenecks may arise elsewhere on network resources, *e.g.,* links shared by a server farm, etc. Implementing differentiation at the transport layer takes a more global view of congestion, but requires a cooperative environment, *i.e.,* adoption of a new protocol. We have shown that networks supporting a mix of transport protocols might be designed to share the network effectively. We believe that combining size-based differentiation with peak access rate policing and/or user self-admission resulting from impatience, may be a reasonable avenue to maximizing the overall utility of a best effort network, without excessive complexity.

Appendix

I. Proof for Lemma 1

Proof: We will prove a more general case where a pre-determined time-varying link capacity is considered. Let J denote the set of flows that share a single link with capacity $c(t),\ t \geq 0$. Consider a bandwidth allocation $\mathbf{x} = (x_j(t),\ j \in J,\ t \geq 0)$ that allocates positive bandwidths to more than one flows during some time interval $[t^*,\ t^*+\tau)$ for some $\tau > 0$. We denote the set of flows that receives positive bandwidth during $[t^*,\ t^*+\tau)$ as $J^+(t^*, t^*+\tau) \subseteq J$ where $|J^+(t^*, t^*+\tau)| > 1$. Furthermore, define $A(t,\infty) = \{j | j \in J,\ a_j > t^*\}$. Consider the flow $k = \mathrm{argmin}_{j \in J - A(t,\infty)}\{f_j\}$ where f_j is the finishing time of flow j. In the sequel we will show that one can always improve flow k's delay and thus its BTD without increasing any other flow's delay by slightly altering $\mathbf{x}$.

First, consider the case where $k \in J^+(t^*, t^*+\tau)$. Let $\mathbf{x}' = (x'_j(t),\ j \in J,\ t \geq 0)$ be an alternative feasible bandwidth allocation which only differ from $\mathbf{x}$ during the time interval $[t^*, f_k)$ in that (1) a full 'reduced' capacity $\tilde{c}(t) = c(t) - \sum_{j \in A(t,\infty)} x_j(t)$ is allocated to flow k until it finishes, (2) arbitrary bandwidths allocated to all flows $j \in J - A(t,\infty) - k$ during $[t^*,\ f_k)$ requiring that $\int_{t^*}^{f_k} x'_j(t) = \int_{t^*}^{f_k} x_j(t)$, and (3) no changes for flows in $A(t,\infty)$. Clearly flow k will finish earlier than f_k under the new $\mathbf{x}'$, since $x_k(t) < \tilde{c}(t)$ during at least $[t^*,\ t^*+\tau)$. Meanwhile, since other flows in $j \in J - A(t,\infty) - k$ completes the same amount of work during $[t^*,\ f_k)$ and none of them finishes before f_k even under $\mathbf{x}$, they do not see a change in their delay. Finally, no flow in $A(t,\infty)$ sees any change under $\mathbf{x}'$ since they have the same bandwidth allocation.

Next we consider the impact of $\mathbf{x}'$ for the scenario where $k \notin J^+(t^*, t^*+\tau)$. Clearly, flow k's delay is still smaller under $\mathbf{x}'$ since $x_k(t) = 0$ during $[t^*,\ t^*+\tau)$. Same arguments apply for the other flows, and thus the theorem follows. ■

II. Proof for Theorem 1

Proof: According to Lemma 1 we only need to determine the optimal 'service' order of the flows in $J(t)$, the set of on-going flows at time t. Without loss of generality, we index the flows in $J(t)$ according to the non-decreasing order of $p_j \cdot p_j(t)$ with ties broken arbitrarily, *i.e.,* $p_i \cdot p_i(t) \leq p_j \cdot p_j(t),\ \forall i < j$. Also for convenience we assume unit link capacity.

Consider a service order Ψ that does not follow the SPTP rule, *i.e.,* there exists at least one pair of flows (i,j) such that $p_i \cdot p_i(t) < p_j \cdot p_j(t)$ but i is served after j. For convenience we call such pair to be 'non-conforming.' Note that there must exist at least one non-conforming pair that is in consecutive service order within Ψ. Let i and $j = i+1$ denote one non-conforming and consecutive pair of flows. Now if we consider a new service order Ψ' that is the same as Ψ except it swaps the service order of i and j. The difference in the residual BTD is

$$\sum_{k \in J(t)} \frac{d_k^{\Psi'}(t)}{p_k} - \sum_{k \in J(t)} \frac{d_k^{\Psi}(t)}{p_k} = \frac{p_i(t)}{p_j} - \frac{p_j(t)}{p_i} < 0.$$

The last inequality is due to that $p_i \cdot p_i(t) < p_j \cdot p_j(t)$. Continuing this swapping procedure one will have a service order within finite steps that satisfies SPTP with each swapping step resulting in less overall residual BTD. Note that having multiple service orders satisfying SPTP can only happen when there is a tie in $p_k(t) \cdot p_k$. These service order however incur no difference in the overall residual BTD. Thus a service order that follows SPTP must minimize the overall residual BTD. ■

III. Proof for Theorem 2

Proof: For convenience we set $c_l = 1$ and re-normalize all the parameters accordingly. We decompose the residual delay (and thus the residual BTD) for each flow into two components: (1) the total service time consumed by the flows on the same type of route that finish before that flow completes (thus including itself), and (2) the service time consumed by the flows on the other route type that are considered to be served before the given flow. Since the capacity should always be fully allocated to one of the route types (Lemma 2) and we assume SPTP to be the intra-route policy, we can determine the service order for the flows on the m-hop route and that for the flows on all 1-hop routes regardless of how they are scheduled with respect to those on the other route type. The service order for route type $s = 0,1$ thus follows the non-decreasing order of $\tilde{p}^s_j(t)$, as defined in the text, and $\tilde{p}^s_j(t)$ is in fact the first delay component for flow j (the jth flow to finish) on route type s.

Thus to minimize the overall BTD, we should minimize the residual BTDs due to the second component. Note that the second component is determined by how one interleaves the two service schedules for the m-hop flows and 1-hop flows. Without loss of generality we consider when to start serving each of the m-hop flows among the 1-hop flows one by one. Suppose we start serving m-hop flow j immediately after k 1-hop flows finish. The total residual BTD contributed by the second component to the m-hop flow j and all 1-hop flows is

$$\Delta b(j,k) \quad = \quad \frac{\tilde{p}^1_k(t)}{p^0_j} + p^0_j(t) \cdot \sum_{i=k+1}^{n_1(t)} \frac{1}{p^1_i},$$

where $\tilde{p}^1_0(t) = 0$, and $k = 0$ corresponds to the case where the m-hop flow j is served before all 1-hop flows. This is true for all $j \in \{1,\dots,n_0(t)\}$ and $k \in \{0,\dots,n_1(t)\}$.

We now may denote the number of 1-hop flows that should be served before serving the j^{th} m-hop flow for the purpose of minimizing $\Delta b(j,k)$ as $k^*_j(t) = \text{argmin}_{k\in\{0,\dots,n_1(t)\}}[\Delta b(j,k)]$. Simple derivation can show that $k^*_i \le k^*_j$, $\forall i < j$ since $p^0_j \cdot p^0_j(t)$ is non-decreasing in j. Thus there exist a set of $\{k^*_j,\ j = 1,\dots,n_0(t)\}$ that minimizes the set of $\{\Delta b(j,k),\ j = 1,\dots,n_0(t)\}$ as well as $\sum_{j=1}^{n_0(t)} \Delta b(j,k)$, *i.e.*, the total BTD incurred for all flows due to the second delay component. ■

Acknowledgment

The authors would like to thank Prof. Zhi-Li Zhang for his help to facilitate the review process, and the anonymous reviewers for their comments to improve the paper.

References

[1] J. Roberts, "Flow aware networking for effective quality of service control," *IMA Workshop on Scaling*, October, 1999.

[2] A. Feldmann, R. Cáceres, F. Douglis, G. Glass, and M. Rabinovich, "Performance of web proxy caching in heterogeneous bandwidth environment," *Proc. IEEE INFOCOM*, vol. 1, pp. 107–116, 1999.

[3] N. Bansal and M. Harchol-Balter, "Analysis of SRPT scheduling: Investigating unfairness," *Proc. ACM SIGMETRICS*, pp. 279–290, 2001.

[4] R. Perera, "The variance of delay time in queueing system M/G/1 with optimal strategy SRPT," *AEU, Archiv fuer Elektronik und Uebertragungstechnik*, vol. 47, no. 2, pp. 110–114, 1993.

[5] M. Harchol-Balter, N. Bansal, B. Schroeder, and M. Agrawal, "Implementation of SRPT scheduling in web servers," *Proc. 7th Annual Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 11–35, 2001.

[6] X. Chen and J. Heidemann, "Preferential treatment for short flows to reduce web latency," *USC/ISI Technical Report, ISI-TR-548*, 2001.

[7] S.-C. Yang, "Dynamic resource allocation on networks supporting elastic users," *Ph.D. Dissertation, The University of Texas at Austin*, 2001.

[8] X. Su and G. de Veciana, "Dynamic multi-path routing: Asymptotic approximation and simulations," *Proc. ACM SIGMETRICS*, vol. 29, no. 1, pp. 25–36, 2001.

[9] S. Xu and G. de Veciana. Predictive routing to enhance QoS for stream-based flows sharing excess bandwidth. *Computer Networks*, 42:65–82, May 2003.

[10] A. Shaikh, J. Rexford, and K. Shin, "Load-sensitive routing of long-lived IP flows," *Proc. ACM SIGCOMM*, pp. 215–226, 1999.

[11] 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.

[12] S.J. Golestani and S. Bhattacharyya, "A class of end-to-end congestion control algorithms for the Internet," *Proc. ICNP*, pp. 137–150, 1998.

[13] L. Massoulié and J. Roberts, "Bandwidth sharing: objectives and algorithms," *Proc. IEEE INFOCOM*, vol. 3, pp. 1395–1403, 1999.

[14] S.H. Low and D.E. Lapsley, "Optimization flow control I: Basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, pp. 861–874, 1999.

[15] 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.

[16] R. La and V. Anantharam, "Charge-sensitive TCP and rate control in the Internet," *Proc. IEEE INFOCOM*, vol. 3, pp. 1166–1175, 2000.

[17] P. Hurley, J. Y. Le Boudec, and P. Thiran, "A note on the fairness of additive increase and multiplicative decrease," *Proc. ITC*, 1999.

[18] T. Bonald and L. Massoullié, "Impact of fairness on Internet performance," *Proc. ACM SIGMETRICS*, pp. 82–91, 2001.

[19] G. Fayolle, A. de La Fortelle, J.-M. Lasgouttes, L. Massoullié, and J. Roberts, "Best-effort networks: modeling and performance analysis via large network asymptotics," *Proc. IEEE INFOCOM*, 2001.

[20] T. Bu and D. Towsley, "Fixed point approximations for TCP behavior in an AQM network," *Proc. ACM SIGMETRICS*, vol. 29, no. 1, pp. 216–225, 2001.

[21] G. de Veciana, T.-J. Lee, and T. Konstantopoulos, "Stability and performance analysis of networks supporting rate-adaptive services," *IEEE/ACM Trans. Networking*, pp. 2–14, 2000.

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

[23] A. Goel, M. Henzinger, S. Plotkin, and E. Tardos, "Scheduling data transfers in a network and the set scheduling problem," *Proc. Annual ACM Symp. Theory of Computing*, pp. 189–197, 1999.

[24] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. E. Gehrke, "Online scheduling to minimize average stretch," *Proc. FOCS*, pp. 433–443, 1999.

[25] L. E. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, pp. 678–690, 1968.

[26] 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.

[27] N. Brownlee and K.C. Claffy, "Internet stream size distributions," *Proc. ACM SIGCOMM*, pp. 282–283, 2002.

[28] V. Paxson and S. Floyd, "Difficulties in simulating the Internet," *IEEE/ACM Trans. Networking*, vol. 9, no. 4, pp. 392–403, 2001.

[29] M. E. Crovella, B. Frangioso, and M. Harchol-Balter, "Connection scheduling in web servers," *Proc. USITS*, pp. 243–254, 1999.

[30] S.-C. Yang and G. de Veciana, "Bandwidth sharing: the role of user impatience," *Proc. IEEE GLOBECOM*, 2001.

[31] S. Ben Fredj, T. Bonald, A. Proutiére, G. Régnié, and J. Roberts, "Statistical bandwidth sharing: a study of congestion at flow level," *Proc. ACM SIGCOMM*, 2001.

Shanchieh Jay Yang Shanchieh Jay Yang (S '96 / M '02) received his B.S. from the National Chio-Tung University (Taiwan) in 1995, and M.S. and Ph.D. in electrical and computer engineering from the University of Texas at Austin in 1998 and 2001, respectively. In 2002, he joined the Department of Computer Engineering at the Rochester Institute of Technology where he is currently an assistant professor. His research focused on performance analysis and algorithm development for computer and communication networks, particularly the Internet and sensor networks. He is the recipient of a Huang, Hsien-Hao Scholarship from the Bureau of Engineering Services Corporation, Taiwan, and a TxTEC Graduate Fellowship. His email address is: scyang@ieee.org.

Gustavo de Veciana Gustavo de Veciana(S'88-M'94-SM 2001)received his B.S., M.S, and Ph.D. in electrical engineering from the University of California at Berkeley in 1987, 1990, and 1993 respectively. He is currently a Professor at the Department of Electrical and Computer Engineering at the University of Texas at Austin. His research focuses on the design, analysis and control of telecommunication networks. Current interests include: measurement, modeling and performance evaluation; wireless and sensor networks; algorithms for computer aided design of reliable systems. Dr. de Veciana has been an editor for the IEEE/ACM Transactions on Networking. He is the recipient of a General Motors Foundation Centennial Fellowship in Electrical Engineering and a 1996 National Science Foundation CAREER Award, and co-recipient of the IEEE Bill McCalla Best ICCAD Paper Award for 2000.