Size-based Adaptive Bandwidth Allocation: Optimizing the Average QoS for Elastic Flows

Shanchieh Yang *(scyang@ece.utexas.edu)*, Gustavo de Veciana *(gustavo@ece.utexas.edu)*
Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712

Abstract— **With a view on improving *user perceived* performance on networks supporting elastic flows, *e.g.,* multimedia/data file transfers, we identify the key properties that an online dynamic bandwidth allocation policy should have. We then propose a family of bandwidth allocation criteria which depends on the residual work of on-going transfers. Analysis and simulations show that allocating bandwidth in this fashion can improve the user perceived average bit transmission delay (BTD), *i.e.,* delay/flow size, by up to 70% at 80% traffic load over traditional approaches. A simple implementation based upon TCP Reno, exemplifies how one might approach practically realizing such gains. Further studies on simple network topologies show that as the penetration of the proposed transport mechanism increases, users will have the proper incentives to upgrade from TCP Reno, and that the overall performance is better for all users once the penetration exceeds 20%.**

I. Introduction

RECENTLY much attention has been paid to characterizing how network bandwidth is shared among data transfers, *i.e.,* TCP mediated transfers of best-effort flows. This work has, for the most part, focused on the character of the 'equilibrium', *e.g.,* fairness, reached by various network/user adaptation mechanisms when a *fixed* set of flows share the network, *e.g.,* [1], [2], [3], [4]. It is however unclear how 'fair' bandwidth allocations impact the user perceived performance in a *dynamic* setting where users come and go. The work in [5], [6], [7], [8], [9] leads the way in this direction focusing on stability and performance in the dynamic regime. Our aim in this paper is to investigate how one might enhance, if not optimize, average user perceived performance from the ground up through a new class of transport mechanisms.

Users transferring data files are typically said to be elastic in the sense that they have a wide tolerance to changes in the transmission rate throughout the transfer. A reasonable performance measure for such elastic users/flows may be the time it takes to complete the transfer, particularly, but not exclusively, when interactivity is involved, *e.g.,* web browsing. Transfer delays alone are not necessarily representative of user satisfaction as there may be a wide disparity in the size of ongoing transfers [1]. In fact, some transfers may and should be expected to take longer to complete, *e.g.,* multimedia rich documents or bulk backups, and thus the sensitivity to delay of such users is likely to be diminishing in file size. In this case a natural quality of service metric would be the *Bit Transmission Delay* (BTD)[2], *i.e.,* delay/file size. Note that the reciprocal of a flow's BTD is the user perceived throughput. Thus minimizing the average user perceived BTD is coupled with increasing the average perceived throughput. Other performance metrics could and may be considered in the future. We will however focus on the average BTD as a reasonable proxy for user perceived performance.

Upon initiating a file transfer the amount to be sent is typically known on the sender side, *e.g.,* static web content. This, however, is currently ignored by transport and network level mechanisms which determine how bandwidth is shared among ongoing elastic flows. In this paper we take a novel approach by investigating how one might employ the file size information to optimize the average BTD users will see. Recognizing that ease of deployment would be an issue we propose mechanisms that are akin to TCP, in that they are decentralized, robust, and likely scale fairly well. We will show that dynamic bandwidth allocations that depend on the remaining volume of data to be sent can significantly enhance the average BTD seen by elastic flows.

The rest of the paper is organized as follows. A formal description of the dynamic bandwidth allocation problem is given in the next section. In §III we consider a dynamic set of flows sharing a single bottleneck link and derive and compare policies that enhance the average BTD over the commonly used 'fair share'. Then, in §IV, we investigate the interaction among elastic flows on various routes and extend our qualitative findings to general networks, by considering a prototypical linear network in detail. Based on these insights we propose and discuss a generalized size dependent bandwidth allocation criterion, SABA, in §V. Fluid-flow simulations are presented in §VI to exhibit the potential performance gains achieved by using SABA over traditional fairness criteria. In §VII we briefly discuss a proof of concept implementation, *i.e.,* a transport mechanism, which realizes SABA, called SAReno. Packet level simulations of the proposed transport mechanism are then conducted showing the gains achieved by SAReno over Reno. Various other design issues are discussed including the performance in networks supporting heterogeneous transport mechanisms. Conclusions and final remarks are given in §VIII.

II. Problem Description and Related Work

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.[3] The set of links traversed by route r is captured by a 0-1 matrix A where A_{lr} is 1 if router r tra-

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.

[1] Studies show that the files being transferred on the Internet exhibit great variability in size, see *e.g.,* [10].

[2] BTD is similar to 'slow-down' used for scheduling problems, *e.g.,* [11], [12].

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

verses 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 departs. 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. As mentioned earlier we will focus on the bit transmission delay as the performance measure of interest where the BTD for flow j is given by $b_j = d_j/p_j$. We summarize the notation in Table I and formally define the problem of interest as follows.

Problem 1 (Dynamic Bandwidth Allocation)

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

TABLE I
NOTATION SUMMARY

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	$d_j, b_j = d_j/p_j,\ \frac{1}{\lvert J\rvert}\sum_{j\in J} d_j/p_j$ 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 allocations that minimize the average BTD for arbitrary arrivals as BTD-optimal.

Authors of [5], [6], [7], [8], [9] 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, the only existing work that attempts to find a BTD-optimal policy at the network level was conducted in [11]. 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 [12]. 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 metric [9], [12], [13]. 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 this problem, in the sequel we will pursue this problem in the 'transient' regime where it 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 problem, 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. Optimizing Average BTD: Single Link Case

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.

A. 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 their fair 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 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 the others - see the appendix. 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.

B. Size-dependent differentiation

A well known size-dependent scheduling, or 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. Then SRPT assigns the full link capacity to a flow $j^* \in J(t)$ with the smallest residual work, 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 (with fixed capacity) [15], and was recently shown to be 2-competitive for the average BTD metric [12]. With a view on further enhancing performance and developing allocation policies that can be implemented, below we propose several other 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 corresponds to a greedy policy which at any time seeks to minimize the overall 'residual' BTD assuming there are no future arrivals.

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 were no additional arrivals.

A proof of this result is given in the appendix. Thus 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 incurring 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.

C. Performance gains of size-dependent differentiation

Recently [13] 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 new policies proposed above, *i.e.,* SPTP, RPT-WS, and PTP-WS, vs. fair sharing via simulation.

Simulations were conducted for a 10 Mbps link shared by flows arriving according to Poisson processes and have sizes selected from a bounded Pareto distribution with mean 5 KBytes. 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 of $\alpha = 1$ in RPT-WS and PTP-WS, and 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.

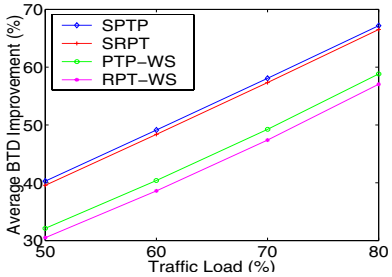

Fig. 1. Average BTD improvement for SRPT, SPTP, RPT-WS, and PTP-WS over fair sharing as traffic load increases.

Above analysis provides various bandwidth allocation approaches that favor small flows, as an avenue towards minimizing 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 [13] and [16] 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 [10]. Our own experiments also confirm that starvation is indeed not a concern, particularly as compared to fair sharing. Refer to [14] for a more detailed discussion.

IV. Optimizing Average BTD: Network Case

Although the single link case suggests one should 'always' favor small flows 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 and which can 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 were a

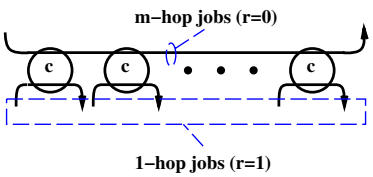

Fig. 2. Linear network: m equal capacity links.

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. We then formally define a class of dynamic bandwidth allocation criteria that one might use to enhance the average BTD performance on general networks.

A. Example: Symmetric Linear Network

To simplify our analysis, suppose that the 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. That is, at any given time t, an 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, regardless of the value of $y_r(t)$. 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, $\tilde{p}_j^s(t) = \sum_{i \leq j\ \&\ r_i = r_j} p_i^s(t),\ s = 0, 1$, 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

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

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.

Despite its lengthy proof (given in the appendix), (1) in Algorithm 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.

Consider for example the m-hop route. If the full capacity were allocated to it, the first flow (in SPTP order) will take $\tilde{p}_1^0(t)/c = p_1^0(t)/c$ seconds to complete and have a weight $1/p_1^0$ - thus a weighted throughput of $c/(p_1^0 \cdot p_1^0(t))$. One can show that this corresponds to the highest weighted throughput one can achieve by allocating the full capacity to this route for any time window - *i.e.,* the left hand side of (1). By contrast, when flows can be served in parallel, *e.g.,* 1 hop flows on distinct routes, one might achieve the maximum weighted throughput by considering a time window in which 'multiple' flows complete. In particular if the full capacity is allocated to the 1-hop routes, then $\tilde{p}_k^1(s)/c$ is the time to complete k flows (in SPTP order for each 1-hop route) and $\frac{1}{k}\sum_{l=1}^{k}(1/p_l^1)$ is their associated weight. Considering all possible windows into the future, one will obtain the term on the right of (1). 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. 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 an 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 [1], [2], [4], 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, $(\mathbf{x}^*(t), t \geq 0)$, 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

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

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

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

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

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

Bandwidth allocations associated with maximizing the utility functions defined in (2) with *fixed* weights has been widely considered [1], [2], [3], [4]. Notice that maximizing such overall user utility functions subject to resource constraints naturally favors flows that use fewer resources, therefore achieving higher service parallelism. Our premise is that the introduction of per-flow weights depending on the residual size of flows enable SABA to achieve better average BTD. In the next section we shall discuss how the choice of weight functions might impact SABA's performance.

V. Role of Residual Size Dependent Weights

The challenge to devising size dependent weights is to provide both appropriate intra and inter-route bandwidth allocations, while not precluding possible implementation. In particular we propose to parameterize a flow's weight as

$$w_j(t) \;=\; w_{\text{in}}(p_j(t))\cdot\left[\frac{\sum_{k\in J_{r_j}(t)}(w_{\text{ex}}(p_k(t)))^{1/\beta}}{\sum_{k\in J_{r_j}(t)}(w_{\text{in}}(p_k(t)))^{1/\beta}}\right]^\beta, \qquad (3)$$

where $J_{r_j}(t)$ is the set of flows sharing route r_j with flow j, and the internal (intra-route) and external (inter-route) weight functions w_{in} and w_{ex} are non-increasing in the residual flow sizes. Note that we have selected weights that only depend on the 'residual' size[4].

Notice that by using the same function for w_{in} and w_{ex}, the weight of each flow based on (3) will depend on only its own residual size, *i.e.,* $w_j(t) = w(p_j(t))$. If it is permissible to coordinate among flows that share the same route, *i.e.,* have access to their residual flow sizes, one may employ different functions for w_{in} and w_{ex}, in which case a flow's weight depends on the residual sizes of all flows on the same route. In the sequel, we will discuss the possible benefits of the latter option and how one might select w_{in} and w_{ex}.

A. Decoupling Intra & Inter-route Bandwidth Allocation

An important character of our proposed weight function in (3) is that it allows one to decouple the impact of residual flow sizes on the intra and inter-route bandwidth allocation by using different functions for w_{in} and w_{ex}. Facts 1 and 2 below characterize how bandwidth would be allocated among flows sharing the same route and across routes– proofs are given in the appendix.

Fact 1: At any time t the SABA bandwidth allocation for two flows, i and j, sharing the *same route* satisfies

$$\frac{x_i^*(t)}{x_j^*(t)} \;=\; \left(\frac{w_i(t)}{w_j(t)}\right)^{1/\beta} \;=\; \left(\frac{w_{\text{in}}(p_i(t))}{w_{\text{in}}(p_j(t))}\right)^{1/\beta}.$$

Fact 2: Let $y_r(t) = \sum_{j\in J_r(t)} x_j(t)$ be the total bandwidth allocated to the flows sharing route r at time t. The first order optimality condition associated with SABA can be restated as, $\mathbf{y}^*(t) = (y_r^*(t),\; r \in R)$ is optimal if for any other feasible allocation $\mathbf{y}(t) = (y_r(t),\; r \in R)$,

$$\sum_r v_r(t)\,\frac{y_r(t)-y_r^*(t)}{(y_r^*(t))^\beta} \;\leq\; 0,$$

where

$$v_r(t) \;=\; \left(\sum_{j\in J_r(t)} w_j(t)^{\frac{1}{\beta}}\right)^\beta \;=\; \left(\sum_{j\in J_r(t)} w_{\text{ex}}(p_j(t))^{\frac{1}{\beta}}\right)^\beta$$

denotes an 'aggregate weight' associated with route r.

Note that the relative bandwidth allocated to flows sharing the same route only depends on the selection of w_{in}, while the aggregate route bandwidth depends on w_{ex}. This allows one to use different functions to determine the intra and inter-route SABA bandwidth allocations. In particular, as the results in §III and IV suggest, one might want to discriminate heavily against large flows for the intra-route allocation, but trade off such discrimination with parallelism when it comes to inter-route bandwidth allocation. Our simulations suggest that this decoupling scheme can achieve a better average BTD than that resulting from using the same residual size based weight for both the intra and inter-route bandwidth allocations.

[4]Our experience based on simulation suggests that there are only marginal differences in performance if one also introduces the original size.

B. Design of Weights

In this subsection we briefly discuss how w_{in} and w_{ex} might be selected. The goal is to provide residual size-based differentiation favoring small flows, thus we consider decreasing functions in the residual flow size. Two possible candidates are the exponential and the reciprocal functions, *i.e.,* $w(p_j(t)) = e^{-\alpha p_j(t)}$ and $w(p_j(t)) = (p_j(t))^{-\alpha}$, respectively, where $\alpha > 0$ allows one to vary the degree of size-based differentiation. In particular, one may use a larger α_{in} for w_{in} to provide a more significant discrimination for intra-route bandwidth allocation, and a smaller α_{ex} for w_{ex}. Note that the weight functions are based on the 'residual' sizes. Thus 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 toward completion. Similarly if a given bandwidth allocation achieves good parallelism, then it will be reflected in the relative increase in weights for the associated flows, and this characteristic of the allocation will be further emphasized.

Based on our experiments, we observe that SABA in general achieves similar performance, particularly when compared to traditional fairness criteria, as long as one provides reasonable differentiation in favor of small flows. In particular, we observe a slightly better performance gain if one decouples the intra and inter-route bandwidth allocation, by using, for example, a larger value for α_{in} than α_{ex}. Furthermore, due to the fact that increases in β reduce the impact of the size-dependent weights (see Fact 1 and 2), it is advisable to use a small value of β to achieve a better average BTD. With the above guidelines in mind, but not aiming at optimizing the choice of parameters, we consider a version of SABA that employs reciprocal weight functions with $\alpha_{in} = 5$ and $\alpha_{ex} = 1$ for w_{in} and w_{ex}, respectively, and $\beta = 1$ (utility function associated with proportional fairness), and present simulation results for this case in the next section.

VI. Performance Gains under SABA: Fluid-flow Simulation

We conducted simulations to validate the performance gains of SABA over traditional fairness criteria, such as max-min, proportional, and potential delay fair bandwidth allocation [2]. The 'idealized' fluid-flow simulations exhibited in this section are discrete event simulations where events correspond to flow arrivals and departures and bandwidth allocations were *computed* (for the criterion being considered) and then *frozen* during inter-event periods. This cuts down simulation time significantly allowing a reasonable exploration of various system parameters.

We first consider a 5-link linear network where each link has capacity 10 Mbps, see Fig.2. We assume that flows arrive to each route according to Poisson processes with the same arrival rate and the flow size distribution is bounded Pareto with mean 5 KBytes. Fig.3 shows the average BTD achieved by SABA, the greedy algorithm proposed in §IV, and the three traditional bandwidth allocation policies. Observe that SABA and the greedy algorithm significantly outperform the three other policies, *e.g.,* a performance improvements of 58% when the links are 80% loaded. Note that the three traditional criteria result in similar performance with max-min having a slight advantage [5]. In the remainder of this section we will focus on max-min as our baseline criterion for performance comparisons on our network.

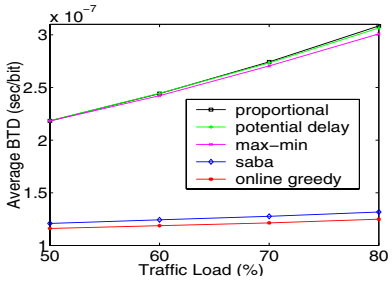

Fig. 3. Average BTD achieved by SABA, the greedy algorithm, and various fairness policies on a 5-link linear network.

We also conducted simulations for the symmetric star network shown in Fig.4, as well as a collection of random mesh topologies. Each of the random networks has 8 nodes and a 40% connectivity, *i.e.,* each node on average connects to 40% of other nodes, and link capacities are uniformly distributed between 10 to 20 Mbps. The star topology is arguably a good abstraction for networks with bottlenecks at the access points and transparent big backbone pipes, while the mesh ones might capture bottlenecks connecting peering points between domains. In both cases we assume the same bounded Pareto flow size distribution, Poisson arrivals with sources and destinations uniformly chosen among the access domains or mesh network nodes, respectively, and shortest hop routes.

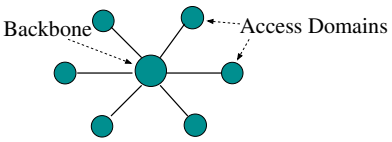

Fig. 4. A star network with six 10 Mbps access links.

Fig.5 shows the average BTD performance improvements achieved by SABA versus max-min fair bandwidth allocation on a star network as well as an average over 15 random topologies as the traffic load increases. In comparison to the linear network case, we observe a higher performance improvement for the symmetric star network and comparable improvements for the simulated random topologies, *e.g.,* 70%, 57%, and 58% for the star, random, and linear networks at 80% traffic load, respectively. These results substantiate the potential for performance gains that can be achieved by using SABA over traditional fairness criteria.

VII. Implementation of SABA: SAReno

The SABA criterion proposed in §IV-B requires that the allocated bandwidth *instantaneously* track the optimal solution to a network optimization problem which in turn depends on the changing network state. In practice, decentralized transport mechanisms, *e.g.,* TCP, can be designed to approximate

[5] Authors in [5] also suggested that various traditional fairness criteria achieve similar performance in the dynamic regime.

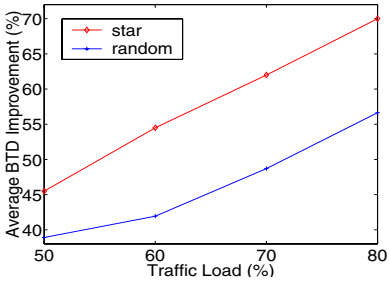

Fig. 5. Average BTD improvements achieved by SABA over max-min fair allocation on the star and random networks as the traffic load increases.

such idealized allocations, see *e.g.,* [1], [3], [4], [17]. Below we present a simple preliminary implementation of SABA by incorporating the residual flow size factor into the TCP Reno control mechanism.

A. SAReno: Reno with Size-based Differentiation

A key phase for TCP Reno is the congestion avoidance phase, where the user, *i.e.,* sender, additively increases his window size (transmission rate) when receiving acknowledgments and multiplicatively decreases it if network feedback indicates congestion, *e.g.,* time-outs or marked packets. The default linear increase rate and the multiplicative decrease ratio for TCP Reno are

$$\delta^{\text{Reno}} = 1/\text{cwnd} \quad \text{and} \quad \kappa^{\text{Reno}} = 0.5 \cdot \text{cwnd},$$

respectively, where cwnd is the current congestion window size in packets. In other words, assuming no loss every round trip time the cwnd increases by approximately 1 packet, while upon an indication of congestion it is reduced by a half.

To realize SABA we propose to modulate the 'additive increase rate' and the 'multiplicative decrease ratio' at the sender side by the residual flow size. We call this size-dependent user adaptation mechanism SAReno. We considered the case where each user only has access to his own residual flow size information, *i.e.,* SABA with $w_{\text{in}} = w_{\text{ex}}$. SAReno obtains the initial size of the transfer from the application layer and then keeps track of the residual size by monitoring the sequence numbers of the acknowledged packets. For simplicity, we quantize flow sizes into five regions, and define the residual size dependent linear increase rate and multiplicative decrease ratio associated with SAReno as in Table II. For example, a SAReno flow with 20

TABLE II

PARAMETERS FOR SARENO: δ (1/CWND), κ (CWND)

	range of $p(t)$ (packets)				
	$[0,10)$	$[10,50)$	$[50,200)$	$[200,10^3)$	$[10^3,\infty)$
$\delta(p(t))$	10	5	1	0.5	0.25
$\kappa(p(t))$	1.0	0.9	0.5	0.1	0.01

packets unacknowledged at time t has $\delta^{\text{SAReno}}(t) = 5/\text{cwnd}$ and $\kappa^{\text{SAReno}}(t) = 0.9$ cwnd. Recall that the key idea is that the fewer number of packets left to be acknowledged by the receiver, the more aggressive (larger δ and κ) a flow should be, and vice versa. Based on our experience, we note that one should not use values which are too small for the linear increase rate for flows with large residual number of packets to send. This is because one may still want large flows to increase quickly to achieve reasonable throughputs when no small flows are present. The decrease ratio however can be set very small for large flows allowing one to give 'aggressive' priority to small flows during congestion. In fact if flows with a small residual size, say 10 packets yet to be sent and/or acknowledged, do not back off, then they can quickly complete their transfers without individually experiencing re-occurring congestion. Note that this is a preliminary design of SAReno. One can certainly engineer the parameters, or implement SABA based on other versions of TCP or other transport protocols. Our intent here is to provide a proof-of-concept implementation that exhibit the benefits by employing the size dependent differentiation.

B. Performance Gains: SAReno vs. Reno

In this subsection we will present our simulation results comparing the average BTD achieved by SAReno versus Reno using the NS-2 [18] simulator. We focus on the star network considered in §VI with 1 ms propagation delay on each link. We assume the same arrival processes as those used in §VI. The flow size distribution is assumed to be bounded Pareto with mean 50 KBytes, *i.e.,* 100 packets with 500 bytes per packet. We selected a larger mean flow size than is currently typical of TCP transfers on the Internet, so as to exhibit the performance impacts of SAReno on larger flows, *e.g.,* bulk transfers, and ignoring extremely small ones which are likely to complete before entering the congestion avoidance phase.

Fig.6 shows the average BTD achieved by SAReno and Reno with various link packet scheduling mechanisms as the traffic load increases. We considered two different underlying packet scheduling disciplines: First Come Fist Serve (FCFS) and Deficit Round Robin (DRR) [19]. While FCFS is considered the 'default' discipline, we also present the results for DRR to show the performance benefits that would accrue if fair queuing and SAReno were employed together. As seen SAReno outperforms Reno for both the FCFS and DRR cases - by about 30-40% for a range of traffic loads. These improvements are significant, but less than those exhibited in the fluid-flow simulations presented in §VI. This is partly due to the fact that we have only introduced size-based differentiation in the congestion avoidance phase, which may not be reached by small flows. Then again perhaps the performance improvements for these types of flows are not as critical.

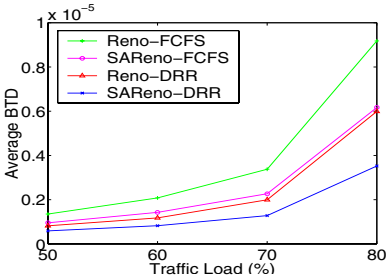

Fig. 6. Average BTD achieved by various Reno/SAReno and FCFS/DRR combinations when traffic load increases.

As seen in Fig.6 the average BTD performance achieved with

FCFS is much worse than that with DRR, for both Reno and SAReno. Networks supporting 'packet level' scheduling fairness will allow transport mechanisms such as SAReno to realize the size-dependent differentiation more quickly, and thus enhance the overall performance. A fair share packet scheduling mechanism, *e.g.,* DRR, exhibits good performance, but requires per-flow state and thus does not scale well when one needs to support a large number of ongoing flows, *e.g.,* on core routers. Further experiments on alternative topologies show that implementing this only at bottlenecks, *e.g.,* access points and or peering points, suffices to achieve these additional gains in performance.

Additional simulations of SAReno were carried out for the set of random mesh networks discussed earlier. We saw performance improvements over Reno ranging from 26-38% with 90% confidence intervals, *i.e.,* similar gains as those found for star networks.

C. Penetration Experiments

An interesting question is how SAReno and Reno flows might fare if they coexist on a network. We conducted preliminary simulations to determine how the performance benefits would vary with the penetration of SAReno flows increases. The simulations presented below are for the 6-branch star network.

In the first scenario the transfers to be mediated via SAReno rather than Reno were selected at random, *i.e.,* fairly homogeneous, according to the penetration level considered. Fig.7 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, SAReno flows will see better average BTD (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 suggest that users will have proper incentives to upgrade from Reno to SAReno.

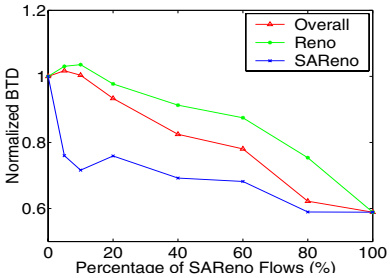

Fig. 7. Normalized BTD as the percentage of (random) SAReno flows increases.

Alternatively SAReno might not be deployed in the homogeneous manner discussed above, but in a more 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 thus examine the average BTD performance as one increases the number of access nodes on the star network that mediate transfers via SAReno. Fig.8 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). One can observe that when one deploys SAReno on a per-domain basis, it has a even quicker impact than the homogeneous random deployment scenario (Fig.7). This is to be expected since the intra-route discrimination can be more effective when SAReno flows originate from the same access domain.

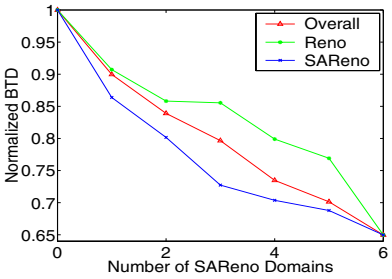

Fig. 8. Normalized BTD when one increases the number of domains that sends SAReno flows.

VIII. Future Work

We have shown that job-size dependent differentiation at the transport level for elastic flows provides an avenue for significant performance improvements, greater than 40 % , in the average user perceived BTD. These results are very encouraging, and suggest in our opinion that this line of research and possibly development warrants further exploration. We note that size-based differentiation can be also realized at the application layer. In fact [20] have proposed and developed an implementation of SRPT on web servers. However, implementing differentiation at the transport level, enables one to address network bottlenecks and/or interactions among various flows on the network resources rather than simply on the end systems. We are currently addressing the question of stability for bandwidth allocation based on residual flow sizes, though we note that the case where weights depend on initial flow sizes can be shown to be stable using the techniques in [5], [8]. Further interesting questions pertain to the impact of size-based differentiation versus fair sharing when traffic fluctuation leads to transient overloads, particularly with impatient users, see [21], [22].

Appendix

I. Proof for Lemma 1

Proof: We will prove a more general case where a predetermined 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 $A(t,\infty) = \{j | j \in J,\ a_j > t^*\}$. as the set of flows that arrive after time t^*. Consider the flow $k =$

$\text{argmin}_{j\in J\setminus A(t,\infty)}\{f_j\}$ 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 $j \in J \setminus A(t,\infty) \setminus k$ 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 $j \in J \setminus A(t,\infty) \setminus k$ 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 ongoing 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 $\tilde{p}_j^s(t)$, as defined in the text, and $\tilde{p}_j^s(t)$ 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

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

where $\tilde{p}_0^1(t) = 0$, 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 $j \in \{1,\ldots,n_0(t)\}$ and $k \in \{0,\ldots,n_1(t)\}$.

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 $k_j^*(t) = \text{argmin}_{k\in\{0,\ldots,n_1(t)\}}[\Delta b(j,k)]$. Simple derivation can show that $k_i^* \leq k_j^*,\ \forall i < j$ since $p_j^0 \cdot p_j^0(t)$ is non-decreasing in j. Thus there exist a set of $\{k_j^*,\ j = 1,\ldots,n_0(t)\}$ that minimizes the set of $\{\Delta b(j,k),\ j = 1,\ldots,n_0(t)\}$ 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. ■

IV. Proof for Fact 1

Proof: Let $\mathbf{x}$ be a feasible bandwidth allocation that only differs from the optimal $\mathbf{x}^*$ at $x_i = x_i^* + \Delta$ and $x_j = x_j^* - \Delta$ for some $\Delta \in \mathbf{R}$ where $r_i = r_j$. Considering the first order condition, we have

$$w_i \frac{x_i - x_i^*}{(x_i^*)^\beta} + w_j \frac{x_j - x_j^*}{(x_j^*)^\beta} \;\leq\; 0,$$

$$\Rightarrow\ \Delta\left(\frac{w_i}{(x_i^*)^\beta} - \frac{w_j}{(x_j^*)^\beta}\right) \;\leq\; 0.$$

Since this is true for either $\Delta > 0$ or $\Delta < 0$, we have that $\frac{w_i}{(x_i^*)^\beta} = \frac{w_j}{(x_j^*)^\beta}$. ■

V. Proof for Fact 2

Proof: We may re-write the left hand side of the per-flow basis first order condition as follows. For convenience, we suppress the time index writing w_j, x_j, p_j and J for

$w_j(t), x_j(t), p_j(t)$ and $J(t)$, respectively.

$$\begin{aligned}
\sum_{j\in J} w_j \frac{x_j - x_j^*}{(x_j^*)^\beta} &= \sum_{r\in R}\sum_{j:r_j=r} w_j \frac{x_j - x_j^*}{(x_j^*)^\beta}, \\
&= \sum_{r\in R}\sum_{j:r_j=r} w_j \frac{x_j - \left(\frac{w_j^{1/\beta}}{\sum_{k:r_k=r_j} w_k^{1/\beta}} y_r^*\right)}{\left(\frac{w_j^{1/\beta}}{\sum_{k:r_k=r_j} w_k^{1/\beta}} y_r^*\right)^\beta}, \\
&= \sum_{r\in R} \frac{(\sum_{k:r_k=r} w_k^{1/\beta})^\beta}{(y_r^*)^\beta} \left(\sum_{j:r_j=r} x_j - \frac{\sum_{j:r_j=r} w_j^{1/\beta}}{\sum_{k:r_k=r_j} w_k^{1/\beta}}\, y_r^* \right), \\
&= \sum_{r\in R} \Big(\sum_{j:r_j=r} w_j^{1/\beta} \Big)^\beta \left(\frac{y_r - y_r^*}{(y_r^*)^\beta} \right).
\end{aligned}$$

The per-route basis condition thus follows with the aggregate route weight for route r being $v_r = (\sum_{j:r_j=r} w_j^{1/\beta})^\beta = (\sum_{j:r_j=r} (w_{\text{ex}}(p_j))^{1/\beta})^\beta$. ■

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

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

[6] G. Fayolle et.al., "Best-effort networks: modelling and performance analysis via large network asymptotics," *Proc. IEEE INFOCOM*, 2001.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] S. Muthukrishnan et.al., "Online scheduling to minimize average stretch," *Proc. FOCS*, pp. 433–443, 1999.

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

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

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

[16] 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.

[17] J. Crowcroft and P. Oechslin, "Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP," *ACM Computer Communication Review*, vol. 28, pp. 53–67, 1998.

[18] "Network Simulator (NS-2)," *http://www.isi.edu/nsnam/ns/*.

[19] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *Proc. ACM SIGCOMM*, 1995.

[20] M. Harchol-Balter et.al., "Implementation of SRPT scheduling in web servers," *Proc. 7th Annual Workshop on Job Scheduling Strateties for Parallel Processing*, pp. 11–35, 2001.

[21] T. Bonald and J. Roberts, "Performance modelling of elastic traffic in overload," *Proc. ACM SIGMETRICS*, pp. 342–343, 2001.

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