The Dissertation Committee for Shan-Chieh Yang
Certifies that this is the approved version of the following dissertation:

Flow-size Based Differentiation to Enhance User Perceived Performance on Networks Supporting Best-effort Traffic

Committee:

Gustavo de Veciana, Supervisor

Harrick M. Vin

Vijay K. Garg

Ross Baldick

San-Qi Li

Flow-size Based Differentiation to Enhance User Perceived Performance on Networks Supporting Best-effort Traffic

by

Shan-Chieh Yang, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2001

Dedicated to my parents, my sister, and Yaping.

Acknowledgments

I wish to thank the multitude of people who have been furthering me. In particular, I would like to extend my most sincere appreciation to my advisor Dr. Gustavo de Veciana for his devoted guidance throughout my work. Working with Dr. de Veciana has been an invaluable and delightful experience, and his keen advice has helped me advance, both professionally and personally. My thanks also go to Dr. San-Qi Li, Dr. Harrick Vin, Dr. Ross Baldick, and Dr. Vijay Garg for serving on my committee and sharing their wisdoms.

I am indebted to my colleagues Xun Su, Steven Weber, Xiangying Yang, Jangwon Lee, John Stine, Na Li, Aimin Sang, Sangkyu Park, Kevin Yang, Philip Girolami for their enthusiastic and insightful comments. I am also grateful to Tae-jin Lee, Chin-fong Su, and Michael Montgomery for sharing their experience with me in my early stage of graduate study.

Special acknowledgments also go to my friends in Austin, Taiwan, and other places for reaching their hands and giving their support. Most of all, I would like to express my sincere thanks to my parents, my sister, and Yaping for their everlasting love, support, and encouragement. This dissertation is dedicated to them.

Flow-size Based Differentiation to Enhance User Perceived Performance on Networks Supporting Best-effort Traffic

Publication No. _____

Shan-Chieh Yang, Ph.D.
The University of Texas at Austin, 2001

Supervisor: Gustavo de Veciana

A fundamental, yet usually ignored, challenge to evolving data networks is enabling better Quality of Service (QoS) for best effort traffic. While over-provisioning network resources was considered to be a simple, possibly less expensive solution, the lack of fairly predictable Internet traffic and growth model makes it almost impossible to ensure an "efficiently" dimensioned network. Even if a single carrier could overcome this problem and appropriately provision his network, the heterogeneity of the inter-networking infrastructure would make coordinating inter-carrier agreements to avoid mismatches exceedingly difficult. Recognizing that over-provisioning is at most a partial solution, we approach the problem by "intelligently" share resources that extends the "dynamic range" of traffic loads where networks can ensure adequate system and user perceived performance.

Best effort service involves for the most part transfers of files for which the transmitter has a-priori knowledge of the file's size. Two plausible QoS measures for such transfers are delay and bit transmission delay (BTD), i.e., delay/file size,

among which the latter captures more "savvy" users who recognize that big files take longer to transfer. In this talk I will discuss how one might design size-based transport mechanisms and routing algorithms to enhance the overall network and user performance in a cooperative environment. Our view is that by simply introducing such size information, one may produce a "robust" best effort network that achieves better performance for all transfers for a range of traffic loads, even with the existence of impatient users.

Table of Contents

List of Tables

List of Figures

List of Notation

L	$L = \{l\}$, set of links
R	$R = \{r\}$, set of routes
J	$J = \{j\}$, set of flows
A	$A = \{A_{lr}\}$, link-route incidence matrix
$J(t)$	active set of flows at time t
c_l	capacity of link l
a_j	arrival time of flow j
r_j	route assigned to flow j
p_j	overall transfer size of flow j
$p_j(t)$	residual transfer size of flow j at time t
$\tilde{p}_j^s(t)$	total residual work of flows to be completed on route s prior to completing flow j at time t
d_j	overall time to complete transfer of flow j
b_j	$b_j = d_j/p_j$, bit transmission delay perceived by flow j at the end of the transfer
$x_j(t)$	bandwidth allocated to flow j at time t
$y_r(t)$	aggregate bandwidth allocated to all flows on route r at time t
$w_j(t)$	weight of flow j at time t
$U_\beta(x)$	utility function in terms of bandwidth allocation x parameterized by β
α	parameter determining the impact of a flow's residual size to its weight
β	parameter determining the impact of the allocated bandwidth to a flow's utility
$v_r(t)$	aggregate weight of flows on route r at time t
$w(s,t]$	cumulative work completed during time interval $[s,t]$
$e_c(t)$	minimum cumulative service curve: minimum amount of work that needs to be completed after t units of time since a transfer was initiated
$e_p(t)$	minimum progress service curve: minimum amount of work that needs to be completed during any time interval of length u

Chapter 1

Introduction

As evidenced by traffic loads on the Internet, exchanging data files has been, and will continue to be, a major fraction of the volume carried by data networks. It has been reported that about 95% of the bytes, 90% of the packets, or 80% of the flows on the Internet are made up by TCP mediated file transfers [48]. Although such transfers are referred to as 'best effort' traffic, there has been increasing interest in enhancing user perceived performance particularly, but not exclusively, when interactivity is involved, *e.g.,* web browsing. The traditional approach for achieving good performance on best effort network has been provisioning more bandwidth. Overprovisioning might be arguably a less expensive option than deploying advance technologies on IP-based networks. It is itself however not a simple task to achieve 'efficient' overprovisioning, requiring, among other things, fairly predictable traffic and growth models. Recent experience has shown that aggregated data traffic may exhibit long-range dependency, not to speak of daily non-stationeries as well as bursty growth on longer time scales [46]. Even if a single carrier could overcome this problem and appropriately provision his network, the heterogeneity of the inter-networking infrastructure would make coordinating inter-carrier agreements to avoid mismatches exceedingly difficult. Moreover, just as 'nature abhors a vacuum', one should recognize that highly overprovisioned resources will quickly be

filled by increasing user demands or simply 'garbage' traffic. Together, these arguments suggest that a better user perceived performance, or quality of service (QoS), associated with file transfers is not likely to be based on overprovisioning alone. Our thesis is that it should be complemented by mechanisms to achieve efficient resource allocation that extend the dynamic range of traffic loads data networks can support.

A file transfer typically corresponds to moving a known amount of data and is said to be *elastic* in the sense that it has a wide tolerance to changes in the transmission rate during the transfer [51]. Since elastic flows/users come and go, and possibly share the network with other types of traffic, the available network resources may vary during their lifetimes. In turn, elastic flows can adapt their transmission rates to dynamically share available resources with contending transfers. From a users' point of view, a reasonable performance measure may be the time it takes to complete the transfer. Transfer delays alone however 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

[1] Studies show that the files being transferred on the Internet exhibit great variability in size, see *e.g.,* [14, 46].

[2] The BTD metric is similar to 'slow-down' proposed in scheduling research community, with an additional notion of resource capacity, see *e.g.,* [5, 22, 24].

with increasing the average perceived throughput. Other performance metrics could and may be considered in the future. In this thesis, we however focus on the average BTD or delay as reasonable proxies for user perceived performance.

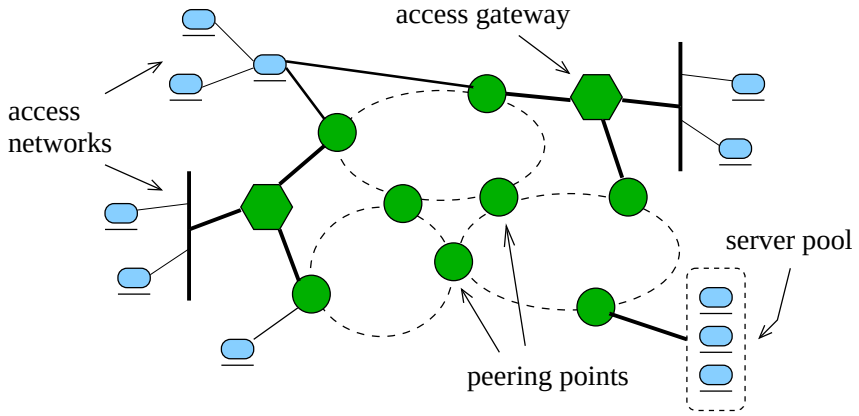


Figure 1.1: A data network abstraction.

The BTD or delay associated with a file transfer clearly depends on what and how network resources are shared among contending flows. As much traffic may traverse beyond local network domain, it is difficult, or even impossible, for local service provider or servers to identify and properly dimension the bottleneck(s), even when they wish to cooperate to establish mutually beneficial service level agreements. For example, Fig.1.1 shows a typical inter-connection of local area networks with high capacity links, *e.g.,* OC-3, OC-12, or 100 Mbps/Gigabit Ethernet, possibly accessible to end users, depending on their end devices, *e.g.,* 28/54 Kbps modems or 10/100 Mbps Ethernet cards. Traffic from these local networks might then be carried by backbones with more than enough capacity, and/or constrained by small peering points connecting transit domains. Other scenarios may also exist and present even more possibilities for bottlenecks to arise. Moreover, by definition, the bottleneck

resource that limits a long lasting transfer may change over time as the traffic pattern changes. Thus to ensure adequate performance for file transfers, we propose to investigate efficient mechanisms that dynamically share network resources, possibly based on changing network status, so as to enhance, if not optimize, user perceived performance.

Several network mechanisms, such as routing, caching, congestion control, queue management, and packet scheduling, which operate possibly independently with limited information exchanges, together determine the bandwidth sharing dynamics on a network and thus the user perceived performance. On the one hand, caching and routing schemes determine which network resources will be used to realize a file transfer and thus which flows will contend for network resources. On the other hand, mechanisms such as the congestion control (TCP), queue management (Random Early Detection - RED), and packet scheduling (First Come First Serve - FCFS, or Weighted Fair Queueing - WFQ) together determine how resource capacity is shared among ongoing flows. A degree of freedom that is usually ignored by current network protocols is the 'malleability' of file transfers. More specifically, users realizing file transfers are typically concerned with the transfer's progress over a perceivable time scale, *e.g.,* what percentage of the file has been transferred over the past 5 seconds, rather than the 'instantaneous' transmission rate. This offers more flexibility to network mechanisms as a means to differentiate among ongoing transfers based on their sensitivity, or sense of urgency, to progress, and to allocate network resources accordingly. For example, downloading a 10 MBytes image in 10 seconds may be great versus transferring a 5 KBytes text message in 10 seconds.

Thus one might choose to first allocate more bandwidth to finish the small text message, and then work on transferring the big image file. Note that upon initiating a file transfer the amount to be sent is typically known on the sender side, *e.g.*, static web content. In this thesis we investigate how one might employ known file size information in various network mechanisms so as to enhance the 'overall' user perceived performance.

Achieving good performance on networks supporting file transfers does not only depend on the network protocols, but may also depend on how users respond to the transfer experienced. For example, it is not unusual for users to interrupt a transfer as a result of, say, impatience or perceived poor performance. A recent study [19] claims that 11% of the TCP mediated file transfers were interrupted, corresponding to 20% of the total traffic volume, *i.e.,* 20% of the transmitted data went to waste. This high percentage of the wasted transmission may not be that surprising given the excessively bursty, non-stationary, and growth patterns exhibited on today's Internet. In fact we believe that even carefully provisioned networks will see temporarily and eventually systematic overloads until they are re-provisioned. It is however unclear how users respond to congestion or overloads. Do they interrupt based on the time they have been waiting? Are impatient users typically aware of the file size of the transfer or the speed of their transmission media? To our knowledge, there has been no empirical evidence investigating the possible (types of) user behavior that result in such aborted transfers. This thesis propose this as an important research agenda to be pursued in the sequel. From the network's point of view, the question is how various resource allocation mechanisms will fare when

users exhibit various possible types of impatient behavior. We take one step in this direction by investigating the impact of plausible user behaviors on both the system and user perceived performance under various bandwidth allocation policies.

For analysis purposes, we decouple the complex dynamic resource allocation problem into two sub-problems, bandwidth allocation and routing. A formal network model and problem statement is presented in Chapter 2. In considering the bandwidth allocation problem, we assume the path each file transfer traverses is given and fixed throughout its lifetime. The primary analytical results for bandwidth allocation are presented in Chapter 3. Several practical considerations and a simple implementation of a size dependent bandwidth allocation scheme based on TCP Reno are illustrated in Chapter 4. A set of generic user impatience models and an investigation of their impact on performance is presented in Chapter 5. We will then focus on the routing problem, *i.e.,* fix the underlying bandwidth allocation scheme and consider various size dependent routing policies, and present our analysis in Chapter 6. Concluding remarks and future research directions are given in Chapter 7.

Chapter 2

Network Model

Throughout this dissertation we will consider the following fluid flow network model and use the corresponding notation unless noted otherwise.

We model a network with a set of nodes N and a set of links L, where each link $l \in L$ connects two nodes and 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 from a source to a destination node. The network ***routing*** scheme will assign each transfer request j a route r_j, corresponding to a sequence of links connecting source to destination, upon its arrival. We assume that once a flow is assigned to a route it will stick to the same route until it completes. [1] We denote the set of possible routes by R. The set of links traversed by route $r \in R$ is captured by a 0-1 matrix A where A_{lr} is 1 if router r traverses link l and 0 otherwise.

Once the route has been selected, a ***bandwidth allocation*** policy will determine the bandwidth allocated to all ongoing transfers. We assume that whenever the network state changes, *e.g.,* there is an arrival or departure, the bandwidth allocation policy will dynamically and instantaneously adjust the assigned bandwidth to every ongoing flow. This assumption may be reasonable when the time scale

[1] On the Internet routes are typically 'stable', *i.e.,* for the most part a flow associated with a given transfer follows the same route [45].

between arrivals and/or departures are significantly larger than the speed at which the bandwidth allocation mechanism converges. transfer converges - depending on how actual network mechanisms operate. 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. Note that in the context of serving file transfers, there is no queuing or buffering in the network, *i.e.,* every flow must be assigned a route and a positive bandwidth upon arrival.

The time to complete a 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. Clearly the bandwidth allocated to a flow over time depends on both the bandwidth allocation policy and the routing scheme. Thus the perceived performance of a flow over its lifetime depends on both types of network controls. As mentioned earlier we will focus on the ***Bit Transmission Delay (BTD)*** as the performance measure of interest where the BTD for flow j is given by $b_j = d_j/p_j$. We summarize our notation in Table 2.1 and formally state our primary objective below.

Table 2.1: Notation Summary

Set of Links: L	capacities $c_l, l \in L$
Set of Routes: R	link-route incidence matrix A_{lr}
Set of Flows: 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, $b_j = d_j/p_j$, $\sum_{j \in J} b_j$
	delay, BTD, avg BTD

Objective:

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

This can be used to determine a bandwidth allocation policy, $\mathbf{x}$, as above, or to determine a mapping for each flow j onto a route r_j or both. 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 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 policy. We refer to algorithms that minimize the average BTD for arbitrary arrivals as BTD-optimal.

Chapter 3

Size-based Adaptive Bandwidth Allocation

3.1 Introduction

Recently much attention has been paid to characterizing how network bandwidth is shared among file 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, see [26, 31, 32, 36, 39, 41] and references therein. 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 [7, 10, 17, 20, 49] leads the way in this direction focusing on stability and performance in the dynamic regime. Our aim in this study is to investigate how one might enhance, if not optimize, average user perceived BTD performance from the ground up through a new class of bandwidth allocation policies, while assuming that the routes associated with each flow are pre-determined and fixed throughout the flow's lifetime, *i.e.,* no routing decisions are involved in minimizing the average BTD.

The work in [7, 10, 17, 20, 49] considered stochastic models that capture the dynamic behavior of *existing* network mechanisms that mediate file transfers, *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 [24]. 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 [22]. 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 [4, 22, 49]. However, to our knowledge, no systematic allocation criterion and associated transport mechanism have been proposed to enhance user perceived performance on a network. In this study we take a novel approach by investigating how one might employ the file size information for a bandwidth allocation policy so as to optimize the average BTD users will see.

Due to the inherent difficulty of this problem, we will begin by considering the dynamic bandwidth allocation 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 average '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 jobs in a BTD-optimal manner. The insights derived from the transient regime analysis will then be used to construct a

new class of bandwidth allocation criteria which will be shown to exhibit significant performance improvements over traditional fairness criteria.

This chapter is organized as follows. In §3.2 we consider a dynamic set of flows sharing a single bottleneck link, and derive/compare policies that enhance the average BTD over the commonly used 'fair sharing' discipline. Then, in §3.3, 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. Design options for SABA are discussed in §3.4. Fluid-flow simulations are presented in §3.5 to exhibit the potential performance gains achieved by using SABA over traditional fairness criteria. Conclusions and final remarks are given in §3.6.

3.2 Optimizing Average BTD: The 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.

3.2.1 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 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 3.2.1. *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.

3.2.2 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) [50], and was recently shown to be 2-competitive for the average BTD metric [22]. 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 3.2.1. *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 average 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 3.2.1 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.

3.2.3 Performance gains of size-dependent differentiation

Recently [4] 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, versus 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 [15, 30]. Fig.3.1 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 found that 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.

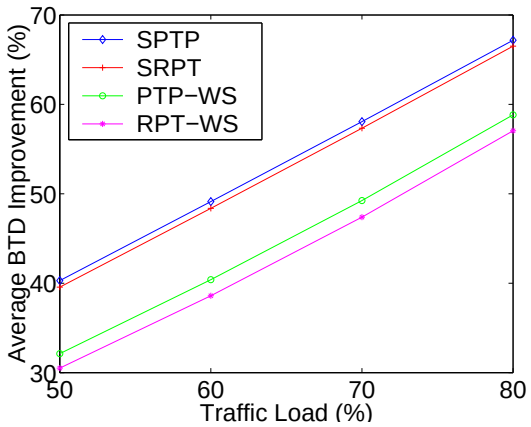


Figure 3.1: The average BTD improvement achieved by SRPT/SPTP/RPT-WS/PTP-WS over fair sharing when traffic load increases.

Fig.3.2 further exhibits the average BTD performance of RPT-WS (PTP-WS) as α varies for a link with an 80% traffic load. Note that abscissa corresponds to the α on a logarithmic scale. Clearly for α sufficiently large the performance achieved by RPT-WS and PTP-WS is fairly close to those of SRPT and SPTP. One can observe that our dynamic weighted bandwidth allocations quickly become superior to fair sharing as α increases.

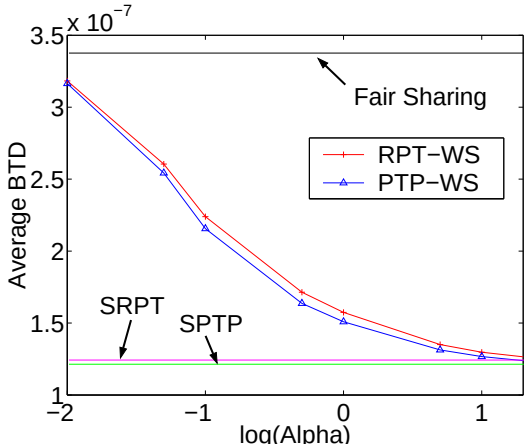


Figure 3.2: The average BTD achieved by RPT-WS/PTP-WS as compared to that under fair sharing and that under SRPT/SPTP when α increases at 80% traffic load.

3.2.4 Remarks about size-dependent differentiation

We have explored various bandwidth allocation approaches that favor small flows, as an avenue towards minimizing the average BTD for the single link case. More complex policies may be considered. For example, Bender *et.al.*[5] consider a Dynamic Earliest Deadline First Policy that attempts to estimate the completion times associated with serving each flow. Observing that all these policies, including SRPT and SPTP, commonly aim at finding the flow that can finish the fastest and incur least overhead to other flows, we believe that only marginal performance differences will be exhibited there[1].

A criticism brought against these SRPT-like polices is their potential to induce starvation for large flows. We will refer the afore-mentioned size-dependent

[1]Reader may refer to [15] to see an argument comparing SRPT with the Dynamic Earliest Deadline First policy.

differentiation schemes as SD, as opposed to FS for fair sharing, since they achieve similar performance from the starvation perspective. Studies in [4] and [47] 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 [14, 46]. Our own experiments also confirm that starvation is indeed not a concern, particularly in comparing with FS. In fact, we found that for most cases (various simulation runs with different flow size distributions), SD even achieves better performance in terms of the standard deviation for the BTD, the maximum BTD, and the average BTD for the largest 1% flows. With few exceptions, consisting of only 5% of our simulation runs, we see SD exhibits worse maximum BTD, but with the same order of magnitude as FS. These exceptions are mostly for the cases where the flow size distribution is exponential, as opposed to bounded Pareto or deterministic one, and/or the traffic load is close to 100%.

In order to analyze the starvation problem encountered by 'all' flows, we further propose to examine the percentage of flows that experience starvation. We shall define starvation as flows that see more than 10 times of the best possible BTD they could realize on the link. Table 3.1 summarizes the results achieved by FS and SD schemes with traffic loads of 80% and 90%, respectively. As seen, SD performs significantly better than FS under all scenarios. In particular when the flow size distribution is bounded Pareto under SD, we hardly see flows ($< 0.01\%$) that experience starvation.

Table 3.1: Percentage of flows experience starvation

Traffic Load	Bounded Pareto	Exponential	Deterministic
80%	FS(5%), SD(0.001%)	FS(10%), SD(0.6%)	FS(10%), SD(4%)
90%	FS(15%), SD(0.01%)	FS(36%), SD(2.2%)	FS(37%), SD(2.1%)

3.3 Optimizing Average BTD: The 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.3.3 including m links with the same capacity c, and m short and 1 long route. Even if there were a flow with a small residual size on

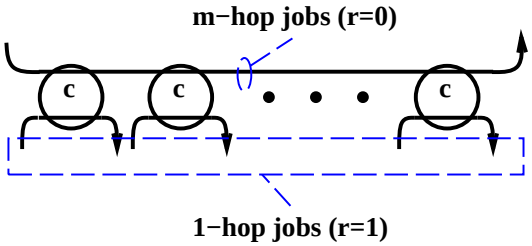


Figure 3.3: Linear network: m equal capacity links.

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.

3.3.1 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 §3.2. 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 3.2.1 for flows sharing a single link, one can show a 'no-sharing' result for the symmetric linear network.

Lemma 3.3.1. *A BTD-optimal inter-route bandwidth allocation* $\mathbf{y}^*$ *for flows on a symmetric linear network (see Fig.3.3) is such that at any time* t *either* $y_0^*(t) = 0$ *or* $y_0^*(t) = c$.

Combining the necessary condition in Lemma 3.3.1 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 s prior to completing flow j at time t. We assume ties are broken arbitrarily. Now we may present our policy.

Algorithm 3.3.1 (Greedy Algorithm for Linear Network). *At any time t, $y_0^*(t) = c$ and $y_1^*(t) = 0$ if*

$$\underbrace{\frac{1}{p_1^0} \cdot \frac{c}{p_1^0(t)}}_{\substack{\textit{max-wgt-thruput} \\ \textit{(m-hop route)}}} > \underbrace{\max_{k=1,2,\cdots,n_1(t)} \left[(\frac{1}{k} \sum_{l=1}^{k} \frac{1}{p_l^1}) \cdot (\frac{k \cdot c}{\tilde{p}_k^1(t)}) \right]}_{\substack{\textit{max-weighted-throughput} \\ \textit{(1-hop routes)}}} \tag{3.1}$$

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

Theorem 3.3.1. *At each point in time the Algorithm 3.3.1 minimizes the overall residual BTD for flows on the linear network shown in Fig.3.3, assuming that SPTP is the intra-route policy and there are no future arrivals.*

Despite its lengthy proof (given in the appendix), (3.1) in Algorithm 3.3.1 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 $p_1^0(t)/c$ 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[2] - *i.e.,* the left hand side of (3.1). 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 (3.1). 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.

3.3.2 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 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

[2]This fact also implies why SPTP is the greedy policy for the single link case.

realization, below we propose a general class of bandwidth allocation criteria where per-user weights that depend on residual sizes are considered. Following [31, 39, 41] we define a class of size-dependent adaptive bandwidth allocations (SABA) as follows.

Definition 3.3.1 (Size-based Adaptive Bandwidth Allocation). *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,*

$$\begin{aligned} \mathbf{x}^*(t) &= \arg\max_{\mathbf{x}(t)} \sum_{j \in J(t)} w_j(t) \cdot U_\beta(x_j(t)) \\ &\qquad s.t. \sum_{j \in J(t)} A_{lr_j} x_j(t) \leq c, \quad \forall\, l \in L, \end{aligned}$$

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

$$U_\beta(x) = \begin{cases} \log\ x & \beta = 1, \\ (1-\beta)^{-1} \cdot x^{1-\beta} & \beta \geq 0 \ \ and \ \ \beta \neq 1. \end{cases} \tag{3.2}$$

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 (3.2) with *fixed* weights have been widely considered [26, 31, 32, 36, 39, 41].

Notice that maximizing such overall user utility functions subject to resource constraints naturally favors flows that use fewer resources, therefore achieving higher service parallelism. As has been indicated in [7, 41], increases in β, and thus changes in the utility function to make it grow less rapidly with x, result in less discrimination against flows using more resources and reduce the impact of the weights. Table 3.2 exhibits the trends by showing several well known criteria associated with different βs assuming fixed weights. Note that as $\beta \to \infty$, the criterion becomes max-min rather than weighted max-min fair, *i.e.,* the weights have no effect on the bandwidth allocation. Our premise here is that the introduction of per-flow weights

Table 3.2: Impact of β on bandwidth allocation.

β	Criterion
0	Weighted Maximum Throughput
1	Weighted Proportional Fairness
2	Weighted Potential Delay Fairness
$\to \infty$	Max-min Fairness

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.

3.4 Role of Residual Size Dependent Weights

The challenge in 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

$$w_j(t) \quad = \quad w_{\text{in}}(p_j(t)) \cdot \left[\frac{\sum_{k \in J_{r_j}(t)} (w_{\text{ex}}(p_k(t)))^{1/\beta}}{\sum_{k \in J_{r_j}(t)} (w_{\text{in}}(p_k(t)))^{1/\beta}} \right]^{\beta}, \tag{3.3}$$

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[3].

Notice that by using the same function for w_{in} and w_{ex}, the weight of each flow based on (3.3) 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}.

3.4.1 Decoupling Intra and Inter-route Bandwidth Allocation

An important character of our proposed weight function in (3.3) 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}. Theorems 3.4.1 and 3.4.2 below characterize how bandwidth would be allocated among flows sharing the same route and across routes– proofs are given in the appendix.

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

Theorem 3.4.1. *At any time t the SABA bandwidth allocation for two flows, i and j, sharing the* same route *satisfies*

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

Theorem 3.4.2. *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

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

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 §3.2 and 3.3 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.

3.4.2 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.,*

$$w(p_j(t)) \;=\; e^{-\alpha p_j(t)} \quad \text{and} \quad w(p_j(t)) \;=\; \frac{1}{(p_j(t))^{\alpha}}, \tag{3.4}$$

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}.

Despite the fact that both favor flows with smaller residual size, the two functions shown in (3.4) differ in the way they discriminate. We shall use a simple example to exhibit this difference. Consider the intra-route bandwidth allocation characteristics shown in Theorem 3.4.1. By using the two different weight functions, we get

$$\frac{x_i^*}{x_j^*} \;=\; e^{-\frac{\alpha_{\text{in}}}{\beta}(p_i - p_j)} \quad \text{and} \quad \frac{x_i^*}{x_j^*} \;=\; \left(\frac{p_i(t)}{p_j(t)}\right)^{-\alpha_{\text{in}}/\beta},$$

respectively. This means that by using the exponential weight function, one will discriminate based on the 'absolute' difference in residual flow sizes, while the reciprocal weight provides an alternative which depends on the ratio of the residual sizes. In general, for a given α a more aggressive discrimination is achieved by using the exponential rather than the reciprocal weight function. Note that above discussion is for the bandwidth allocation in the static regime, *i.e.,* at a fixed time t. However the dynamics associated with SABA provide a further means of differentiation.

3.4.3 Dynamic Character of SABA

The above discussions focused on how SABA allocates bandwidth among flows sharing the network at a fixed time, but the dynamic nature, *i.e.,* dependence on the residual flow size, also plays a role. In particular if a flow is given priority, *i.e.,* more bandwidth, then its residual size is likely to go down quickly, and hence, its weight or priority to increase further. For example, assuming that $w_{\text{in}} = w_{\text{ex}}$ and considering the two weight functions in (3.4), the change of a flow j's weight in time can be expressed by its first derivative, *i.e.,*

$$\frac{dw_j(t)}{dt} = \alpha x_j^*(t) w_j(t) \quad \text{or} \quad \frac{dw_j(t)}{dt} = \alpha \frac{x_j^*(t) w_j(t)}{p_j(t)},$$

respectively. Note the additional factor of $1/p_j(t)$ for the dynamics associated with the reciprocal weight function. It is an open question, given the differences in how they differentiate based on residual size in both the static and dynamic regimes, which weight function provides a better average BTD. Nevertheless, for both cases, 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.

3.4.4 Design Options of SABA

Let us summarize the design options for SABA that achieves good average BTD performance, *i.e.,* the type of function used for w_{in} and w_{ex}, the values of α_{in} and α_{ex}, and the value of β. 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, 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 $\alpha_{\text{in}} = 5$ and $\alpha_{\text{ex}} = 1$ 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.[4]

3.5 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 [39]. 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* and then *frozen* during inter-event periods, according to the bandwidth allocation criterion under consideration. This cuts down simulation time significantly allowing a reasonable exploration of various system parameters.

[4]Throughout this dissertation, we will use this design option for all our subsequent discussions of SABA.

3.5.1 Linear Network

We first consider a 5-link linear network where each link has capacity 10 Mbps, see Fig.3.3. In this case we can further compare the SABA to the greedy algorithm suggested in §3.3. 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.3.4 shows the average BTD achieved by SABA, the greedy algorithm, and the three traditional bandwidth allocation policies. Observe that both SABA and the greedy algorithm significantly outperform the three fair bandwidth allocation policies. In fact they achieve similar performance improvements of up to 58% when the links are 80% loaded. Note also that the three traditional criteria result in similar performance with max-min having a slight advantage over the other two[5]. For succinctness and simplicity, in the remainder of this section we will focus on max-min as our baseline criterion for performance comparisons on our network.

3.5.2 Star and Random Networks

We have also conducted simulations for additional representative networks, including the symmetric star network shown in Fig.3.5. The star network is arguably a good abstraction of current Internet where the bottlenecks reside at the source and/or destination access links with transparent big backbone pipes [7]. In other words, the nodes on the spokes can be associated with access domains (both source

[5]Authors in [7] also suggested that various traditional fairness criteria achieve similar performance in the dynamic regime.

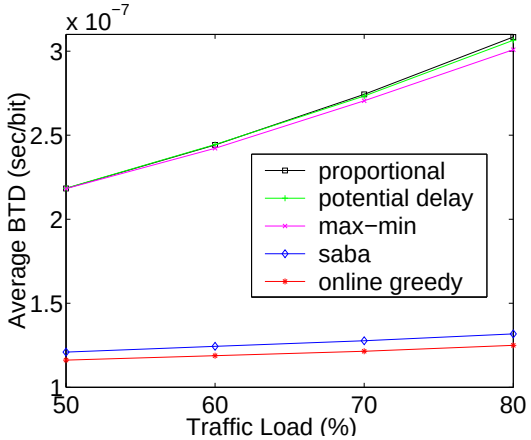


Figure 3.4: The average BTD achieved by SABA, the greedy algorithm, and various fairness policies on a 5-link linear network.

and destinations), while the center node represents a high capacity backbone. Again we shall assume the same bounded Pareto flow size distribution, and Poisson arrivals with sources and destinations uniformly chosen among the access domains, *i.e.,* no flow begins or ends at the center node.

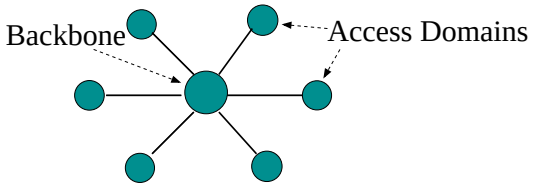


Figure 3.5: A star network: 6 10 Mbps access links.

We further consider a simple collection of random mesh topologies with 8 nodes and a 40% connectivity, *i.e.,* each node on average connects to 40% of other nodes, and where link capacities are uniformly distributed between 10 to 20 Mbps. We assume the arrivals are Poisson with source and destination randomly chosen among the 8 nodes, and the routes associated with each arrival are determined

based on a shortest hop routing algorithm. These topologies are meant to capture the scenario where data transfers may traverse multiple domains (nodes), and the bottlenecks reside on peering points. The value of capacities we assumed for both the star and random networks are for simulation purposes. We however believe that our results are representative of the performance gains that would be achieved.

Fig.3.6 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

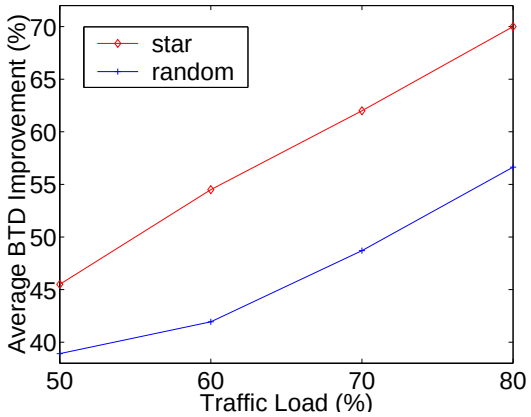


Figure 3.6: The average BTD improvements achieved by SABA against max-min fair allocation on the star and random networks as the traffic load increases.

network case, we observe a higher performance improvement for the symmetric star network and a 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.

3.5.3 Remarks on Starvation

We have also examined the set of starvation metrics described in §3.2 along with the simulations above. As in the single link case, SABA performs well in terms of starvation metrics as compared to traditional fairness criteria for networks. One difference is that there are more occasions, about 20% of the total simulation runs (as compared to 5% for the single link case), where SABA incurs larger maximum BTDs than, say, under max-min fairness. We believe this is due to large flows on long routes which are doubly penalized and thus experience very large BTDs. For example, in examining the starvation problem for flows (recall the metric we proposed in §3.2) we see significantly lower percentage of starving flows under SABA (about 0.13%) than under max-min (12%) at 80% traffic load. The above results confirm that SABA will in fact also perform better than traditional fair bandwidth allocation in terms of starvation.

3.6 Conclusion

In this chapter, we have proposed a new set of dynamic bandwidth allocation criteria that makes use of the residual flow size to improve user perceived performance. In particular we have focused on reducing the average BTD seen by users, and thus effectively also increased the user perceived throughputs. Recognizing that the difficulty in analyzing performance for networks sharing bandwidth in a dynamic fashion we have considered simple examples and derived greedy policies that perform close to optimal. These examples motivate two natural properties that one would like to achieve in allocating bandwidth to reduce the average BTD: (1)

favor small flows among those sharing the same route and, (2) tradeoff favoring small flows and achieving high service parallelism when allocating bandwidth across routes. We proposed a new class of bandwidth allocation criteria, SABA, which depends on not only the number of flows on each route but also the residual file sizes. Our fluid-flow simulations showed that one could achieve a 70% average BTD performance improvement by using SABA over traditional fairness criteria at 80% traffic load.

These are encouraging performance gains providing significant evidence on the possible benefits of size-dependent differentiation, suggesting in our opinion that this line of research and possibly development warrants further exploration. In the next chapter we will provide a simple transport layer implementation based on TCP Reno, and exhibit its performance gain versus Reno. We note that size-based differentiation can be also realized at the application layer. In fact [21] have proposed and developed an implementation of SRPT on web servers. However, implementing differentiation at the transport layer enables one to address network bottlenecks and or interactions among various flows on the network resources rather than simply on the end systems. An open question is the stability condition for bandwidth allocation based on residual flow sizes. We note that allocation based on initial flow sizes can be shown to be stable via the same method as in [7, 17]. 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. We will discuss this issue in Chapter 5.

3.7 Proofs of Theorems

3.7.1 Proof for Lemma 3.2.1

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 $k = \text{argmin}_{j \in J \setminus A(t,\infty)}\{f_j\}$ 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. □

3.7.2 Proof for Theorem 3.2.1

According to Lemma 3.2.1 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

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

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. □

3.7.3 Proof for Lemma 3.3.1

First it should be clear that one needs only consider bandwidth allocations that always allocate the same bandwidth to all 1-hop routes. Otherwise, as long as there is an 1-hop route that has at least one active flow but receives less bandwidth than some other 1-hop route at some time t, one can always improve the active flow's delay (BTD) without compromising others' performance by increasing its allocated bandwidth to be the same as others. Furthermore, for a given bandwidth allocation, one may consider the aggregate bandwidth allocated to any route as a 'pre-determined time-varying' capacity shared by the flows on that route. Thus by Lemma 3.2.1, we know that one can always improve the overall BTD if jobs on the same route are not served one by one. Based on the above arguments, we only need to show that "one can always improve the overall delay (BTD) upon any bandwidth allocation that concurrently allocates positive bandwidth to one m-hop flow and at least one 1-hop flow". The rest of the proof is essentially the same as the arguments given in Lemma 3.2.1, with a new twist that the 1-hop flows, if there are more than one, see changes in the same manner. □

3.7.4 Proof for Theorem 3.3.1

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 3.3.1) 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

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

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. □

3.7.5 Proof for Theorem 3.4.1

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 $\Delta \in \mathbb{R}$ where $r_i = r_j$. Considering the first order condition, we have

$$\begin{aligned} w_i \frac{x_i - x_i^*}{(x_i^*)^\beta} \ + \ w_j \frac{x_j - x_j^*}{(x_j^*)^\beta} \ &\leq \ 0, \\ \Rightarrow \quad \Delta \left(\frac{w_i}{(x_i^*)^\beta} \ - \ \frac{w_j}{(x_j^*)^\beta} \right) \ &\leq \ 0. \end{aligned}$$

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

3.7.6 Proof for Theorem 3.4.2

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.

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

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

Chapter 4

Towards Practical Implementation of SABA

4.1 Introduction

The study presented in Chapter 3 assumes an ideal scenario where the allocated bandwidth *instantaneously* tracks the optimal solution to a network optimization problem which in turn depends on the changing network state. In practice, not unlike other bandwidth allocation criteria, algorithms to achieve SABA in a dynamic network would only do so approximately. Recent studies [16, 31, 36, 41, 54] suggest how one might approximately achieve various 'fixed' weighted fairness criteria with the utility functions we considered earlier in defining SABA. A common theme considered in this body of work is the use of a decentralized user (sender) rate control mechanism with network feedback, such as TCP [29]. In this chapter, we will consider various practical issues associated with designing a network mechanism that approximates SABA, and present a simple preliminary implementation by incorporating a residual flow size factor into the TCP Reno congestion control mechanism.

The transfer rate of a file transfer, although elastic, may in practice be constrained by end system's interface capacity, *e.g.,* a modem or Ethernet card, or protocol limitations. In the protocol limitation case, for example, transfers mediated by TCP with receiver window size of 64 Kbytes and round trip delay of 20 ms can

only achieve throughput up to approximately $(8\times64\times10^3)/(20\times10^{-3}) = 25.6$ Mbps. Depending on the link (router) capacity along the path a file transfer traverses, the limitation incurred by the protocols or the end systems may or may not result in a 'peak rate constraint' for the transfer. As pointed out in [23], if the peak rate constraints associated with elastic flows are small relative to the bottleneck capacity and the traffic load is low, the average perceived throughput may be close to the peak rate. With the Internet today being designed and provisioned by independent entities and the existence of possibly complex service level agreements among them, it is not an easy task to predict where the bottlenecks for file transfers will be - see Chapter 1. An interesting question thus may be how the presence of peak rate constraints under various bottleneck scenarios changes the performance achieved by both SABA and traditional fairness criteria. We will begin by investigating the average BTD performance for elastic flows with peak rate constraints under various bottleneck scenarios when one uses SABA or max-min fair bandwidth allocation. We take max-min fairness to be our representative traditional fairness criterion. Our investigation based on fluid approximation will show that by using SABA elastic flows can achieve BTD close to the best possible, *i.e.,* the inverse of their peak rate constraints, even in a heavy load regime - this is not the case when bandwidths are allocated according to max-min fairness.

Given the strong evidence suggesting the potential benefits brought by SABA, even with the presence of peak rate constraints, we present a simple implementation of SABA, called SAReno, based on TCP Reno. Recent studies suggested that an additive increase and multiplicative decrease rate control algorithm, such as TCP,

achieves approximately weighted proportional fairness [31, 41], weighted potential delay fairness [23], and F_A^h fairness [54], to name a few depending on the modeling assumptions. The weight associated with a connection in these studies however is usually found to be fixed and is typically considered to be inversely proportional to the round-trip time of the transfer. In practice when flows come and go, one may argue that the user perceived average BTD, or throughput, under TCP is close to that achieved by any of the afore-mentioned fair bandwidth allocation - recall that these fairness policies achieve similar performance in the dynamic regime. One exception, however, is the throughput seen by users transferring small files. Indeed due to the limitation imposed by the slow start phase of current TCP mechanism, the transfer rate of small flows is unlikely to reach a good share of bandwidth before completion, see [6, 12, 23, 40, 55, 56] for detailed discussions. In particular [6, 56] have proposed alternatives to the slow start phase aiming at speeding up transfers with small 'initial' sizes. In this study, we will focus on implementing 'residual' size dependent differentiation in the congestion avoidance phase of TCP, which we believe will benefit 'all' transfers.

This chapter is organized as follows. In §4.2 we investigate the performance impacts of introducing peak rate constrained flows on various bottleneck scenarios assuming a fluid flow model. The proposed implementation SAReno will be presented in §4.3 accompanied by several design guidelines. Packet level simulations of the proposed transport mechanism are conducted using ns-2 [1] to show the gains achieved by SAReno over Reno, including discussions of several network design issues and the benefits of 'gradually' deploying such a mechanism. Concluding remarks

are given in §4.4.

4.2 Performance Impact of Peak Rate Constraints

4.2.1 Single Link Model

We begin by considering a single link with capacity c, which is shared by elastic file transfers each with a peak rate limit of $\bar{x}$. For analysis purposes we again assume the fluid flow model described in Chapter 2. The core question to be discussed is how $\bar{x}$, as compared to c, impacts the average BTD achieved by SABA versus traditional fair bandwidth allocation mechanisms.

Recall that when there is no peak rate constraint, a single link that uses fair allocation or SABA can be modeled by an M/G/1-PS or M/G/1-SRPT (in the limiting regime) queue, respectively. However, if the maximum rate each flow can accommodate is limited, the system behaves quite differently. In particular the link capacity may not be fully utilized when the number of flows is less than $c/\bar{x}$. We shall assume $\bar{x}$ divides c for simplicity. Based on the results in [13], one can derive the stationary distribution for the number of peak rate constrained flows on a single link under the fair share discipline. Unfortunately we have not been able to find an analogous result for the SRPT case. We note, however, that serving peak rate constraint flows with SRPT policy on a single link may be modeled by a 'SRPT routing/scheduling on a parallel server network'. That is, flows arrive to a network with $m = c/\bar{x}$ servers each with capacity $\bar{x}$. At any given time the m flows (if there are more than m) with the smallest residual work will be served by the m servers, while other flows wait at a central buffer until a flow finishes or another

smaller one arrives. A flow might be preempted and served on different servers. This generalized SRPT scheme corresponds to a routing/scheduling policy on a identical parallel server network [34]. Notice that, in general, a single link with peak rate constrained flows may not be modeled by the parallel link model described above, since policies such as fair sharing may allocate $2\bar{x}$ capacity equally to 3 flows which is not permissible by the parallel link model. Nevertheless, as we will show via simulation below, SABA on a single link with peak rate constrained flows achieves performance close to the best one can get - the peak rate under various scenarios.

A good indicator that distinguishes the character of a peak rate constrained system is the probability that the link capacity is fully utilized, *i.e.,* at least m ongoing flows in the system. Drawing from results for the peak rate constrained fair sharing link, we can derive this probability as follows:

$$f_{\text{full}}(m,\rho) = \frac{(m\rho)^m/m!}{(1-\rho)\sum_{k=0}^{m-1}(m\rho)^k/k! + (m\rho)^m/m!}$$

where ρ is the traffic load. The value of $f_{\text{full}}(m,\rho)$ corresponds to how often the flows are 'multiplexed' through the link. In other words, when no more than m flows are active, the system behaves like a circuit switched network, which presumably reduces the value of 'any' bandwidth allocation policy that determines the shared bandwidth to each elastic flow. As we will show in the sequel, the performance gain achieved by SABA in general grows with $f_{\text{full}}(m,\rho)$.

To examine the performance achieved by SABA and fair sharing with peak rate constrained flows, we conducted simulations on a 10 Mbps[1] link with vari-

[1]The capacity and peak rate constraints considered in this section are for presentation purposes,

ous m and ρ values. We again assume Poisson arrivals and bounded Pareto flow size distribution with mean 5 Kbytes. Fig.4.1 and Fig.4.2 show the average BTD improvement achieved by SABA over fair sharing along with the 'optimal improvement' and $f_{\text{full}}(m, \rho)$ when one varies m or ρ while fixing the other. The optimal improvement corresponds to comparing the peak rate under consideration with the performance achieved by fair sharing.

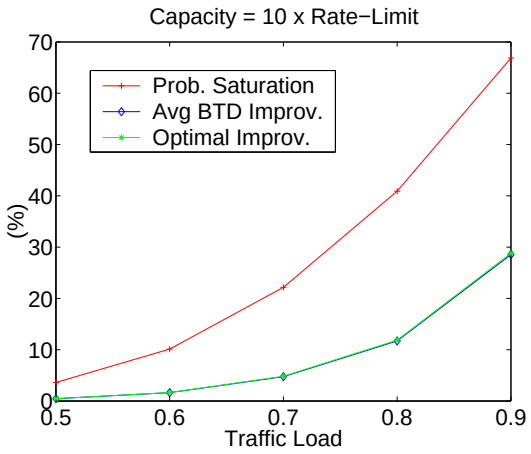


Figure 4.1: Average BTD Improvements achieved by SABA over FS, the optimal improvement over FS, and the probability of saturation when one varies the traffic load.

First observe that the average BTD improvement of SABA in general grows with $f_{\text{full}}(m, \rho)$, which increases as the traffic load increases and/or as the rate limit is getting closer to the link capacity. This is expected since the size-based differentiation can only take effect when the number of flows exceeds m - hence achieving multiplexing gain. Furthermore, the fact that SABA performs close to optimal un-

and may be different from actual systems. We however believe the results provide good qualitative insights.

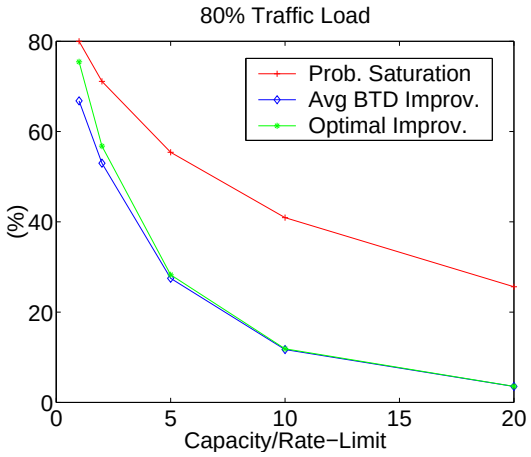


Figure 4.2: Average BTD Improvements achieved by SABA over FS, the optimal improvement over FS, and the probability of saturation when one varies the rate limit ratio.

der all scenarios suggests that the SABA scheme fully exploits the multiplexing gain and achieves almost the best possible BTD for all flows. In the case of a light load or when the peak rate constraint is much smaller than the link capacity, *i.e.*, $f_{\text{full}}(m, \rho)$ is small, almost all flows see a throughput close to the peak rate even under fair sharing, and hence SABA can only manage to improve performance marginally - which is again the best one can do.

The above results suggest the overall impact of introducing a 'homogeneous' peak rate constraint for all flows sharing a single link. In practice, file transfers going through a bottleneck link might be limited by different peak rates. For example, a university access gateway may serve both dial-up and Ethernet users which see diverse peak rate limits due to the speed of their end systems. Hence we conducted further experiments for such heterogeneous scenarios. We considered the same simulation setup as above but with flows of different peak rate constraints,

10 Mbps and 500 Kbps. Fig.4.3 exhibits the average BTD achieved by SABA over fair sharing for each flow type as well as the overall average BTD when the traffic load is 80%. As seen, both bandwidth allocation policies achieve almost the best possible performance for flows with the lower peak rate constraint, while SABA significantly outperforms fair sharing for those that can send at a higher peak rate. Overall, SABA achieves increasingly better performance as the percentage of high peak rate flows increases. The above trend remains true when we experiment with other combinations of peak rate constraints.

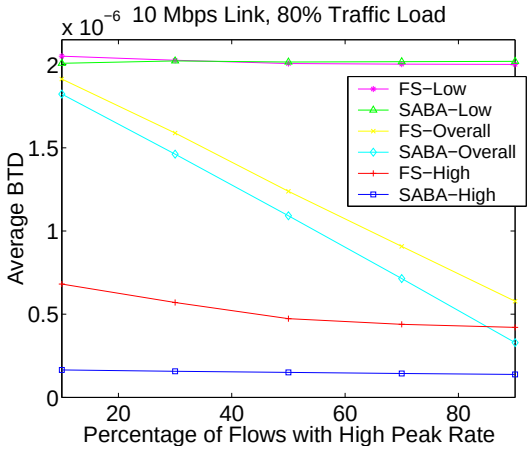


Figure 4.3: Average BTD achieved by SABA and fair sharing for flows with different peak rate constraints (10 Mbps and 500 Kbps) as the percentage of high peak rate constrained flows increases.

4.2.2 Networks as Traffic Concentrators

While the above analysis provides valuable insights on the impacts of introducing the peak rate constrained flows on a single bottleneck link, in practice a network may have multiple and possibly changing bottlenecks. In particular, a

typical scenario is that user flows are 'concentrated' into increasingly large capacity links until reaching the backbone, and then go to the destination access network. We shall consider the simple two-hierarchy network as shown in Fig.4.4, and assume all elastic flows entering the network have the same peak rate constraint 250 Kbps. In the considered network, each peak rate constrained flow enters one of the k Level

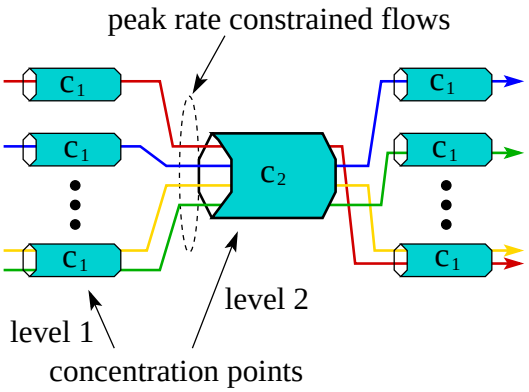


Figure 4.4: A two-hierarchy network that concentrate peak rate constrained elastic flows.

1 concentration points (with capacity c_1), then shares the Level 2 concentration point (with capacity c_2) with all other flows, and finally reaches destination through one of the destination access links. To achieve fair comparison shall we assume that the aggregate Level 1 capacity, *i.e.,* kc_1, remains constant at 10 Mbps - the same as the single link capacity considered in the previous experiments. We also assume the same arrival characteristics as before.

Based on our analysis for the single link case, the performance achieved by SABA or a fair bandwidth allocation, say max-min, will depend on c_1 and c_2. We first consider scenarios where we fix $c_2 = kc_1$=10 Mbps but vary k and thus c_1 to

observe the impact of the design of concentration points close to the users. Note that with $c_2 = kc_1$, the Level 2 pipe will not be the bottleneck to constrain the flow bandwidth. Fig.4.5 exhibits the average BTD achieved by SABA and max-min bandwidth allocation policy when the number of Level 1 concentration points increases with total offered load being 8 Mbps (80% of system capacity). As seen, while using more but smaller Level 1 links aggravates the average BTD drastically under max-min, one may achieve a relatively constant performance close to the optimal, *i.e.,* BTD $= 4 \times 10^{-6}$ secs/bit $=$ (peak rate)$^{-1}$ secs/bit, by using SABA. This suggests SABA's superior robustness to the system environment, and benefit when one has limited options to aggregate traffic. In particular, as k grows and c_1 decreases, resulting in a less effective bandwidth sharing under max-min fairness, the size-based differentiation achieved through SABA offers increasing performance improvements, *e.g.,* 28% to 62% when k increases from 5 to 20. We further note that SABA achieves higher performance improvements here than it would have achieved on a single link with capacity c_1 and the same peak rate constrained flows - recall Fig.4.2. This suggests that on a network where bottlenecks change over time, one will see a higher effect of multiplexing and thus a more significant benefit brought by SABA.

We next examine the impact of the capacity of the Level 2 link. It is clear that the smaller c_2 is the more likely it is for the Level 2 concentration point to become the bottleneck. For this set of experiments, we assume $k = 10$, $c_1 = 1$ Mbps, a 6 Mbps total offered load, and vary c_2 from 6 to 10 Mbps. We again assume all flows are constrained at 250 Kbps. Fig.4.6 shows the average BTD of peak rate

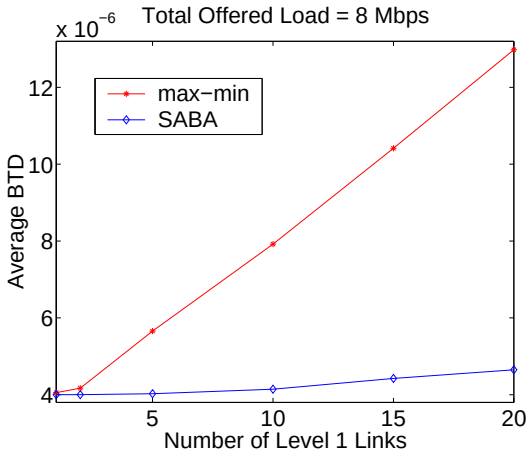


Figure 4.5: Average BTD achieved by SABA and max-min bandwidth allocation on the network shown in Fig.4.5 when one increases the number of Level 1 links with 8 Mbps offered load.

constrained flows achieved by SABA and max-min as one increases c_2. Observe that the average BTDs achieved under either max-min or SABA remain similar as long as the Level 2 capacity exceeds the offered load ($c_2 \geq 7$ Mbps), with SABA outperforming max-min by about 24% in this case. Again this 24% improvement is larger than SABA would have achieved on a single link with capacity c_2 and the same peak rate constrained flows - recall Fig.4.1 and Fig.4.2. Furthermore, as c_2 goes to 6 Mbps, *i.e.,* the total offered load goes to 100% of the bottleneck capacity, the average BTD under max-min quickly explodes while SABA still maintains a good performance close to the peak rate.

The above experiments suggest that, unlike fair bandwidth allocation, SABA is 'robust' - performs well on various plausible access network scenarios with different peak rate constraints. In particular, under the fluid flow model almost all flows see

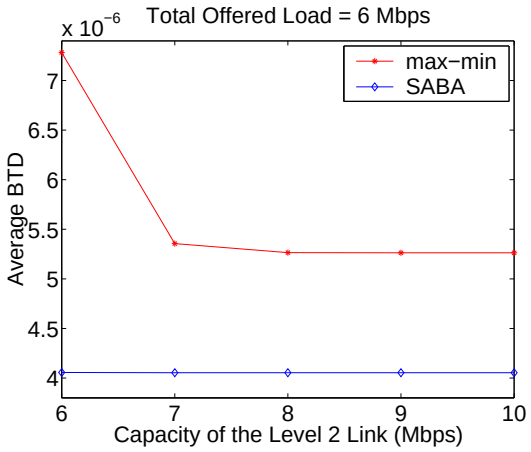


Figure 4.6: Average BTD achieved by SABA and max-min bandwidth allocation on the network shown in Fig.4.5 when one increases the Level 2 link capacity with 6 Mbps offered load.

transfer rate close to their peak rates under SABA[2] for all scenarios we simulated. The benefit brought by SABA is particularly significant when the offered load is close to, possibly temporarily, the bottleneck capacity as would be expected in concentration networks that groom data traffic on scarce broadband links in the access/metro area networks.

4.3 SAReno: Reno with Size-based Differentiation

4.3.1 Algorithm Description

In this section we present a simple implementation of SABA based on TCP Reno in the congestion avoidance phase, where the user additively increases his window size (transmission rate) when receiving acknowledgments and multiplicatively

[2] As we will discuss in the next section, in practice file transfers with small file sizes are unlike to achieve good transfer throughput due to current transport protocol limitation.

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

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

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 will only consider the case where each user only has access to his own residual flow size information, *i.e.,* SABA with $w_{\text{in}} = w_{\text{ex}}$. A key assumption for SAReno is that the transport protocol can obtain the initial size information from the application layer. This is a reasonable assumption since for most file transfers today, the sender knows a-priori the size of the locally stored files, *e.g.,* static contents associated with web servers. SAReno 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 4.1. For example, a SAReno flow with 20 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 the 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

Table 4.1: Parameters for SAReno: δ (1/cwnd), κ (cwnd)

	range of $p(t)$ (packets)				
	$[0, 10)$	$[10, 50)$	$[50, 200)$	$[200, 1000)$	$[1000, \infty)$
$\delta^{\text{SAReno}}(p(t))$	10	5	1	0.5	0.25
$\kappa^{\text{SAReno}}(p(t))$	1.0	0.9	0.5	0.1	0.01

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 information.

4.3.2 Performance Evaluation: 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 [1] simulator. We focus on the star network considered in Section 3.5 with 1 ms propagation delay on each link connecting an access domain to the center backbone node. We assume the same arrival processes as those used in Section 3.5. The flow size distribution is assumed to be bounded Pareto with mean 50 KBytes, *i.e.,* 100 packets with 500 bytes per

packet. Notice that we have selected a larger mean flow size than is currently typical of TCP transfers on the Internet. This was done in order to clearly exhibit the performance impacts of SAReno on, say, bulk transfers, ignoring extremely small flows which may not even enter the congestion avoidance phase, *i.e.,* complete before reaching the slow-start threshold. In the sequel, in addition to presenting SAReno's performance improvements over Reno, we will also discuss additional network issues that further impact performance.

Fig.4.7 shows the average BTD achieved by SAReno and Reno with various link packet scheduling mechanisms as the traffic load increases. We consider two different underlying packet scheduling disciplines: First Come Fist Serve (FCFS) and Deficit Round Robin (DRR) [52]. While FCFS is considered the 'default' discipline, we also present the results for DRR to show the performance benefits over FCFS particularly when SAReno is employed. As seen SAReno outperforms Reno for both the FCFS and DRR cases - by about 30-40% for a range of traffic loads.

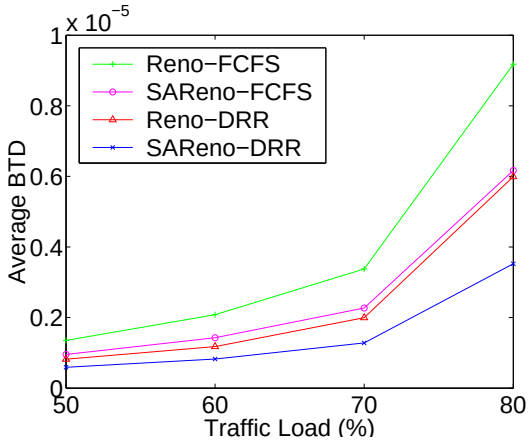


Figure 4.7: The average BTD achieved by various Reno/SAReno and FCFS/DRR combinations when traffic load increases.

The 30-40% improvements, though significant, are less than those exhibited in the fluid-flow simulations presented in Section 3.5. This is partly due to the fact that we have not introduced size-based differentiation in the slow-start phase[3] during which small flows (in the number of total packets to send) will spend most of their lifetimes. Moreover, as new flows initially start with a window size at one packet, it is unlikely for small flows to reach a transmission rate (window size) that captures the preferential treatment they are being given. This lack of differentiation for small flows compromises the benefits resulting from size-based differentiation introduced in the congestion avoidance phase. Then again performance improvement for this type of flows may not be as important.

Observe further from Fig.4.7 that the average BTD performance achieved with FCFS is much worse than that with DRR, for both Reno and SAReno. This is because under FCFS the first few packets of new arrivals often see large initial queuing delays resulting from the build-up of queues by existing flows. This 'initial delay' hurts performance for almost all flows, particularly for the small ones, which presumably have relative short lifetimes. To reduce these performance draw-backs, one may consider alternative packet scheduling mechanisms, *e.g.,* DRR or WFQ. DRR, as a representative of such 'fair' packet scheduler, allows new flows, even with only one packet in the queue, to quickly catch up with existing ones in window size and thus achieves better performance. By attempting to achieve 'packet level' scheduling fairness, these policies enhance the size-dependent differentiation

[3]One may also consider speeding up small flows by altering the slow-start algorithm, see *e.g.,* [6, 56] for two proposals that probe and estimate available network resources.

implemented by the proposed transport level mechanisms.

The last observation on Fig.4.7 is that one would see a relative 'flat' performance improvement, about 30%, achieved by SAReno over Reno as traffic load increases for the FCFS case, while it increases up to 40% with DRR discipline. We provide an intuitive explanation regarding this matter. Note that as traffic load increases, the impact of the initial delay under FCFS mentioned above becomes more significant and thus further compromises the benefits of size-based differentiation. This cancels out the *increases* in performance improvement that we would have expected as the load increases. By contrast, DRR eliminates impact of these initial delays.

A fair share packet scheduling mechanism, such as 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 in the backbone. Under these circumstances, one may consider a simpler scheduling mechanism, such as FCFS or a hierarchical implementation of DRR (HDRR) packet scheduler, which only distinguishes aggregate flows from/to different domains (ingress/egress nodes). To compare the performance achieved by SAReno with various scheduling policies implemented at the core router, we considered a simple 'bridge' network with small access links and a larger core pipe in the middle, as shown in Fig.4.8. The four access links all use DRR while the center one may choose to use either FCFS, HDRR, or DRR.

Fig.4.9 shows the average BTD performance under SAReno on the bridge network when the center node uses either DRR, HDRR, or FCFS as the traffic load

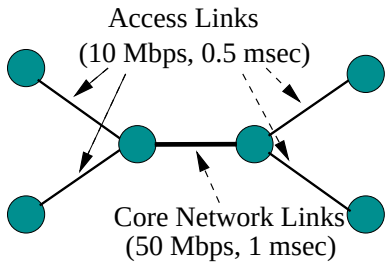


Figure 4.8: A bridge network with large bandwidth on center (core) link.

increases. One can see that regardless of which scheduling policy is used in the core, the performance achieved is similar. Thus as expected only the choice of packet

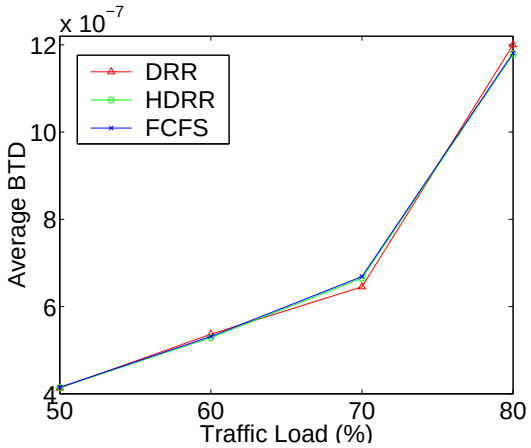


Figure 4.9: The average BTD performance achieved under SAReno with different packet scheduling on the center (core) link as traffic load increases on the bridge network.

scheduler at the bottlenecks, typically at the access networks or peering points , is important from the point of view of enhancing user perceived performance. Thus for the case where core routers have much higher capacities than access networks or peering points, one may choose to deploy simple packet scheduling mechanism, *e.g.,* FCFS, at the core, and per-flow packet scheduler only at the edges.

As in the case of the fluid-flow simulations, we have also conducted simulations on random mesh networks. Fig.4.10 shows the average BTD improvements, with 90% confidence intervals, achieved by SAReno over Reno (with DRR) for these topologies as the traffic load increases. One can clearly see a 26-38% performance improvement that is similar to those in the star network. We explicitly plot these set of results with 90% confidence intervals[4] to exhibit the consistent performance gains one can achieve by employing SAReno instead of Reno for data transfers.

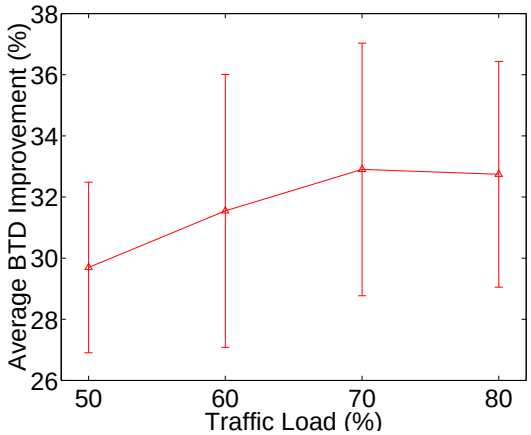


Figure 4.10: The average BTD performance achieved under SAReno over Reno with DRR for various complex topologies.

4.3.3 Penetration Experiments

An interesting question is how SAReno and Reno flows might fare if they coexist on a network. We conducted preliminary simulations to examine how the performance benefits would accrue as the penetration of SAReno flows increases.

[4]The deviation of results for other plots are relatively smaller than the one shown in Fig.4.10 and thus we omit plotting those confidence intervals for a clear presentation.

The simulations presented below concern the 6-branch star network.

We first consider a random penetration scenario, where the transfers to be mediated via SAReno rather than Reno were selected at random according to the penetration level being simulated. Fig.4.11 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 suggest that users will have proper incentives to upgrade from Reno to SAReno.

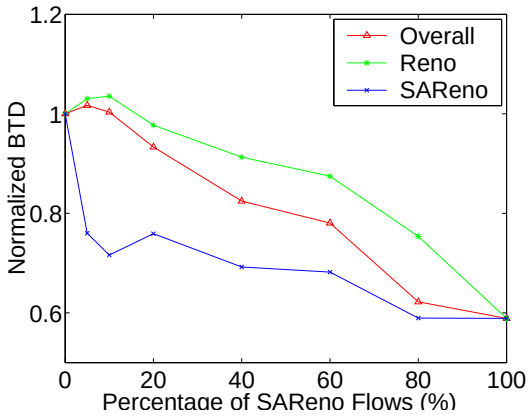


Figure 4.11: Normalized BTD (over the case where all flows are Reno) as the percentage of (random) SAReno flows increases.

Alternatively SAReno might not be deployed in the homogeneous manner discussed above, but in a more clustered manner corresponding to, say, access do-

mains 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.4.12 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 are normalized by that when all flows are mediated via Reno. 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 - see Fig.4.11. This is to be expected since the intra-route discrimination can be more effective when SAReno flows originate from the same access domain.

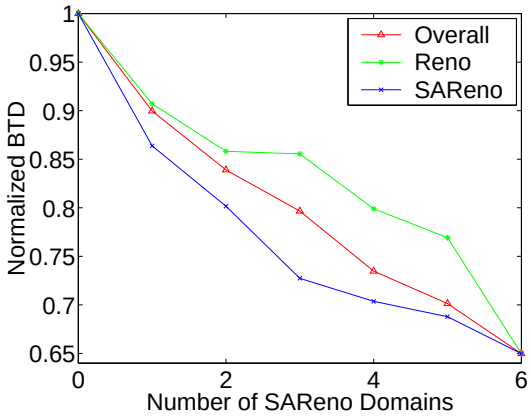


Figure 4.12: Normalized BTD (over the case where all flows are Reno) when one increases the number of domains that sends SAReno flows.

4.4 Concluding Remarks

In this chapter, we have further examined the benefits in performance brought about through SABA in more practical settings. We have investigated the performance impact of introducing peak rate constraints on elastic flows under various bottleneck scenarios. Our analysis suggests that by contrast to traditional fairness criteria, SABA provide a 'robust' performance close to the best one can achieve - the peak rate for all bottleneck scenarios. We have also presented a simple implementation, SAReno, based on TCP Reno, that makes use of the residual flow size to improve the average user perceived BTD performance. By using ns-2, we have shown that SAReno can achieve up to 40% performance improvement over Reno at 80% traffic load. The benefit increases with the penetration of SAReno service, and incentives (in terms of achieved performance) are likely to promote users to upgrade.

These are encouraging gains and provide significant evidence on the possible benefits of residual size based differentiation, suggesting in our opinion that this line of research and possibly development warrant further exploration. We note that size-based differentiation can be also realized on top of other network mechanisms. For example, [21] have proposed and developed an implementation of SRPT on web servers. Further examples may be found in [6, 40, 55, 56] which aim at alleviating the limitation imposed by TCP slow-start control to flows with small 'initial' transfer sizes. These proposals either require monitoring and recording network loads and estimating a good initial window size for each flow [6, 56], or classify TCP flows into two classes, short and long ones, and employ differentiation in packet scheduling/queue management mechanisms at core routers [40, 55]. While

the above schemes, including ours, may be complement to each other, we believe that implementing 'residual' file size differentiation at the 'transport' layer enables one to address network bottlenecks and/or interactions among various flows on the network resources, and thus benefits all flows with different sizes in a variety of network scenarios.

Chapter 5

Model of User Impatience

5.1 Introduction

File transfers over the Internet typically adapt their transmission rates to dynamically share congested links among contending transfers. Investigations on the performance seen by such transfers has mostly focused on an underloaded regime. However, as the congestion level goes up, depending on the way bandwidth is shared, some or all flows may see poor performance, possibly leading to aborted transfers, *i.e.,* stopped before completion, due to *user impatience.* Empirical evidence collected from representative servers [2, 19, 42] suggests that a non-negligible volume of data may correspond to aborted transfers, *e.g.,* [19] found that 11% of all transfers were interrupted, corresponding to 20% of the transfered volume. Users may abort their transfers, *e.g.,* push the stop button on the browser, for various reasons, such as incorrect document address, long connection setup time, or poor performance during transfer. Our focus herein is on user impatience with respect to *transfer dynamics* once a connection is established. Of particular interest will be the interaction between user impatience characteristics and bandwidth sharing policies.

As we have discussed in the previous chapters, introducing residual flow size differentiation into the transport mechanism can significantly enhance the average BTD performance for elastic file transfers. By contrast to the traditional fair band-

width sharing policies, the key to size based differentiation is to exploit the range of user tolerances to bandwidth or delay in order to benefit the whole. One can speed up small transfers, while large ones see a negligible performance degradation so that a better overall system performance can be achieved. Given the difficulties in traffic modeling, planning, and dimensioning of links for data transfer networks, transient overloads may be unavoidable. Re-examining the design objectives underlying bandwidth sharing on today's networks leads to further questions. What happens to user perceived performance when the system is overloaded? Is a graceful degradation achieved? How do various bandwidth sharing mechanisms fare when users are impatient?

While 'user impatience' leads to aborted transfers, the notion of 'unacceptable performance' may vary among users. For example, one user may expect his transfer to constantly progress in excess of some rate, while another may only wish to finish his transfer within a certain time period regardless of how it progresses. Modeling user impatience can be fairly complicated since it may depend on the size of the transfer, how long the transfer has lasted, as well as other subjective/semantic characteristics. The work in [23] independently conducted a preliminary investigation of the performance impact of user impatience on a single link where the link capacity is equally shared by on-going transfers, *i.e.,* fair sharing. Their analysis however only considers one type of impatience behavior where all users have a fixed delay constraint and assumes fixed bandwidth allocation for all flows throughout their transfer (before completion or being aborted). Despite their simplified model, the results shown in [23] suggest that when the system is highly overloaded (190%)

fair sharing can only achieve a goodput slightly above 50%. Our goal here is to provide a general model of user impatience and investigate and compare the impacts on both system and user perceived performance when one employs the residual size based differentiation or fair share bandwidth allocation.

This chapter is organized as follows. In §5.2 we evaluate two bandwidth sharing policies, *i.e.,* fair sharing and size-based differentiation, when users are 'not' impatient. We intend to exhibit and compare performance degradation incurred by the two policies when the system is moving from an underloaded to an overloaded regime. In §5.3 we present two generic models capturing a wide diversity of plausible user impatience behaviors. Note that in practice user behavior can be very complex. Our goal is to achieve a better understanding of how the characteristics of user impatience impact the performance achieved by the two types of bandwidth sharing schemes. After identifying several performance metrics for evaluating systems with aborted transfers, we will provide a detailed discussion accompanied by simulation results in §5.6. An extension of our generic approach to the case where users' perception of performance is associated with transferring a 'cluster' of files, as might correspond to a single web page access, is discussed in §5.7. Concluding remarks are given in §5.8.

5.2 Bandwidth Sharing: Underloaded versus Overloaded Regimes

We again consider the fluid flow model defined in Chapter 2. To capture the performance during transfer, we assume all requests are granted, *i.e.,* no incorrect address or denied access, and the response time from initiation to the time the

server starts transferring is zero[1]. Once a transfer is initiated, it contends with on-going flows on a fixed set of network resources throughout its lifetime, *i.e.,* fixed routing until completion or the user aborts his transfer. For analysis purposes, we will consider various traditional fairness criteria and SABA on a single bottleneck link in this chapter, *i.e.,* all data transfers contend for a single resource's capacity. Note that this may be a reasonable assumption considering that the bottleneck of data transfers typically is at the edge of the network, *e.g.,* access routers or concentration links, and assuming that one can neglect other factors, such as the heterogeneous round-trip delay, impacting the bandwidth allocation. For the single link case, various fairness policies considered in the literature essentially correspond to providing an equal share of the link capacity to all ongoing flows. We shall refer to these as Fair Sharing policies (FS). For presentation purposes, we shall also refer SABA on a single link as Size-based Differentiation (SD) in this chapter. Before examining the interaction of user impatience with the two bandwidth allocation policies, we consider how FS and SD perform when moving from an underloaded to heavy or even overloaded regime assuming no aborted transfers.

5.2.1 Increasing the Dynamic Range of Operation with Acceptable QoS

We begin our discussion by examining two queues, M/G/1-PS and M/G/1-SRPT, which model FS and SD, respectively. We exhibit the difference in the traffic load the queues can support while achieving the upper limit on the average BTD seen by users. We plot two lines in Fig.5.1 that connect points indicating

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

the maximum load one can support under PS (y-axis) and SRPT (x-axis) given the required average BTD values. The two lines are associated with the underlying 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 SRPT versus PS. For example, for the case of bounded Pareto distribution, PS can support up to a 60% load while SRPT can support up to 90% traffic load, while meeting the same average BTD. This 30% difference in supportable traffic volume represents a significant revenue increase, or additional flexibility in provisioning such networks.

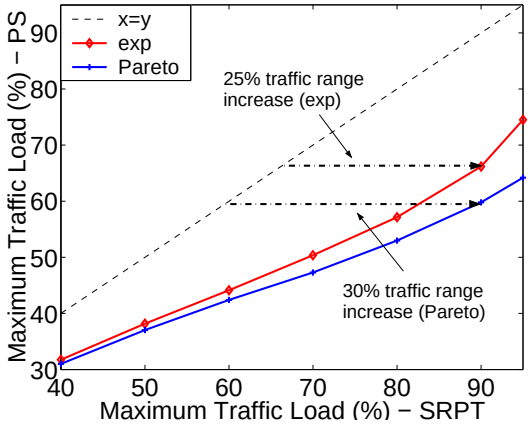


Figure 5.1: Maximum traffic load one can support under M/G/1-PS (y-axis) and M/G/1-SRPT (x-axis) for same average BTD requirements.

One may also view the numerical results shown in Fig.5.1 from an alternative perspective where the performance improvement achieved through SD is translated to an equivalent capacity increase required for a FS-based system. Fig.5.2 shows the percent capacity augmentation required for the above two queues to achieve the same performance as the traffic load on the system increases. The x-axis cor-

responds to the load in percentage for the system without capacity augmentation. The results are shown again for the cases of bounded Pareto and exponential flow size distributions. We also plot a lower bound of the percent capacity augmentation for any flow size distribution based on [4]. The lower bound is of interest as it is valid for all flow size distributions, however as can be seen for distributions of practical interest, *e.g.*, bounded Pareto, it is not tight. For the traffic loads considered in this regime, one can see that a 40-50% additional capacity might be required, using a FS type discipline, to achieve the same average BTD under SD.

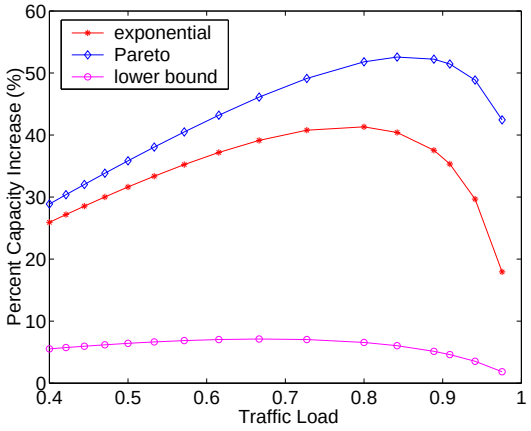


Figure 5.2: Percent capacity increase needed for a M/G/1-PS queue to achieve the same average BTD as a M/G/1-SRPT queue does as the traffic load increases.

5.2.2 Uniform Performance Degradation under Heavy/Overload Regime

We may now present simulation results for flows sharing a single (FS or SD) link when the offered load increases from 70%, to 90%, and then to 110%. We assume that the link capacity is 10 Mbps, the transfer flows arrive according to Poisson processes, and the file size distribution of the transfers is bounded Pareto

with mean 5 Kbytes. Note that the results shown for the overloaded cases are 'transient' in the sense that they are collected on a finite event simulation for an unstable system.

Fig.5.3 exhibits the cumulative proportion of flows that perceive a normalized average BTD less than the value shown on the x-axis. We normalize the average BTD such that the value of 1 indicates the smallest possible BTD one can achieve, *i.e.,* when a flow has the link capacity to itself. As can be seen, under SRPT (bot-

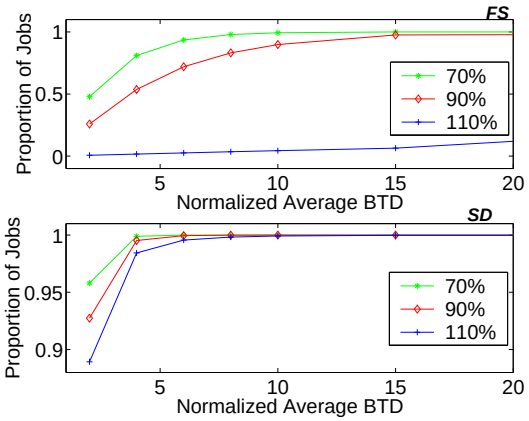


Figure 5.3: Cumulative proportion of flows that perceive different BTDs under PS (top) and SRPT (bottom) with 70%, 90%, and 110% traffic load.

tom) more than 98% of flows can achieve a normalized BTD of 4 or better even when the traffic load is 110 % – see graph at bottom. By contrast the proportion of flows achieving good BTD drops dramatically for the PS case – see graph at top. In particular, the number of flows under PS that experience a BTD that is 20 times worse than the best drops to more than 80%. This suggests that when the system is overloaded SRPT can still achieve a reasonable overall performance while it will quickly become unacceptable using PS.

To see the role of flow-size differentiation we further compared the average BTD experienced by sets of flows having different sizes. Fig.5.4 shows the average BTD (on a logarithmic scale) perceived by sets of flows with increasing size. We divided flow sizes into 6 bins, where the first bin are the flows that have size in the interval $[10^3, 10^4)$ bits, the second in $[10^4, 10^5)$, and so on. As seen, when moving from the underloaded to the overloaded regime, SD maintains good performance for small to medium size flows (graph at bottom), while FS degrades performance 'uniformly' for all flows (graph at top). Notice that since most transfers are small in size, as suggested in by, *e.g.,* [2, 14], SD will benefit the majority of demands (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.

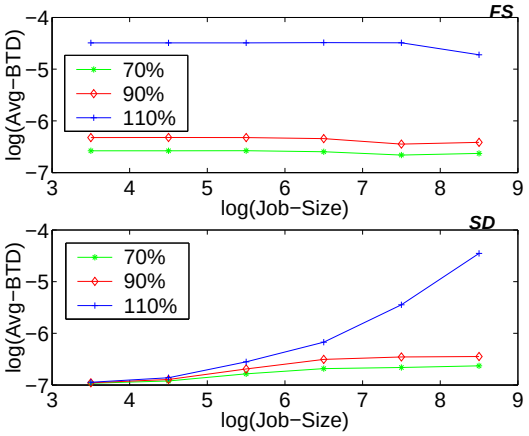


Figure 5.4: Average BTD for different size flows under FS (top) and SD (bottom) in the underload (70%), heavy-load (90%), and overload (110%) regime.

The above results suggest that, in the regime where users do not abort their transfers, SD not only increases the acceptable range of traffic loads over FS for a

given average BTD requirement, but also provides a *graceful* performance degradation when the system experiences transient overloads. In practice users may interrupt their transfers before completion due to, say, poor transfer performance, the 'effective' traffic load might also be reduced 'gracefully' under SD and thus produce an even better performance for the remaining users. In the next few sections, we will investigate how user impatience, resulting in aborted transfers, impacts performance achieved by the FS and SD policies.

5.3 Modeling User Impatience

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

Definition 5.3.1. *We call $e_c(t)$ a* minimum cumulative service *(MCS) curve for a user if it represents the minimum amount of work that needs to be completed after t units of time since a transfer was initiated. That is, such a user will abort his transfer at time $t > 0$ if and only if $w(0,t] < e_c(t)$.*

Definition 5.3.2. *We call $e_p(u)$ a* minimum progress service *(MPS) curve for a user if it represents the minimum amount of work that needs to be done during*

any time interval of length u during the transfer. That is, such user will abort his transfer at time $t > 0$ if and only if $w(s,t] < e_p(t-s)$, for some $0 < s < t$.

The key difference between the two models is that the MPS curve captures a user's 'time-invariant' expectation of perceived performance, and thus can be used to evaluate the transfer progress at shorter time scales from the current time to the past, while only the largest time window $(0,t]$ is used by the MCS users. For convenience we refer $e_c(t)/t$ as the minimum expected 'cumulative throughput' at time t, and $e_p(u)/u$ as the minimum expected 'transfer rate' at time scale u. Fig.5.5 shows an example of how a user might evaluate his transfer performance based on MCS and MPS curves. Let p denote the total size of the transfer. We consider two service curves which have the exact same shape but different meanings. We let $e_c(t) = rt$ and $e_p(u) = ru$. This means that the user is expected to have a constant minimum cumulative throughput for the MCS case and a constant minimum transfer rate (the slope of $w(0,t]$) for the MPS case. Observe that the transfer completes for the MCS case since the cumulative work $w(0,t]$ always stays above $e_c(t) = rt$. By contrast, the MPS curve imposes a more stringent constraint and thus the transfer will be aborted when the user perceives a slower transfer rate than r.

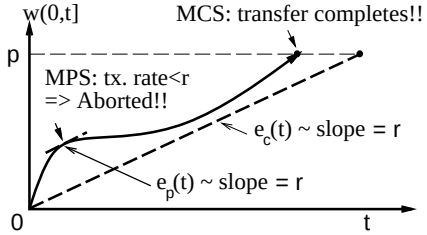


Figure 5.5: Ex: evaluate transfer performance based on MCS and MPS curves.

In general $e_c(t)$ and $e_p(u)$ can be any non-decreasing function with respect to the 'elapsed time' t and the 'evaluation time window' u, respectively. Fig.5.6 shows several characteristic MCS ((a), (b), (c), and (f)) and MPS ((d) and (e)) curves. The service curves shown in Fig.5.6 (a) and (d) are the ones we considered in the previous example. Note that for these two cases, users evaluate performance from the very beginning of the transfer. Furthermore, for the case shown in (d), it is assumed that the user can monitor the 'instantaneous' transmission rate. Typically, however, users may be patient at the very beginning of a transfer, *i.e.,* there is some 'grace period' before users might start evaluating performance. Drawing on an analogy from the leaky bucket constraint [11], we can introduce such grace period by letting

$$\text{MCS: } e_c(t) \;=\; rt - \sigma, \qquad \text{MPS: } e_c(u) \;=\; ru - \sigma, \tag{5.1}$$

- see Fig.5.6 (b) and (e). Note that for the MPS curves, the introduction of σ not only provides an 'initial' grace period but also a 'time scale' over which users evaluate the transfer rate. This is a more reasonable assumption than that used for the user behavior exhibited in (d).

The leaky bucket type of function can be further generalized to the case where a user wishes to evaluate his cumulative throughput or transfer rate at multiple time scales by defining

$$e_c(t) \;=\; \max(0,\; r_1 t - \sigma_1,\; r_2 t - \sigma_2, \cdots), \tag{5.2}$$

$$e_p(u) \;=\; \max(0,\; r_1 u - \sigma_1,\; r_2 u - \sigma_2, \cdots). \tag{5.3}$$

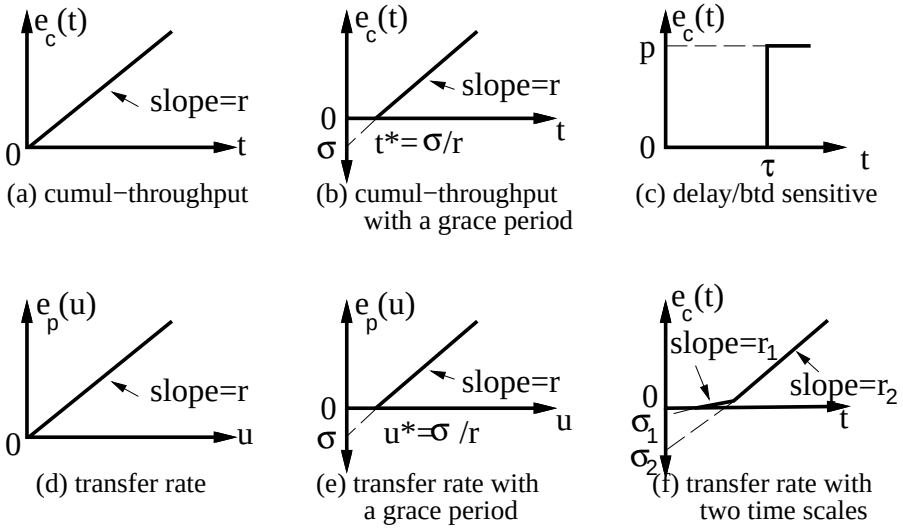


Figure 5.6: Examples of MCS ((a), (b), (c), and (f)) and MPS ((d) and (e)) curves.

When $0 \leq r_i \leq r_j$ and $0 \leq \sigma_i \leq \sigma_j$ for all $0 < i < j \leq n$, the MCS (MPS) curve is convex, and reflects the scenario where users can tolerate a slower cumulative throughput (transfer rate) for a small time period, but expect to see a higher one on larger time scales. Note that a convex MCS curve might be arguably representative for typical users who are aware of their transfer sizes. More specifically, a user with a larger file to transfer may expect to wait longer, but not so long in proportional to the size of the transfer - expecting an increasing throughput at a larger time scale. Fig.5.6 (f) shows an example where the user evaluates his cumulative throughput on two time scales.

Another type of user may not be concerned with how exactly his transfer progresses. Instead, he may only wait for τ units of time for his transfer to complete, as shown in Fig.5.6 (c). Note that a user who exhibits such impatience behavior

is a limiting case of those shown in (a), (b), and (f), and is more 'elastic' in the sense that he has higher flexibility in allocating bandwidth during the transfer, *e.g.*, considering $\tau = p/r$ where p is the size of the transfer and r is the corresponding parameter for behavior (b). When the value of τ is independent of the size of the transfer, we call such users (fixed) delay sensitive. Alternatively, a user may be aware of that the transfer delay depends on the size of his transfer. To model such cases, one can set $\tau = p/r$ with a fixed r, which implies that the user expects a maximum BTD of $1/r$ regardless of the transfer size. We call such users BTD sensitive. Similarly, other parameters, such as σ, can also depend on the file size to reflect that the user may evaluate his cumulative throughput or transfer rate less frequently if the size is larger.

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

5.4 Performance Metrics with Aborted Transfers

Before we discuss the interaction between user impatience behavior and bandwidth allocation policies, we shall first identify the metrics one may use to evaluate system as well as user perceived performance when users exhibit impatience behavior. The fact that users may abort their transfers due to unsatisfactory performance may result in mixed consequences. On the one hand, the aborted

transfers may be translated to denied service, and the work that was done for those aborted transfers is wasted[2] and thus contributes to what one may call 'badput'. On the other hand, with some transfers leaving the system prior to completion, the effective traffic load is reduced and thus the rest of the transfers may see a reasonable performance even when the system is overloaded. Table 5.1 exhibits several metrics that we will use to evaluate both user perceived performance and system efficiency.

Table 5.1: Metrics when users abort transfers

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

5.5 Analytical Model for M/G/1-SRPT Queue with Homogeneous Delay Sensitive Users

We supppose all flows will see the delay in the steady state in an M/G/1-SRPT queue. Let c denote the link capapcity and $f(p)$ and $F(p)$ be the probability density function and cumulative distribution functions of the flow sizes, respectively. According to [4], the steady state delay for flow of size $p < p^*$ can be expressed as

$$\mathbb{E}[D(p)] = \left(\int_0^p \frac{ds}{1-\rho(s)} + \frac{\lambda}{2} \frac{\int_0^p s^2 f(s) ds + (F(p^*) - F(p))p^2}{(1-\rho(p))^2} \right) \frac{1}{c},$$

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

where $p^* = \arg\{p \mid \rho(p) = 1\}$ and $\rho(p) = \lambda \int_0^p s dF(s)/c$. Notice that the above equation is also valid for an undeloaded system, *i.e.,* $\lim_{p\to\infty} \rho(p) \leq 1$, in which case the term $F(p^*)$ will be replaced by 1 and all flows will see a finite delay in the steady state. Now since $\mathbb{E}[D(p)]$ is monotinically increasing in p, one can always find a $\bar{p}$ for a given $\bar{d}$ such that

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

Now one may approximate the goodput and throughput associated with a M/G/1-SRPT queue with homogeneous delay sensitive users by $\bar{u} = \rho(\bar{p})$ and $\bar{v} = \lambda \int_0^{\bar{p}} f(s)ds$, respectively.

5.6 Bandwidth Sharing vs. User Impatience

This section investigates, via simulation, how FS and SD perform when users exhibit different impatience behaviors, *i.e.,* the ones we discussed in §5.3. We examine various scenarios wherein all users have the same impatience behavior. We will once again consider the single link case described in §5.2.

We will begin by considering users who are sensitive to cumulative service. Fig.5.7 through 5.10 show the system performance achieved by SD and FS under four MCS type of behaviors. The parameters used for each case are shown on top of the figures. We plot the average completion and incomplete rate on the left and the goodput, badput, and residual work per second on the right for each behavior. The results for the FS and SD cases are shown in adjacent bars (FS: left, SD: right).

Observe first that for these behaviors, SD performs mostly better than FS

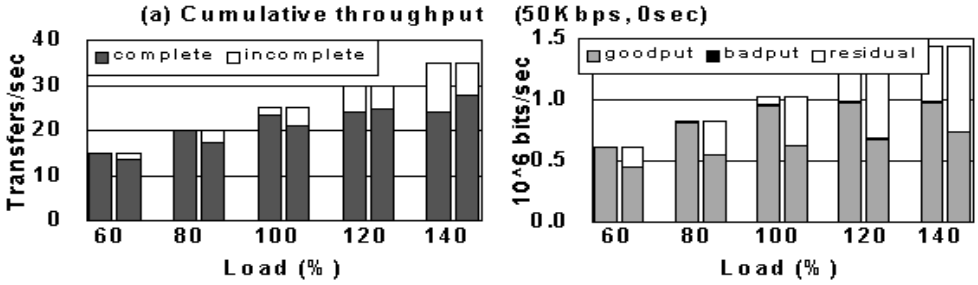


Figure 5.7: Performance under FS and SD with cumulative throughput (zero grace period) sensitive users.

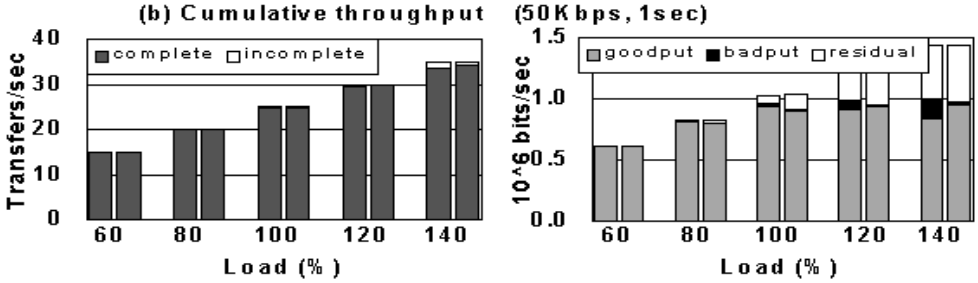


Figure 5.8: Performance under FS and SD with cumulative throughput (1 sec grace period) sensitive users.

except for the case shown in Fig.5.7, where users are sensitive to the cumulative throughput of 50 Kbps with a zero grace period, and in terms of goodput. The reason is that without the initial grace period, large transfers may be discontinued early on, *e.g.*, right after initiation, under SD when small ones are also present. These large files, although only few in number, contribute a large portion of the total work, hence a reduction in goodput. This is evidenced by the fact that with such user impatience behavior and in the overload regime, SD can complete more transfers per second but achieves less goodput. Note however that when the large

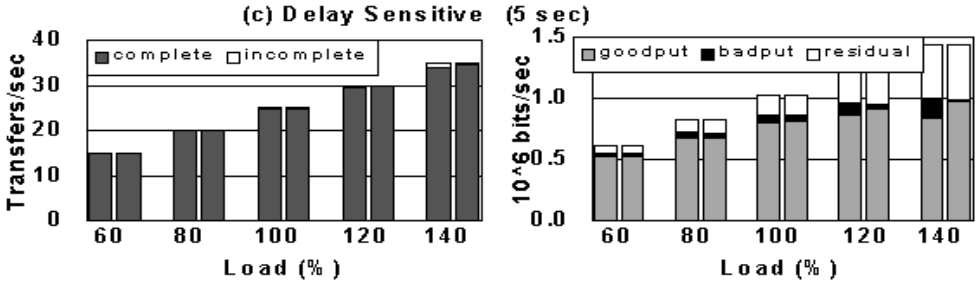


Figure 5.9: Performance under FS and SD with Delay sensitive users.

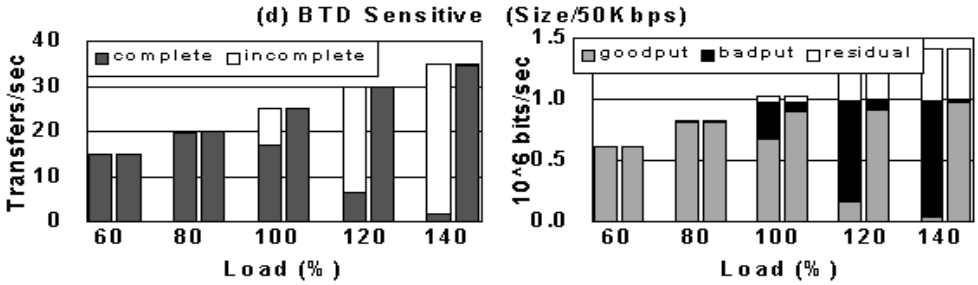


Figure 5.10: Performance under FS and SD with BTD sensitive users.

flows stay in the system, they may see a similar performance as they would have seen under FS - recall our results shown in Fig.5.4. In fact, the zero grace period is not only unreasonable for users to evaluate throughput but also limits the SD's flexibility in allocating bandwidth to various size transfers.

Now if we add a 1 second grace period before users start evaluating throughput, SD not only catches up but further outperforms FS upon system overloads - see Fig.5.8. Meanwhile, with the inclusion of a grace period, both FS and SD allow almost every job to complete, except some very large ones in the overloaded regime to reduce the actual traffic load (to be below 1 Mbps). The difference in

this case between FS and SD is that upon aborting large transfers, a larger portion of those files has been transferred under FS, resulting in a more significant badput. In fact this larger badput phenomenon under FS applies to all cases we explored below. This suggests that for most cases, SD achieves a more efficient utilization of resources.

Similar to having a grace period, users who are sensitive to delays also allow SD to be flexible in allocating bandwidth. As seen, results shown in in Fig.5.9 are similar to those in Fig.5.8, except less goodput is incurred for the case of delay sensitive users. This is due to the 5 second limit and 1 Mbps capacity which makes it impossible to complete transferring files of size larger than 5 Mbits. Note that the choice of a 1 second grace period for case (b), which allows most small transfers to complete as the 5 second delay constraint does, reflects that users who are transferring small files are not in a position to assess the 'cumulative throughput'. Fig.5.10 further shows an example for the case when users' delay constraint depends on file size - constraint equals to file-size/50 Kbps. One can easily see that while SD maintains a very good system performance, FS performs poorly in the overload regime. This is because by providing an equal share of resources to the transfers, almost every flow fails to complete before its expected size-dependent delay, and, furthermore, upon abortion a good portion of the file has been transferred, resulting in a very poor badput. In other words, under FS the transfers will suffer from a 'uniform' degradation of performance.

Next we turn our focus to users who are sensitive to the marginal progress being made, *i.e.,* those modeled by MPS curves. We consider users whose minimum

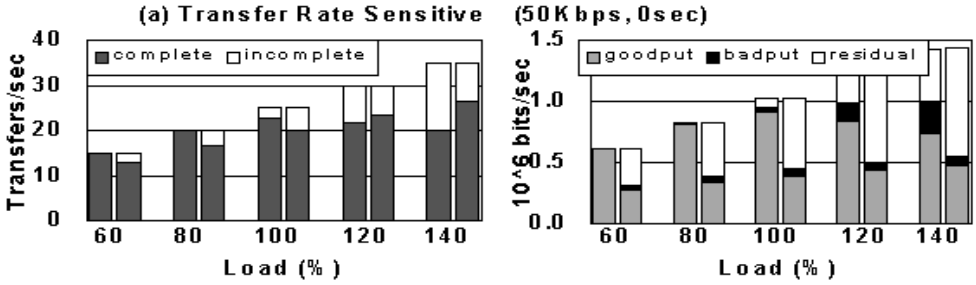


Figure 5.11: Performance under FS and SD with transfer rate (zero grace period) sensitive users.

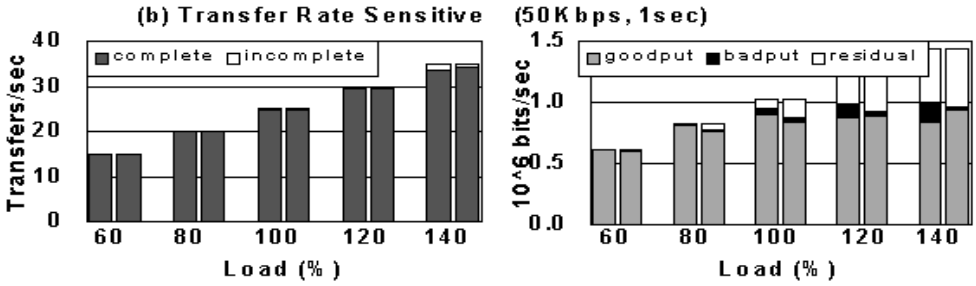


Figure 5.12: Performance under FS and SD with transfer rate (1 sec grace period) sensitive users.

expected transfer rate is 50 Kbps, but at different time scales and under different circumstances. Paralleling the previous comparisons, we plot the system performance achieved by FS and SD in Fig.5.11 through 5.14. The results shown in Fig.5.11 are for the case where users have a minimum 'instantaneous' transfer rate requirement, *i.e.*, zero grace period for evaluating the transfer rate. These results are similar to those in Fig.5.7, but with less goodput and more severe badput. This is due to the fact that the transfer rate constraint is more stringent than the cumulative throughput. Again as we introduce a 1 second time scale for users to evaluate the

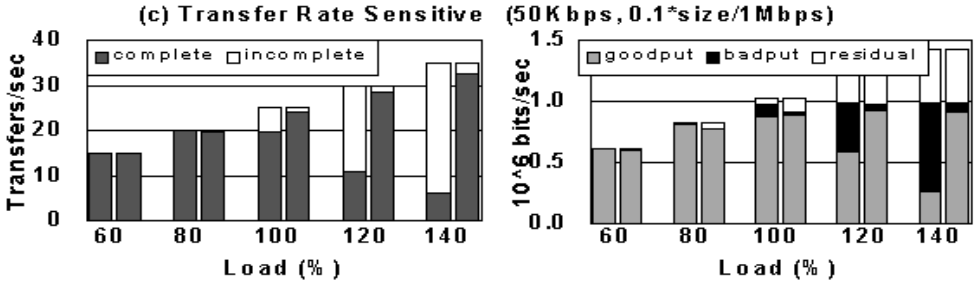


Figure 5.13: Performance under FS and SD with transfer rate (size dependent grace period) sensitive users.

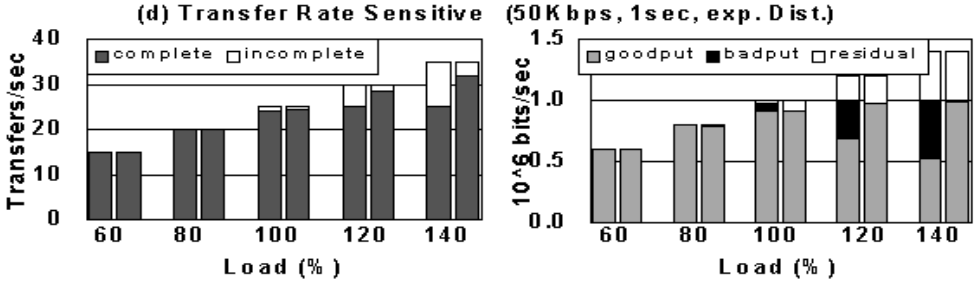


Figure 5.14: Performance under FS and SD with transfer rate (1 sec grace period) sensitive users (exponential flow size distribution).

transfer rate, the system performance can be brought back to a reasonable level, as shown in Fig.5.8. We further consider the case where the time scale to evaluate performance is, instead of fixed for all transfers, proportional to the file size, *i.e.,* users who transfer larger files will evaluate their transfer rates less frequently. The results for this case are shown in Fig.5.13. Interestingly, they are similar to those of BTD sensitive users, *i.e.,* the system performance degrades dramatically in terms of all metrics under FS, but not for the SD case.

The last scenario we explored for the case of transfer rate sensitive users is

the impact of file size distributions on performance. The results shown in Fig.5.14 were obtained with the same parameters used for those in Fig.5.12, but instead of bounded Pareto size distribution an exponential distribution with the same mean size is used. As seen the performance degrades under FS. We believe this is due to the fact that exponential distribution results in more similar mid-size file sizes than the bounded Pareto case. In turn, more transfers suffer a uniform performance degradation and thus are unable to achieve the minimum transfer rate.

Overall these results exhibit the performance impacts of various user impatience behaviors from the system's point of view. From the users' point of view, it may be important to maintain a good average BTD for completed transfers. Fig.5.15 shows the average BTD performance of completed transfers achieved under FS and SD for all impatience behaviors discussed above. As predicted, SD reduces the average BTD ranging from 1/3rd to less than 1/10th of that under FS when traffic load increases. Note that although for few impatience behaviors SD performs worse than FS in terms of goodput, the completed transfers indeed see an order of magnitude better performance in terms of BTD when the system is heavily loaded.

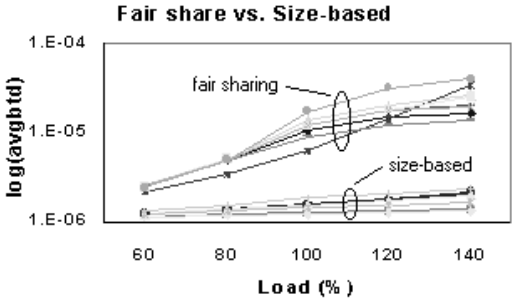


Figure 5.15: Average BTD for completed transfers under FS and SD.

5.7 Extension to Aborting Clusters of Files

Our approach can be further extended to the case where each user's perception of performance is associated with transferring 'a cluster of files', as opposed to 'individual' files. This models certain data transfer applications, such as web browsing, where each user access, *e.g.,* a web page, may contain several simultaneous[3] file transfers. A possible user perception of performance for an access, or a cluster of files, may be based on the 'completely transferred work' associated with the whole cluster. The completely transferred work at time t for a cluster of files is the total size of files within the cluster that were completed up to time t, and is denoted by $w'(0,t]$ assuming the access is initiated at time 0. Note that this definition models users who are able to process the information provided by a file only after the whole file is downloaded. As a user continuously receives completed files, *i.e.,* the web page starts coming together, his notion of acceptable performance may require that the completely transferred work exceeds the minimum expected cumulative work. We once again apply our MCS curve model introduced in §5.3 to capture such user impatience behavior associated with a cluster of files by letting $w'(0,t] < e_c(t)$ be the condition to aborted not-yet-completed transfers within the cluster at time t.

We shall use an example to illustrate the user aborting behavior with respect to the completely transferred work when FS or SD is used. Consider a batch request that arrives at time 0 and consists of three files of size 1, 2, and 3. Assume the service capacity is 1. Fig.5.16 shows $w'(0,t]$ under FS and SRPT (the extreme case

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

of SD), along with a sample MCS curve. One can see that the completely transferred work under SRPT is always above that achieved by FS. Indeed this is always true assuming the service capacity is fixed. This fact implies that under SRPT, or SD type of policies, a user who is sensitive to cumulative service is more likely to be satisfied with his performance.

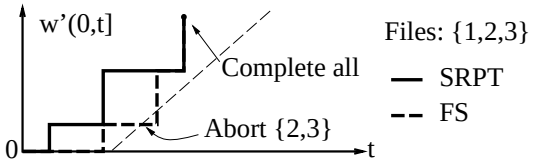


Figure 5.16: Example: User aborting behavior with respect to the completely transferred work under FS and SD.

Simulations were also conducted for this scenario with various user impatience behaviors and the two bandwidth sharing schemes as in §5.6. We shall present a representative set of results in Fig.5.17. As seen, SD performs better than FS for all traffic loads and for all performance metrics even for the case of zero grace period, recall Fig.5.7. This observation extends to other user behavior considered previously, and suggests that SD is even more beneficial when users evaluate performance on a higher level where transfers are correlated.

5.8 Concluding Remarks

Our study addressed an important yet usually neglected question: how users' response to transfer performance impacts the design of bandwidth sharing schemes. We found that with most characteristic user impatience behaviors, SD is more effective at reducing the traffic load than FS when the system is heavily or overloaded,

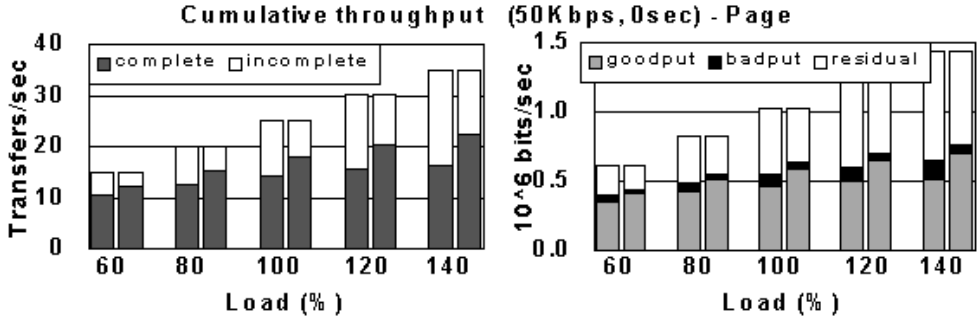


Figure 5.17: Performance achieved by FS and SD when users evaluate performance of transferring clusters of files.

and thus leads to a better network efficiency as well as user perceived performance. In particular, for a typical impatience behavior where users expect their transfer delays to be (concave) increasing in the file size, *e.g.,* BTD sensitive users, SD significantly outperforms FS by orders of magnitude in terms of, say, goodput when the system is overloaded at 140%. As network resources can be periodically, but temporarily, overloaded with data transfers, our approach ensures that users will not perceive performance degradation too badly.

Recognizing that user impatience behavior can be very complicated, an empirical study that possibly captures when and what drives users to abort their transfers will be valuable. In particular, our extension provided in §5.7 suggests that a possible research direction is to consider and validate experimentally the existence of high level or application specific user impatience models. Moreover, one may view user impatience as criteria for user self-admission control - the higher value a transfer possesses for a user, the more patient the user is. This is clearly, in our opinion, a better approach to alleviate transient overloads on best effort networks

than, say, network administered admission control, which can only maintain 'finite' number of transfer classes, presumably to capture how users value their transfers. One may refer to [8] for a discussion on measurement-based admission control for elastic flows. The results discussed in this chapter exemplify the possible impact a self admission control might have on system performance as well as network revenue.

Chapter 6

Routing

6.1 Introduction

'Static' shortest hop path routing has been employed to route elastic file transfers, *i.e.,* best effort traffic, since near the beginning of the Internet. With the increasingly dynamic traffic loads and user demands for performance, one might advocate the use of a 'dynamic' routing scheme that adapts to the changing network states, *e.g.,* the distribution of traffic loads, and enhances the average user perceived performance. In this chapter, we investigate what good state dependent routing algorithms for elastic flows might be, assuming one has a good knowledge of various aggregate state information. In particular we consider the degree of improvement in the average BTD performance that can be achieved over known approaches.

Dynamic routing for elastic flows has drawn relatively little attention as compared to that for QoS stream flows. A fundamental challenge in routing elastic flows is the inter-play between routing decisions, *i.e.,* with which other flows a new arrival will contend for bandwidth, and how network bandwidths are shared among contending flows. It is not clear whether the same routing algorithm performs well when the network bandwidths are shared according to different policies. Consider for example a parallel link network with 5 identical links, each with the same capacity of 1 Mbps. Suppose elastic flows arrive according to Poisson processes and the flow size

distribution is assumed to be bounded Pareto with mean 5 KBytes. Fig.6.1 shows the average BTD performance achieved by two routing algorithms when the flows on each link are served according to fair sharing (FS) (see Chapter 3 for definition) or First Come First Serve (FCFS), as the traffic load increases. One can see that the 'Min-Flow' algorithm, which dynamically sends arrivals to the link with fewest ongoing flows, achieves better performance for the FS case. By contrast, the 'D-Size' routing mechanism proposed in [27] for assigning jobs on parallel FCFS server networks, is clearly a better choice for the FCFS case. Without going into details the above example clearly exhibits that the choice of routing algorithm heavily depends on the underlying bandwidth allocation policy. Note that FCFS may not be a

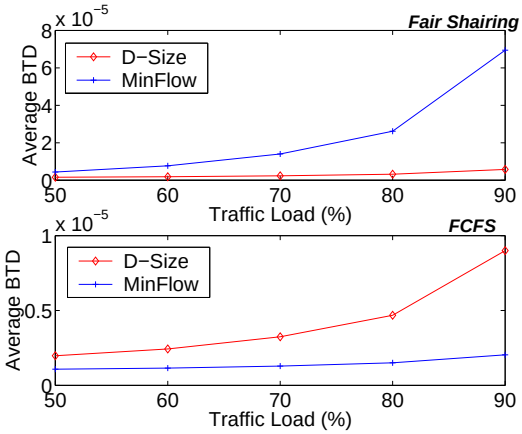


Figure 6.1: Average BTD achieved by two routing algorithms on a parallel FS or FCFS link network as the traffic load increases.

reasonable 'bandwidth allocation policy' for transferring elastic files, since files are in fact transferred on a packet by packet basis, and thus at a larger time scale they are expected to be served simultaneously towards completion. The above simple example solely illustrates that a routing algorithm must be compatible with the

bandwidth allocation policy.

Various fairness bandwidth allocation criteria have been proposed and suggested to approximate how current network bandwidth is shared among, say, TCP sessions [7, 28, 31]. Although the bandwidth allocations associated with these fairness criteria are different in the static regime, *i.e.,* for a fixed number of flows, the average user perceived performance achieved by them has been shown to be similar - see Chapter 3 and [7]. In this study we will focus on networks that allocate bandwidth according to max-min fairness and our proposed SABA criterion, which we have shown to achieve a significantly different (better) average BTD performance.

Note that for networks supporting elastic flows the routing mechanism is typically decoupled from network mechanisms that determine how bandwidth is allocated among flows. Without explicitly knowledge of the possibly time-varying bandwidth a flow will see if placed on a given route, a routing algorithm can only estimate the expected performance based on the current 'state' and the underlying bandwidth allocation policy. Studies in [38] and [43] considered networks that allocate bandwidth according to the max-min fairness criterion, and proposed heuristic routing algorithms that estimate the 'max-min fair share rate' associated with each arrival on a per link or per route basis. Unlike QoS stream flows where each flow reserves the bandwidth it needs, the path an elastic flow is routed impacts not only the bandwidth allocated to itself but also that of existing flows. Thus, in order to achieve a better 'overall' performance, we propose to further estimate the impact of route selection on not only the new arrival but also the ongoing flows.

In order to estimate the impact of routing decisions on all flows, we pro-

pose to incorporate flow size information into a dynamic state dependent routing algorithm. In particular, we will begin by examining possible routing mechanisms assuming one knows the exact size of all flows. The rationale for doing so is to understand the desired properties of a 'good' routing algorithm associated with a given bandwidth allocation policy, and how one might approximately achieve such properties in practice. Because it is difficult to analyze routing algorithms on general networks, our analysis for optimal routing focuses on a 'parallel link network' in the 'transient' regime. The parallel link model may be viewed as a simplified scenario where one chooses among parallel routes where each route has a single bottleneck with no interference traffic whatsoever. These results may then be extended to the general network case by adding controls that prevent excessive use of network resources. Alternatively, one may consider the parallel link setup as a plausible model for a 'server farm', which consists of a set of servers that run in parallel to serve file transfer requests. The transient regime analysis allows one to characterize the transient behavior of flows with different sizes, and thus provides an avenue for determining online greedy policies, where each arrival is routed so as to minimize the overall BTD for the current set of ongoing flows. With analytical results for optimal routing, even in this simplified scenario, we are more confident on the closeness of the performance improvement achieved by our proposed heuristics for more general scenarios.

This chapter is organized as follows. Related work is summarized in the next section, followed by our investigation of optimal routing for parallel link networks in §6.3. We then discuss how one may realize our findings via practical routing algo-

rithms for parallel link networks and general topologies in §6.4 and §6.5, respectively. Concluding remarks are given in §6.6.

6.2 Problem Description and Related Work

We assume the fluid flow model defined in Chapter 2 with the same notation. We focus on a routing problem that determines a set of links, *i.e.,* the route, to which a new transfer will be assigned upon arrival. Flows are assumed to remain on the same route until completion. The primary goal is to minimize the average BTD experienced by all transfers for a given underlying bandwidth allocation policy. In particular we will consider networks that allocate bandwidth according to max-min fairness and SABA.

To our knowledge the only work that aims at finding 'optimal' routing for elastic flows on general networks is [25]. This work however considers a different framework where the routing algorithm also determines how much bandwidth is allocated to each arrival, either by assigning a weight or explicitly computing the bandwidth. Furthermore, they consider a 'pseudo dynamic' regime where flows arrive sequentially to the network but do not exit[1], and the objective is to achieve 'global fairness' and maximize 'total throughput' (the sum of 'static' bandwidth allocations to flows in the network). By carefully examining their proposed somewhat complex approach, one finds that their key idea is similar to that in heuristic algorithms proposed in [38, 43], *i.e.,* to provide load balancing without excessive use of

[1] A comment was made at the end of [25] to suggest that their algorithm also works in a true dynamic regime. It is however not clear how the algorithm performs in such scenario.

network resources.

The heuristic algorithms proposed in [38] and [43] perform well for elastic flows on max-min fair networks when compared with commonly considered shortest hop path, shortest-widest path[2], and widest-shortest path[3] routing algorithms. Algorithm in [38] finds the route which has the smallest additive link cost $1/r_l^k$ where r_l is the estimated max-min fair share rate on link l and $0.5 \leq k \leq 2$ is a constant. It was then further discussed in [37] how one may use such an algorithm in a multi-service network setup where QoS stream flows and elastic flows coexist but use different routing schemes. Algorithm proposed in [43] alternatively employs a 'per-route' basis max-min fair share rate, *i.e.,* the minimum max-min fair share rate one may see on any link along a given path, and explicitly accounts for the hop count associated with each route. A comparative study later conducted in [44] showed that no clear winner between the two max-min fair share rate dependent algorithms can be identified in terms of the average throughput performance under various scenarios.

The above mechanisms provide great performance improvements over traditional approaches. It is however not clear how they perform on a SABA network, and how close they are to the optimal assuming that one knows apriori the flow size of new arrivals and possibly some aggregate 'flow-size' information associated with each link, *e.g.,* the total residual work to be done and/or the total original size of

[2]The shortest-widest algorithm finds the route(s) with the largest minimum max-min fair share rate along the route, and if there are more than one it will choose the one with the smallest hop count.

[3]The widest-shortest algorithm finds the shortest hop count route(s), and if there are more than one it will choose the one with the largest minimum max-min fair share rate

active flows. To investigate how one might incorporate such size information into a routing algorithm, we consider the optimal routing problem for a simpler scenario where flows arrive to a parallel link network, as shown in Fig.6.2. Note that when max-min fair bandwidth allocation (as well as other traditional fairness policies) is applied to parallel link network, this is equivalent to a Fair Sharing (FS) discipline on each link. By contrast, in the SABA case, link capacities will be shared based on residual size dependent weights of the flows on each link. For analysis purposes, we will consider the limiting regime where each link employs Shortest Remaining Processing Time First (SRPT) policy, *i.e.,* an infinite weight will be given to the flow with the smallest residual size, with ties broken arbitrarily.

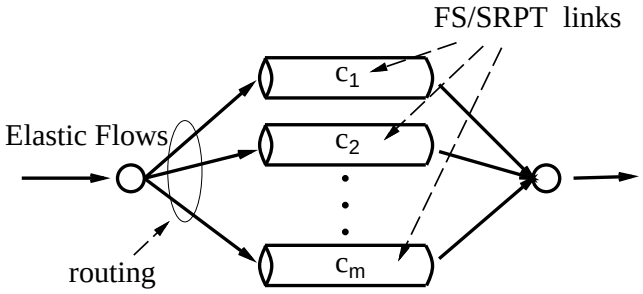


Figure 6.2: A m parallel FS/SRPT link network.

The parallel link (server) topology has been thoroughly studied in the context of the classical scheduling problems. For the most part this body of work either assumes the service discipline on each link is First Come First Serve (FCFS) or considers algorithms that determine both where to send the arrivals and how they should be served on each link. Recall our example illustrated in §6.1. The 'D-Size' algorithm proposed in [27] is a recent example of studies investigating how to send

jobs to parallel FCFS links. As we have shown in Fig.6.1, this particular good algirthm for FCFS links does not work well when the links are using, say, FS. For the latter case, it has been shown that the offline problem is NP-hard [18]. Among the various approximations, see [33] for a survey, Awerbuch *et.al.*[3] proposed an algorithm which achieves logarithmic competitiveness. This work is closest to ours. We will later compare our proposed algorithm with theirs to exhibit how close our algorithm performs to this best known algorithm. To our knowledge there has been no study that investigate routing problems on parallel link networks assuming each link employ either FS or SRPT policy. In the next section we will consider parallel FS and SRPT link networks and investigate the routing algorithms that minimize the average BTD for both cases under various scenarios.

6.3 Optimal Routing on Parallel Link Networks

6.3.1 Non-existence of Online Optimal Routing

We will begin by presenting a simple example which shows that there is no online BTD-optimal routing policy for flows arriving to a parallel FS or SRPT link network. Consider a network with 2 parallel links where both links have unit capacity, as shown in Fig.6.3. Suppose at time 0 a request with size 5 arrives and sees two existing flows with size 2 and 3 which were routed to each of the two links but have not been served - see Fig.6.3. Now consider two scenarios where another request with size 4 arrives either at time 1 or time 3, and there are no additional arrivals after that. A simple calculation shows that the BTD-optimal routings for the two new requests under the two scenarios are different if the bandwidth allocation policy

used on the links are either SRPT or FS.[4] The solid arrows in Fig.6.3 correspond to the optimal routing when the size 4 flow arrive at time 1, while a totally opposite routing decision, indicated by the dashed arrows, gives the minimum overall BTD if the size 4 flow has arrived at time 3. This exemplifies the fact that, without knowing future arrivals, one cannot determine the best routing strategy in a online manner. We formally state this fact as follows.

Fact 6.3.1. *There is no online BTD-optimal routing for arbitrary flow arrivals on a parallel FS or SRPT link network.*

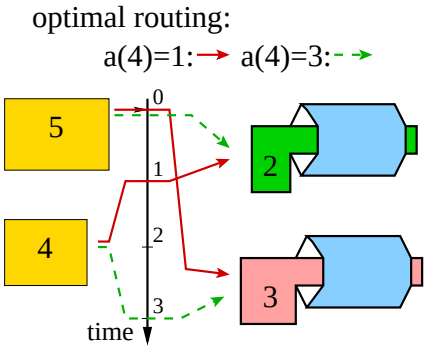


Figure 6.3: BTD-optimal routings for a 2-parallel FS/SRPT link network with different arrival times for flow with size 4.

6.3.2 Optimal Routing in the Transient Regime

The above example shows that the manner in which flows of different sizes are mixed impacts the overall performance. Since there is no online BTD-optimal policy, we shall investigate in the sequel the optimal routing for flows on parallel FS

[4]Note that in general the BTD-optimal routing on a parallel FS link network may be different from that for the SRPT case.

or SRPT link networks in the 'transient' regime, where all flows arrive and are ready to be routed/served at a given time and there are no additional arrivals thereafter. It is our intention to understand, via the transient regime analysis, what type of mixture of flow sizes might achieve good average BTD (and delay) performance, and thus to develop effective and practical routing algorithms for elastic flows, when and if the sizes are known.

We begin by discussing the similarities and differences in the overall BTD for flows on a FS or SRPT link in the transient regime. Suppose n flows arrive at time 0 to a single link with capacity c. Without loss of generality, we index the flows according to the non-decreasing order of their sizes - recall our notation defined in Chapter 2. The overall BTD for these n flows in the transient regime under SRPT and FS are given by

$$b_{\text{all}}^{\text{srpt}} = \sum_{j=1}^{n} \frac{d_j^{\text{srpt}}}{p_j} = \sum_{j=1}^{n} [p_1 + p_2 + \cdots + p_j] \cdot \frac{1}{p_j \cdot c}$$

and

$$b_{\text{all}}^{\text{fs}} = \sum_{j=1}^{n} \frac{d_j^{\text{fs}}}{p_j} = \sum_{j=1}^{n} [p_1 + p_2 + \cdots + p_j(n-j+1)] \cdot \frac{1}{p_j \cdot c},$$

respectively. Note that FS incurs an extra overall BTD of

$$\sum_{j=1}^{n} b_j^{\text{fs}} - \sum_{j=1}^{n} b_j^{\text{srpt}} = \frac{1}{c} \sum_{j=1}^{n} (n-j). \tag{6.1}$$

Thus for the same number of flows, *i.e.,* n, the overall BTD in the transient regime under FS differs from that under SRPT by a 'constant', *i.e.,* independent of the flow sizes. Despite that this constant is large, on the order of $O(n^2)$, the fact that it only depends on n leads to, as we will discuss in more detail below, a good transient

regime routing algorithm for parallel link networks with SRPT links also works well for those with FS links.

Identical Parallel Link Networks

Let us first consider the case where the parallel links are identical, *i.e.,* have the same capacity. Suppose there are m parallel links, indexed from 0 to $m-1$. The routing algorithm *Permutation A* described below is BTD-optimal in the transient regime for both the FS and SRPT link scenarios.

Algorithm 6.3.1 (Permutation A).

1. *Sort the n flows from the smallest to the largest; with ties broken arbitrarily.*
2. *Assign the jth smallest flow to link $l = \mathrm{mod}\,(j, m)$.*

Theorem 6.3.1. *On a network with m identical parallel SRPT links, Permutation A minimizes the overall BTD for flows in the transient regime.*

Theorem 6.3.2. *On a network with m identical parallel FS links, Permutation A minimizes the overall BTD for flows in the transient regime.*

The key to the proof (given in the appendix) relies on the observation that the smaller a flow is the more sensitive it is to delay, and thus one should route fewer flows with smaller sizes than that flow to the same route with it. Notice that the above observation applies not only to the case where the bandwidth allocation policy is SRPT, which gives priority to small flows, but also to the FS case in the transient regime, since smaller flows also finish earlier when link capacity is equally

shared among ongoing flows. This similarity between FS and SRPT results in the constant difference in (6.1), and hence the same routing algorithm works well for both cases.

Permutation A also minimizes the overall delay of the flows on the identical parallel link network in the transient regime. One can however find a more relaxed routing algorithm that minimizes the overall delay but not the overall BTD.

Algorithm 6.3.2 (Permutation B).

1. *Sort the n flows from the largest to the smallest; with ties broken arbitrarily.*
2. *Make $\lceil \frac{n}{m} \rceil$ groups of m flows selected consecutively from the ordered sequence; if the last group has fewer than m flows, include 'dummy' zero-size flow(s).*
3. *Assign one flow from each group to each of the m links.*

Theorem 6.3.3. *On a network with m identical parallel SRPT links, Permutation B minimizes the overall delay for n flows in the transient regime.*

Theorem 6.3.4. *On a network with m identical parallel FS links, Permutation B minimizes the overall delay for n flows in the transient regime.*

Permutation B is a relaxed version of Permutation A in the sense that no restriction is placed on how flows in the same group, *i.e.,* with similar flow sizes, should be routed to the m links. Intuitively, one needs a more restrictive mechanism when considering the BTD (rather than delay) metric since the difference in delays for flows that are similar in size impacts the overall BTD but not the overall delay. A detailed proof is given in the appendix.

Arbitrary Parallel Link Networks

Theorem 6.3.3 and 6.3.4 for the overall delay metric can be further extended to the case where the links have arbitrary capacities. The key idea in the proof of the two theorems is to let larger flows contribute less in the overall delay. The delay that might be incurred by a flow is indicated by a 'coefficient' that depends on the number of flows that are larger than that flow and are assigned to the same link according to a given routing policy. Now with general link capacities, one can incorporate the link capacities to each coefficient, and, by following the same rule as before, find the optimal routings stated below.

Algorithm 6.3.3. *Generalized Permutation B-SRPT*

1. *Compute the coefficients* $\frac{1}{c_l}, \frac{2}{c_l}, \frac{3}{c_l}, \ldots$ *for each link* l.
2. *Find the smallest* n *coefficients, and sort them from the smallest to the largest with ties broken arbitrarily.*
3. *Assign the largest flow to the link associated with the smallest coefficient, the second largest to the link with the second smallest, and so on, with ties broken arbitrarily.*

Theorem 6.3.5. *On a network with* m *parallel SRPT links each having arbitrary capacity, Generalized Permutation B-SRPT minimizes the overall delay for flows in the transient regime.*

Algorithm 6.3.4. *Generalized Permutation B-FS*

1. *Compute the coefficients* $\frac{1}{c_l}, \frac{3}{c_l}, \frac{5}{c_l}, \ldots$ *for each link* l.

2. *Find the smallest* n *coefficients, and sort them from the smallest to the largest with ties broken arbitrarily.*

3. *Assign the largest flow to the link associated with the smallest coefficient, the second largest to the link with the second smallest, and so on, with ties broken arbitrarily.*

Theorem 6.3.6. *On a network with* m *parallel FS links each having arbitrary capacity, Generalized Permutation B-FS minimizes the overall delay for flows in the transient regime.*

We have not found an analogous extension to the non-identical link network case for Theorem 6.3.1 and 6.3.2 where the target performance metric is the overall BTD.

6.3.3 Diverse Mixture of Flow Sizes for FS and SRPT Links

The transient regime BTD-optimal routing algorithms presented in the previous section suggest that by distributing flows such that each link obtains one flow from each 'size range', *i.e.,* a 'diverse' mixture of flow sizes, the system performs well in terms of minimizing overall BTD and delay. This result is somewhat opposite to the case of parallel links supporting FCFS scheduling discipline. In fact for such a network [27] proposed to distribute flows such that flows with similar sizes go to the same link while each link has roughly the same total load. The rationale is that for M/G/1-FCFS queues the average delay is proportional to the variance of flow

size distribution, and such strategy in fact reduces the variance on each queue and thus performs well. By contrast, the dynamics for flows on a FS or SRPT link is such that smaller flows will finish earlier and induce waiting/delay times for larger flows. To reduce the impact of such waiting/delay times to a given set of flows, one should prevent large flows from routed to the same link. Otherwise, the smaller ones in the set of large flows will incur excessive waiting times for the even larger ones in the same set. This suggests a diverse mixture of flow size is good for the case of FS or SRPT links. We note that for the special case where the arrivals to each FS link are Poisson, the links can be modeled by M/G/1-PS queues, and thus the average delay (or average BTD) may only depend on the mean flow size, regardless of the exact distribution. While the M/G/1-PS queue appropriately captures the stationary behavior of 'randomly' routed arrivals to parallel FS links, possibly with different probabilities, it does not apply to the case where routing mechanisms determine where to send arrivals based on, say, link loads.

Suppose one can manipulate the flow size distribution entering each link by using a routing algorithm. A reasonable question would be what types of flow size distributions lead to better average BTD for a SRPT link. As an attempt to answer this difficult question, below we pose a similar problem in the transient regime, and present our solution.

Problem 6.3.1 (Optimal Flow Size Vector). *Given a number of flows n, a total work load $w = \sum_{j=1}^{n} p_j$, and the minimum flow size p_{min}, find an ordered flow size vector $\mathbf{p} = (p_1, \cdots, p_n)$ where $p_i \leq p_j$, $\forall i < j$ such that the overall BTD of these flows on a SRPT link,* i.e., $b_{all} = \sum_{j=1}^{n} \frac{1}{p_j}(\sum_{k=1}^{j} p_k)$ *is minimized.*

The solution to this optimization problem is stated below.

Theorem 6.3.7. *The solution* $\mathbf{p}^*$ *to the Optimal Flow Size Vector problem is such that*

$$\frac{\sum_{k=1}^{j} p_k^*}{\sum_{k=1}^{j+1} p_k^*} = \left(\frac{p_{min}}{w}\right)^{(\frac{1}{n-1})}, \qquad \forall j = 1, \ldots, n-1, \tag{6.2}$$

where $p_1^* = p_{min}$. *The corresponding minimum overall BTD is*

$$b_{all} = n + (n-1)\left(\frac{p_{min}}{w}\right)^{\frac{1}{n-1}}.$$

Intuitively, one may view the n flow sizes in the Optimal Flow Size Vector problem as n samples from a flow size distribution $F_P^*(p) = \mathbb{P}(P < p)$. Consider that k out of n samples are no less than p_k^*, one can let $F_P^*(p_k^*) = \frac{k}{n}$ and construct such a distribution function. Interestingly, if one follows the above mapping scheme, a good approximate distribution function matching (6.2) can be found by considering a 'bounded Pareto' flow size distribution. For example, consider the optimal solution when $n = 10^4$, $p_{\min} = 10^3$ bits, and $w = 10^9$ bits. Fig.6.4 plots the corresponding $\bar{F}_P^*(p)$ and $\bar{F}_P(p)$ of three bounded Pareto distribution with flow size bounded by $[10^3, 10^9]$ bits and $\alpha = 0.1,\ 0.05,\ 0.01$. Note that these α values exhibit very heavy tail as compared to the typical case where $1 < \alpha < 2$. This example suggests that for a given total work load and number of flows, the set of contending flows that has a 'diverse' mixture of flow sizes may exhibit a good 'transient performance', *i.e.,* the overall BTD in the transient regime.

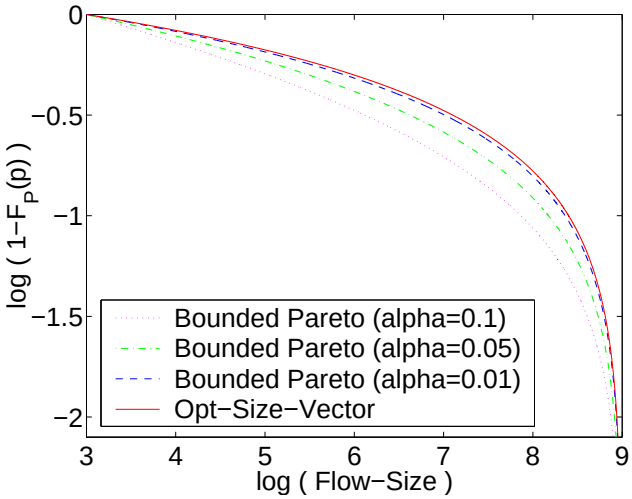


Figure 6.4: Comparing the solution of Optimal Size Vector Problem with several bounded Pareto flow size distribution functions.

The above results and our routing algorithms for parallel link networks in the transient regime suggest that one should design a routing algorithm that distributes flows on a network such that each link/route has flows with diverse mixture of flow sizes contending for bandwidth. In the next two sections we will discuss how one might implement such routing algorithms for parallel link networks and general topologies.

6.4 Flow Size Dependent Routing on Parallel Link Networks

To investigate the potential benefits of 'diverse mixture of flow sizes on each link', in this section we propose and evaluate, via simulation, an online routing algorithm that depends on flow size information on parallel FS or SRPT link networks. We emphasize that, as we suggested in §6.1, the parallel link setup not only provides insights to developing effective routing algorithms for general topologies, but also

is a plausible model of a 'server farm'. Note that servers in a server farm can be physically located close to each other and access a commonly shared file pool, or they may reside in different location (proxy servers or caches) and maintain their own copies of files. For the latter case, a more practical model may account for the consistency of the file copies and access times to geographically distributed servers. We however focus on a simplified parallel link network scenario as a mean to explore the performance impact of size-dependent routing algorithms.

6.4.1 Algorithm Description

We consider 'practical' routing algorithms that depend only on the 'present' network state, *i.e.,* on-line algorithms, and assume no buffering or re-routing of elastic flows, *i.e.,* new arrivals must be assigned to a link upon arrival and stick with it. We propose heuristic algorithms that aim at finding the link on which each new arrival will be routed such that the overall increase in BTD, *i.e.,* including that of the new arrival and those of all ongoing flows, is minimized. This type of algorithm may be considered as greedy in that it minimizes the overall 'cost' (in BTD) when there is no knowledge of the future arrivals. We define the overall BTD increase incurred by a new arrival with arrival time t as

$$\Delta b(t) = \sum_{j \in J(t+\epsilon)} b_j(t+\epsilon) - \sum_{j \in J(t)} b_j(t),$$

where $\epsilon > 0$ is an arbitrarily small value, $J(t)$ is the set of active flows that have been routed at time t, and $b_j(t) = (d_j + a_j - t)/p_j$ denotes the residual BTD of flow j if it were served according to the underlying bandwidth allocation policy, *e.g.,* FS or SRPT, with no additional arrivals after time t. Notice that for the parallel link

network the increase arises solely on the link to which the new arrival is assigned. Without loss of generality, suppose that the size p^* of the new arrival is the k_lth smallest when compared to the remaining flow sizes of the ongoing $n_l(t)$ flows on link l, and we index the ongoing flows on link l and the new arrival according to their residual sizes in non-decreasing order with ties broken arbitrarily[5]. Thus the increase in the overall BTD if the new arrival were assigned to link l at time t can be written as

$$\Delta b_l^{\text{srpt}}(t) = \left(\frac{1}{p^*} \sum_{j=1}^{k_l} p_j(t) + p^* \sum_{j=k_l+1}^{n_l(t)+1} \frac{1}{p_j} \right) \cdot \frac{1}{c_l}, \tag{6.3}$$

and

$$\Delta b_l^{\text{fs}}(t) = \Delta b_l^{\text{srpt}}(t) + \left(\sum_{j \neq k_l} \frac{p_j(t)}{p_j} + (n_l(t) - k_l) \right) \cdot \frac{1}{c_l}, \tag{6.4}$$

for the FS and SRPT link cases, respectively. One may then consider $\Delta b_l^{\text{srpt}}(t)$ ($\Delta b_l^{\text{fs}}(t)$) as the link cost associated with a SRPT (FS) link and assign the new arrival to the link with the smallest cost.

Notice that the two link costs defined in (6.3) and (6.4) provide differentiation to new arrivals based on their flow sizes. For example, in the SRPT case, a new flow which is smaller than all existing flows will find $\Delta b_l^{\text{srpt}}(t) = (\sum_{j=1}^{n_l(t)+1} \frac{1}{p_j}) p^*/c_l$, (the second term in the parenthesis) as the cost of using link l. Assuming the links have the same capacity, the link that incurs the least cost is the one currently serving fewer flows with larger sizes. This matches our intuition where a small flow should find a link with fewer flows such that it can obtain higher bandwidth, and/or

[5] With this indexing rule, $p^* = p_k = p_k(t)$.

a link that has ongoing flows with larger original size such that it will incur less penalty to these large flows in terms of BTD. By contrast, a new arrival that is larger than all existing flows will be assigned to the link which has the smallest $\Delta b_l^{\text{srpt}} = (\sum_{j=1}^{n_l(t)+1} p_j(t))/(p^* c_l)$, (the first term in the parenthesis) in which case the total residual work to be done on each link is the key concern. Hence, a large flow is likely to be assigned to a link that has less overall residual work to be done. Similar differentiation is provided by the use of the link cost for the FS case, but with more emphasis on the number of ongoing flows. This higher emphasis on the number of ongoing flows for the FS case is expected since the bandwidth allocated to each flow depends heavily on the total number of active flows on a link. The above differentiation attempts to, in addition to balancing the link loads, send small flows to links with large flows and large flows to links with small ones in a dynamic fashion, *i.e.,* based on the changing network state. This provides the 'diverse mixture of flow size' as suggested in the previous section in an online fashion.

The link costs shown in (6.3) and 6.4 however require one to determine the 'rank' of a new arrival, *i.e.,* how its flow size compares to all ongoing flows on each link. This may not be feasible in practice, since one needs to maintain an $O(n)$ states for each link and the computation complexity to obtain the link cost is $O(n \log(n))$, where n is the number of ongoing flows. We thus consider an approximation to these link costs by using only a constant number of aggregate link states. More specifically, we propose the following *'unified'* approximate link costs for both the FS and SRPT cases. Considering the aggregate link states shown in Table 6.1, we

define

$$\Delta\tilde{b}_l(t) \quad = \quad \left(\alpha_1 \cdot \frac{w_l(t)}{p^*} + \alpha_2 \cdot p^* \cdot \bar{p}_l(t) + \frac{(1+\alpha_2)}{2} n_l(t)\right) \frac{1}{c_l}, \tag{6.5}$$

where α_1 and α_2 are two 'weighting factors' ranging in $[0, 1]$ that capture the impact of the rank of a new arrival among existing flows on link l. Notice that the first two terms in the parenthesis approximate the two terms in the parenthesis of (6.3), respectively, and the last term captures the different emphasis one may place on the number of ongoing flows on each link based on the flow size of the new arrival. As we will show via simulation in the next sub-section, using this unified link cost provides good performance for both the FS and SRPT link cases.

$n_l(t) = \lvert J_l(t) \rvert$	number of ongoing flows
$w_l(t) = \sum_{j \in J_l(t)} p_j(t)$	total residual work
$\bar{p}_l(t) = \sum_{j \in J_l(t)} \frac{1}{p_j}$	total reciprocal of flow sizes
$p_{l,max}(t) = \max_{j \in J_l(t)}(p_l(t))$	maximum residual flow size
$p_{l,min}(t) = \min_{j \in J_l(t)}(p_l(t))$	minimum residual flow size

Table 6.1: Aggregate Link State Information

Clearly, the key challenge of this approximation is to determine, with the aggregate link states and flow size of the new arrival, appropriate α_1 and α_2 to achieve similar differentiation as that by using the 'exact' total BTD increase link costs defined in (6.3) and 6.4. More specifically, if the new arrival is larger than most of the flows, one should use a larger α_1 to emphasize the total residual work $w_l(t)$, and a smaller α_2 to de-emphasize the total reciprocal flow size $\bar{p}_l(t)$ and total number of ongoing flows $n_l(t)$, and vice versa. We propose the following functions

that derive the two weight factors based on the available aggregate information.

$$\alpha_1(p^*, p_{l,min}(t), p_{l,max}(t)) = \begin{cases} 0 & \text{if } p^* < p_{l,min}(t) \\ (\frac{p^*}{p_{l,max}(t)})^2 & \text{if } p_{l,min}(t) \le p^* \le p_{l,max}(t) \\ 1 & \text{if } p^* > p_{l,max}(t) \end{cases} \quad (6.6)$$

$$\alpha_2(p^*, p_{l,min}(t), p_{l,max}(t)) = \begin{cases} 0 & \text{if } p^* < p_{l,min}(t) \\ \frac{p_{l,min}(t)}{p^*} & \text{if } p_{l,min}(t) \le p^* \le p_{l,max}(t) \\ 1 & \text{if } p^* > p_{l,max}(t) \end{cases} \quad (6.7)$$

Notice that we design the two functions such that they are convexly increasing and decreasing with respect to p^* in the range of $[p_{l,min}(t), p_{l,max}(t)]$ for a given pair of $p_{l,max}(t)$ and $p_{l,min}(t)$. This is to reflect how 'the total residual work of flows small than the new arrival' and 'the total reciprocal sizes of flows that are larger than the new arrival' varies from zero to $w_l(t)$ and $\bar{p_l}(t)$, respectively, as the flow size of a new arrival increases. For example, using the afore-mentioned indexing rule, since $w_l(t) = \sum_{j=1}^{n_l(t)+1} p_j(t)$ and $p_i(t) \le p_j(t),\ \forall i < j$, one should use a convexly increasing weight factor α_1 multiplying $w_l(t)$ to approximate $\sum_{j=1}^{k_l} p_j(t)$

We now may formally state our proposed flow size dependent routing algorithm for a parallel SRPT (FS) link network.

Algorithm 6.4.1 (Minimum Estimated BTD Increase (MEBI)).

1. for each link l

2. compute $\Delta\tilde{b}_l$ for the new arrival;

3. find l^* such that $\Delta\tilde{b}_{l^*}$ is the smallest;

4. assign the new arrival to link l^*.

6.4.2 Simulation Results

We compare MEBI with several commonly considered routing algorithms and a 'close-to-optimal' algorithm proposed by Awerbuch *et.al.*[3]. Although Awerbuch's algorithm also assumes SRPT at each link[6], it requires an additional buffer to store arrivals that are not significantly smaller than those that are currently being served, and compares flow sizes again whenever a flow completes, see [3] for details. Although their algorithm may not be realistic for serving elastic file transfers, we considered it in order to evaluate how our algorithm compares with an approach which has been analytically proven to be close to optimal.

Below we summarize the algorithms under consideration. For simplicity we assume the link capacities are the same. Suppose there are m parallel links, indexed from 0 to $m-1$.

Random Route each arrival randomly (with equal probability) to one of the m links.

Round-Robin Given that last arrival was sent to link l, send the new one to link $\mathrm{mod}((l+1), m)$.

Min-Flow Route each arrival to the link that has fewest ongoing flows.

Min-Work Route each arrival to the link that has the least total residual work.

[6]The problem discussed in [3] is a job scheduling problem where the algorithm determines both where to assign each job and how they are served. Their proposed algorithm coincides with our model in the sense that jobs (flows) are served according to SRPT on each of the parallel servers (links).

Awerbuch Route each arrival to an empty link; if all links have at least one flow in service, route the arrival to the link that are currently serving a flow that has the smallest residual work if the new arrival flow size are significantly smaller than that residual size; otherwise, store the new arrival in a central buffer until one of the flows completes and then treat the smallest flow in the central buffer as a new arrival. See [3] for details.

MEBI See Algorithm MEBI and the associated link costs defined in (6.5).

For algorithms that compare link states, *i.e.,* Min-Flow, Min-Work Awerbuch, and MEBI, we resolve ties by choosing the link with the smallest index. Note that in addition to the link-state dependent algorithms, we also consider two non-state dependent algorithms, namely, Random and Round-Robin routing. We do so to examine the degree of performance improvements achieved under various scenario for systems using link state routing algorithms versus non-state dependent ones.

We conduct fluid flow simulation on a network with 5 parallel links each with a 1 Mbps capacity. The arrivals are assumed to be Poisson, with the files size distribution is either bounded Pareto or exponential, both with the same mean of 5 KBytes. The capacity of each link is shared by ongoing flows based on either FS or SABA. Recall that we consider SRPT, the limiting regime of SABA, for our theoretical analysis, but here we step back to assume SABA [7] links to present a more realistic scenario, where all ongoing file transfers may continuously progress,

[7]The weight functions and parameters for SABA used in this chapter is the same as we suggested in Chapter 3.

i.e., with positive bandwidth, towards completion. Note that we only extrapolate the routing part of Awerbuch algorithm, *i.e.,* where to route requests, and assume that once a request is assigned to a link, it will share the link capacity with others according to SABA. We will not examine the Awerbuch routing algorithm with FS links since it does not present a fair comparison.

Superior performance achieved by MEBI

Fig.6.5 through 6.8 exhibit the average BTD performance achieved by various algorithms under various scenarios as the traffic load increases. The assumed bandwidth sharing policy and file size distribution is indicated on the top of each plot. We first note that MEBI performs 'consistently' better than all other algorithms under all scenarios. In particular it is even slightly better than the Awerbuch algorithm - one that has been analytically proven to be competitive but is complex, perhaps unrealistic for elastic file transfers. Further we note that MEBI works well for both the SABA and FS link cases. This is expected since both bandwidth allocation policies exhibit similar transient behavior as discussed in §6.3.[8]

We further plot in Fig.6.9 the average performance improvement achieved by MEBI over the Min-Flow algorithm, which is generally considered as a good heuristic 'state dependent' routing algorithm for elastic flows. As seen MEBI improves the average BTD performance ranging from 1-8% as compared with Min-Flow under all scenarios as traffic load increases. Although the improvement is marginal, to our knowledge, MEBI is the only routing algorithm that 'consistently' performs better

[8] We have also fine tuned the link cost $\Delta\tilde{b}_l$ separately for the FS and SABA link case and observed slightly better average BTDs for both cases.

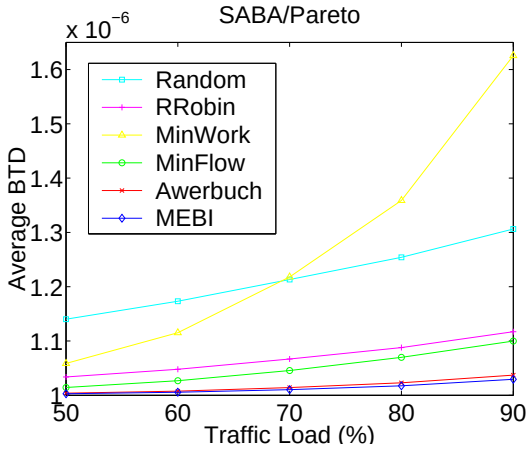


Figure 6.5: Average BTD achieved by various Routing Algorithms on SABA links and with Pareto flow size distribution.

than Min-Flow.

Consistent Performance across Links

Another benefit exhibited by MEBI is that it provides consistent average BTD performance across the parallel links. Table 6.2 below shows the coefficient of variance (CoV) with respect to the average BTD of flows across the 5 parallel links under various scenarios when the traffic load is 80%. As seen, MEBI clearly

	SABA-Pareto	SABA-Exp	Maxmin-Pareto	Maxmin-Exp
Min-Flow	0.0287	0.0731	0.0629	0.0805
MEBI	0.0019	0.0103	0.0521	0.0417

Table 6.2: CoV of average BTD across links at 80% traffic load.

provides better consistency for the average BTD performance across parallel links than Min-Flow. This may be important if, for example, one wish to provide 'fair' transfer performance experience for elastic flows sent to different servers.

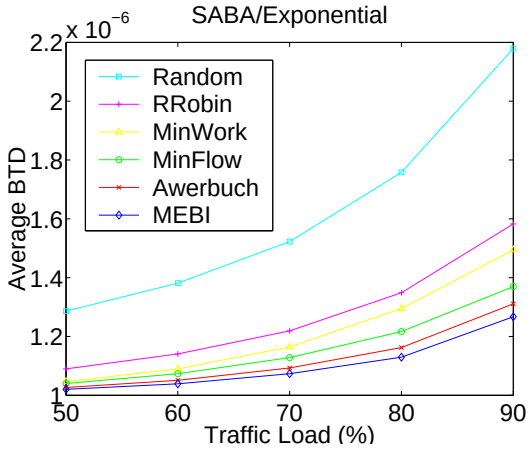


Figure 6.6: Average BTD achieved by various Routing Algorithms on SABA links and with exponential flow size distribution.

Min-Work is not robust

Next we notice that the Min-Work algorithm performs better than the open-loop algorithms but worse than the other three link state dependent ones when the file size distribution is exponential under all traffic conditions (see Fig.6.5 and Fig.6.7), but quickly becomes unacceptable as the traffic load increases for the case of bounded Pareto size distribution (see Fig.6.6 and Fig.6.8). We provide an intuitive explanation for this observation. Note that by choosing the link with the least residual work, one might send new arrivals to links that have many small flows rather than links with one or few large flows. When this happens, the BTD performance for all those small flows degrades dramatically while only one or few large flows maintain good performance. This phenomenon happens more often and aggravates when the file size distribution is bounded Pareto, since in this case most flows are small but with few very large ones.

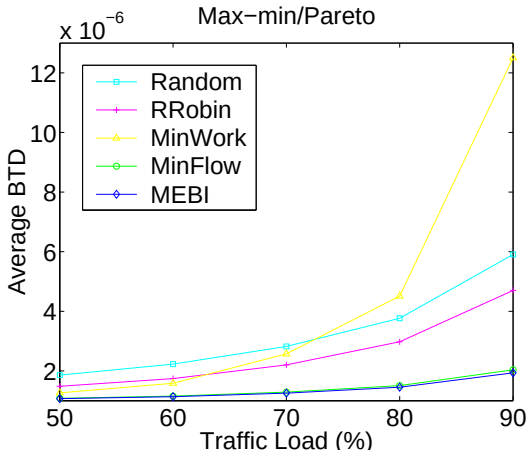


Figure 6.7: Average BTD achieved by various Routing Algorithms on FS links and with Pareto flow size distribution.

Non-state dependent algorithms

Realizing that it might be not permissible to implement state dependent routing algorithms, we also consider the performance achieved by two common non-state dependent routing algorithms versus the state dependent ones. As seen, while the performance achieved by random routing may be far away from that under the state dependent ones, the Round-Robin routing can achieve relatively close performance to those schemes. For example at 80% traffic load, the average BTD achieved by Round-Robin can be as close as 1.08 times that under MEBI (see Fig.6.5), and as far as about twice (see Fig.6.7). We have also tried to realize the 'diverse mixture of flow sizes' with size-dependent Round-Robin algorithms, but so far the trials perform similar to that under the above vanilla Round-Robin routing. The superior performance exhibited by Round-Robin as compared to random routing is likely to be due to that it 'stretches' the inter-arrival times for each link, thus is less likely to

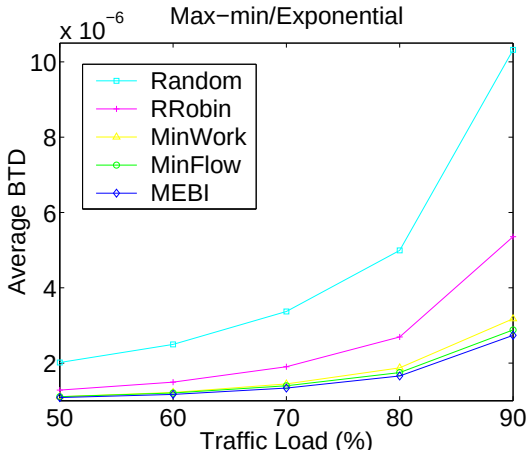


Figure 6.8: Average BTD achieved by various Routing Algorithms on FS links and with exponential flow size distribution.

have many flows sharing the same link, and in turn achieves a better load balancing.[9] We thus conclude that if it is not permissible to use state dependent routing, one may choose Round-Robin or weighted Round-Robin for a FS or SRPT parallel link network.

Impact on number of links

We have also conducted simulation on parallel link networks with different number of links. In general if one fixes the total traffic load and the aggregate capacity of the parallel links, the average BTD increases as the link number increases for all algorithms and for all scenarios. The observations and comparisons among different algorithms described above however remain the same.

[9]Round-Robin routing has also been shown [35] to be the optimal non-state dependent policy on a parallel FCFS server network with respect to the delay time when the flow sizes are independent and identically distributed according to an increasing failure rate distribution.

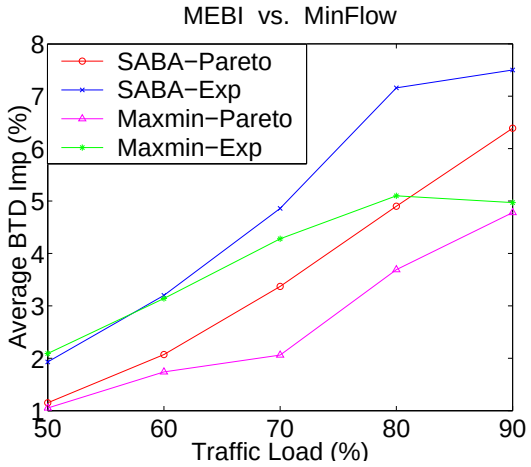


Figure 6.9: Average BTD improvement achieved by MEBI over Min-Flow under various scenarios as traffic load increases.

6.5 Flow Size dependent Routing on General Networks

In this section we extend our proposed flow size dependent routing algorithm from parallel link networks to more general topology networks. For succinctness, we will refer to networks that allocate bandwidth according to max-min fairness as max-min networks, and likewise for SABA networks.

6.5.1 Algorithm Description

A direct approach to extend our greedy algorithm from a parallel link network to a general topology is to compute the total increase in the residual BTD assuming the new arrival were routed on each possible path, and choose the one resulting in a minimum cost. This however requires an extremely high computational effort upon each arrival. For example the worst case complexity would be $O((m+n)^m)$ for the max-min case where m is the number of links and n is the max-

imum number of ongoing flows on the network. Instead one may draw an analogy from commonly known approaches that compute a 'shortest distance path' based on advertised link costs as described below.

It is well known that a key to developing a good routing algorithm on a general network is to prevent excessive use of network resources while dynamically balancing the traffic loads. A typical approach to compromise load balancing with efficient use of network resources is to utilize a 'shortest distance path' routing algorithm, where the distance associated with a path is the sum of the link costs along the path and the link cost accounts for the traffic load on each link. Alternatively, one might explicitly account for the hop counts along with the traffic load on each path to derive the path distance [43], or use 'trunk reservation' to reserve certain amount of network resources for flows traversing the 'shortest hop path' [9]. In this study, we will focus on the minimum additive link cost approach and consider a link cost that not only finds a good tradeoff between load balancing and efficient use of resources, but also account for the aggregate flow size information such that a 'diverse mixture of flow sizes' on each link/route can be achieved. We believe that the impact of incorporating the aggregate flow size information into other approaches will be similar to what we found earlier for parallel link networks.

Recall the link cost defined in (6.5). We shall present our proposed flow size dependent routing algorithm for general networks as follows.

Algorithm 6.5.1 (Shortest Estimated BTD-Increase Routing (SEBI)).

1. *Compute* $\Delta\tilde{b}_l(t)$ *for each link* l.

2. *Use Bellman-Ford algorithm to find the path*

$$r^* \in \text{argmin}\{r \in R(s_i, d_i) \mid \sum_{A_{lr}=1} \Delta\tilde{b}_l(t)\}$$

where $R(s_i, d_i)$ *is the set of possible routes connecting* s_i *to* d_i.

3. *Route the new job on path* r^*.

6.5.2 Simulation Results

We compare SEBI with a 'Shortest Max-min Fair-share Rate (SMFR)' routing algorithm which finds the route

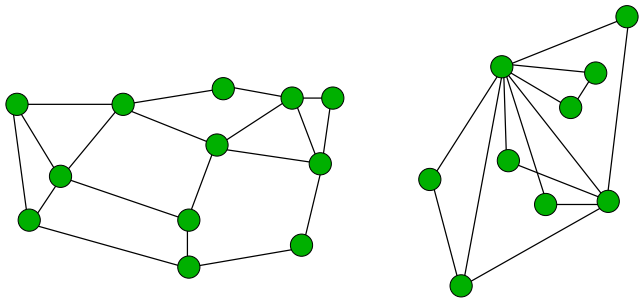
$$r^* \in \text{argmin}\{r \in R(s_i, d_i) \mid \sum_{A_{lr}=1} (n_l(t) + 1)/c_l\}.$$

The shortest MFSR path routing is a generalization of the Min-Flow algorithm discussed for the parallel link network case, and mimics the routing algorithm proposed by in [38]. Refer to [44] for a comparative simulation study of various routing alternatives, including algorithm in [38] as well as commonly considered widest-shortest path and shortest-widest path routing. Our focus here is to evaluate the performance impact of introducing the aggregate flow size information into a general framework of shortest path routing, and exhibit the performance improvements achieved by the two state dependent shortest path routing algorithms versus non-state dependent ones.

Two non-state dependent routing algorithms are considered: Fixed Shortest Hop Path (Fix-SHP) and Round-Robin Shortest Hop Path (RR-SHP) routing. The Fix-SHP is the well know static routing algorithm that routes each arrival to the

path with the smallest number of hops between the source and the destination, with ties broken arbitrarily. We also evaluate a RR-SHP algorithm which may be considered as an improvement over Fix-SHP since it finds all possible paths that has the same minimum number of hops for a given source-destination pair, and routes arrivals to these paths in a round-robin fashion - recall that round-robin routing performs well for the parallel link network case.

We conducted simulations on two representative networks, namely, the US and Europe topology [44] shown in Fig.6.10. We again assume, for all cases, Poisson arrivals and bounded Pareto or exponential flow size distribution with mean 5 KBytes. The source and destination of each arrival are chosen uniformly from all nodes in a given network.



US Topology Europe Topology

Figure 6.10: US and Europe topologies: equal capacity of 100 Mbps on each link.

Fig.6.11 through 6.14 exhibit the average BTD achieved by four afore-mentioned routing algorithms on the US and Europe topologies under various scenarios, as the total flow arrival rate increases. For each topology we consider the scenarios where the bandwidth allocation policy follows either SABA or max-min fair, and the flow

size distribution is either bounded Pareto or exponential.

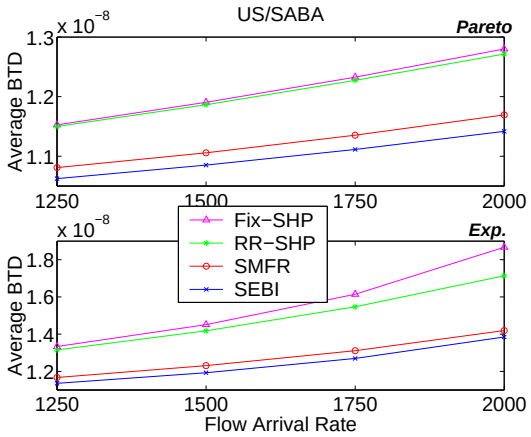


Figure 6.11: Average BTD achieved by Fix-SHP, RR-SHP, SMFR, and SEBI on the US topology with SABA bandwidth allocation as the total flow arrival rate increases.

As expected, SEBI consistently outperforms the SMFR routing under various scenarios, however the improvement is marginal - ranging from 2% to 10%. We have also conducted simulations on various arbitrary topologies as well as randomly generated ones, and similar performance improvements were observed for those (more than 10) topologies. In general, we observe more significant improvements on max-min networks than on the SABA ones. This is also expected since by using SABA, one already achieves a very good average BTD performance. Note that as exhibited in, *e.g.*, [38, 44], routing algorithms that depend on traffic load (load balancing) and number of hops on each path (efficient use of resources) achieves close performance, and none of the algorithms can outperform the others under all scenarios. The key contribution here is that our flow size dependent routing algorithm performs almost consistently better than the SMFR algorithm. Based on our

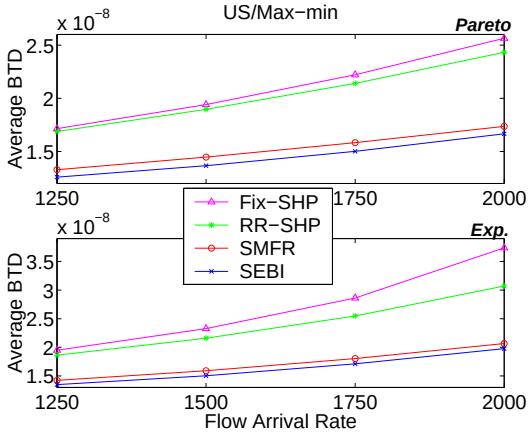


Figure 6.12: Average BTD achieved by Fix-SHP, RR-SHP, SMFR, and SEBI on the US topology with max-min bandwidth allocation as the total flow arrival rate increases.

simulations, only in about 1% cases (various topologies, traffic loads, random seeds, etc), the SEBI routing performs slightly worse than SMFR.

Also can be seen in this set of results is that the RR-SHP routing provides an non-negligible enhancement over the traditional Fix-SHP scheme, especially for the Europe topology. Considering that it requires no link state, the RR-SHP routing scheme might be a good option when it is not permissible to do state dependent routing. In fact, for the scenarios we simulated, the average BTD exhibited by RR-SHP is at most 1.5 (and as close as 1.08) of that under SMFR. We believe a further enhancement of RR-SHP may achieve even closer performance to the state-dependent ones by sending flows over a larger set of 'short' paths, *e.g.,* with hop counts within constant difference of the shortest hop count [53], in a 'weighted' round robin fashion - longer paths have smaller weights.

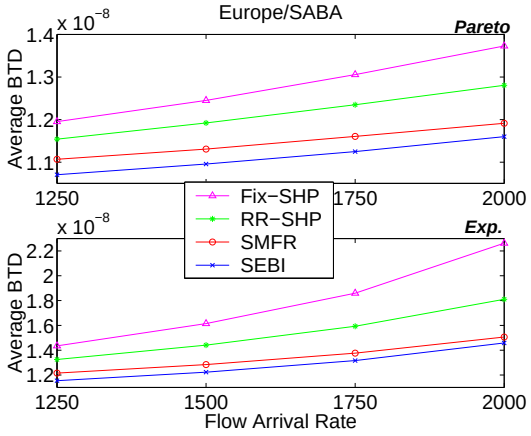


Figure 6.13: Average BTD achieved by Fix-SHP, RR-SHP, SMFR, and SEBI on the Europe topology with SABA bandwidth allocation as the total flow arrival rate increases.

6.6 Conclusion

We have investigated the routing problem for elastic flows on networks that allocate bandwidth according to either max-min fair or SABA criterion. By examining possible optimal routing policies on parallel link networks in the transient regime, we identified that routing algorithms that aim at producing a 'diverse' mixture of flow sizes at each link may achieve a better average user perceived BTD performance. An online routing algorithm in conjunction with a novel link cost is then proposed for parallel link networks to differentiate arrivals based on their sizes so as to mix flows with various sizes on each link. Via simulation we show that our proposed algorithm outperforms all other schemes we considered, including one that is analytically proven to be competitive to the offline optimal. We further extend our routing scheme to the general topology case, where using our proposed link cost allows one to achieve a good balance among 'load balancing',

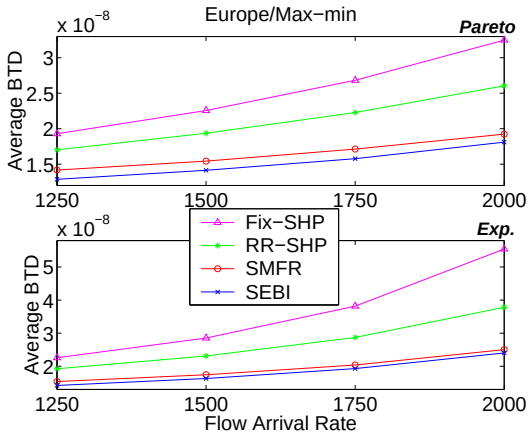


Figure 6.14: Average BTD achieved by Fix-SHP, RR-SHP, SMFR, and SEBI on the Europe topology with max-min bandwidth allocation as the total flow arrival rate increases.

'efficient use of network resources', and 'diverse mixture of flow sizes', and thus a better performance. Simulation results show that one can achieve a 2-10% average BTD performance improvement by using our proposed scheme over the best known heuristic routing algorithm for elastic flows on max-min networks. The improvements, although marginal, are consistent for all scenarios we have considered. Based on our theoretical analysis and the simulation results showing the superior performance over a competitive algorithm, we are confident that our proposed routing algorithm achieves performance close to the best one can achieve.

Knowing the best possible performance one might achieve via a state dependent routing algorithm, an interesting question to ask is whether one can achieve the same performance by using non-state dependent routing schemes for elastic flows. This problem is worth investigation especially since in practice it might not be desirable to implement expensive routing schemes that require maintenance and

advertising of link states for best effort file transfers. We conducted a preliminary investigation towards this end. We show that a routing scheme that sends flows in a round-robin fashion has the potential to achieve a close performance to state dependent ones. However further investigation on how one might enhance a vanilla round-robin routing are needed to answer the question of how close one can get.

6.7 Proofs of Theorems

A lemma will be used to prove Theorem 6.3.1 through 6.3.6.

Definition 6.7.1. *For two vectors $x, y \in (\mathbb{R}^+)^n$, we say x is cumulatively smaller than or equal to y, denoted by $x \overset{cum}{\leq} y$, if $\sum_{k=1}^{j} x_k \leq \sum_{k=1}^{j} y_k$ for all $j = 1, \ldots, n$.*

Lemma 6.7.1. *Consider an ordered vector $\alpha \in (\mathbb{R}^+)^n$ where $\forall 1 \leq i < j \leq n,\ \alpha_i \geq \alpha_j$, and two other vectors $\beta,\ \beta' \in (\mathbb{R}^+)^n$. $\sum_{j=1}^{n} \alpha_j \beta_j \ \leq \ \sum_{j=1}^{n} \alpha_j \beta'_j$ if β is cumulatively smaller than β'.*

Proof:

Let $\Delta_j = \beta_j - \beta'_j$ and $\tilde{\Delta}_j = \sum_{k=1}^{j} \Delta_k$.

$$\begin{aligned}
\sum_{k=1}^{n} \alpha_k \beta_k \ - \ \sum_{k=1}^{n} \alpha_k \beta'_k, \ &= \ \sum_{k=1}^{n} \alpha_k (\beta_k - \beta'_k) \ = \ \sum_{k=1}^{n} \alpha_k \Delta_k, \\
&= \ \alpha_n \cdot (\Delta_1 + \cdots + \Delta_n) \ + \ (\alpha_{n-1} - \alpha_n) \cdot (\Delta_1 + \cdots + \Delta_{n-1}). \\
&\quad + \ (\alpha_2 - \alpha_3) \cdot (\Delta_1 + \Delta_2) \ + \ (\alpha_1 - \alpha_2) \cdot (\Delta_1), \\
&= \ \alpha_n \tilde{\Delta}_n + (\alpha_{n-1} - \alpha_n) \tilde{\Delta}_{n-1} + \cdots + (\alpha_1 - \alpha_2) \tilde{\Delta}_1, \\
&\leq \ 0.
\end{aligned}$$

The last inequality is due to that $\alpha_n > 0$, and $\forall j = 1, \ldots, n-1,\ \alpha_j - \alpha_{j+1} \geq 0$ and $\tilde{\Delta}_j = \sum_{k=1}^{j} (\beta_k - \beta'_k) \leq 0$. □

6.7.1 Proof of Theorem 6.3.1

Without loss of generality, we assume that each of the m identical links has capacity 1, and index the flows in the non-decreasing order of their sizes with ties broken arbitrarily. Let l_j^π denote the link that flow j will be assigned to based on a routing algorithm π. The overall BTD can be expressed as

$$\sum_{j=1}^{n} \frac{1}{p_j} \cdot d_j^\pi \;=\; \sum_{j=1}^{n} \frac{1}{p_j} \cdot \sum_{k \le j, l_k = l_j^\pi} p_k.$$

Now since $\frac{1}{p_j}$ is non-decreasing in j, then according to Lemma 6.7.1, a routing algorithm π^* is BTD-optimal if it always finds the cumulatively smallest vector $d^{\pi^*} = (d_j^{\pi^*},\ j = 1, \ldots, n)$, *i.e.*, $\sum_{k=1}^{j} d_k^{\pi^*} \le \sum_{k=1}^{j} d_k^\pi$ for any other π. Clearly, the smallest possible d_1^π is p_1, the smallest possible $\sum_{k=1}^{2} d_k^\pi$ is $p_1 + p_2$ (assuming $m \ge 2$), ..., the smallest possible $\sum_{k=1}^{m+1} d_k^\pi$ is $p_1 + p_{m+1}$, and so on. This rule is equivalent to Permutation A and thus the theorem follows. □

6.7.2 Proof of Theorem 6.3.2

According to (6.1), the overall BTD of flows on a FS link is more than that on a SRPT link by $\Delta_l = \frac{n_l(n_l-1)}{2c_l}$, where n_l is the total number of flows on link l and c_l is the link capacity. With the link capacities being identical, say c, we can write the increase in overall BTD if one use FS instead of SRPT links but with the same routing algorithm as

$$\sum_{l=1}^{m} \Delta_l = \sum_{l=1}^{m} \frac{n_l(n_l-1)}{2c_l} = \frac{1}{2c} \sum_{l=1}^{m} n_l(n_l-1).$$

Since $\sum_{l=1}^{m} n_l = n$ and $n_l(n_l-1)$ is a convex increasing in n_l, $\frac{1}{2c}\sum_{l=1}^{m} n_l(n_l-1)$ is minimized when the number of flows on each link are 'balanced', *i.e.*, the same n_l

if m divides n or they differ by at most 1 otherwise. Clearly Permutation A gives the most possible 'balanced' n_ls, and, according to Theorem 6.3.1, it also minimizes the overall BTD for the SRPT link case, hence the theorem follows. □

6.7.3 Proof of Theorem 6.3.3

We again assume unit link capacities and use the notation defined in the proof for Theorem 6.3.1. Let $\beta_j^\pi = |\{k|k \leq j, l_k^\pi = l_j^\pi\}|$, the number of flows on the same link as j and are served no earlier than itself (thus includes itself) according to SRPT, associated with a routing π. We may re-write the overall delay of the n flows associated with π as $\sum_{j=1}^n p_j \cdot \beta_j^\pi$. Notice that $\beta_j^\pi = 1$ if flow j were the largest flow on its link according to π, and similarly $\beta_j^\pi = 2$ if it were the second largest, and so on. Now define the vector $p = (p_n, p_{n-1}, \cdots, p_1)$, where the values of the elements are non-increasing. According to Lemma 6.7.1, a BTD optimal routing π^* should find the cumulatively smallest vector $\beta^{\pi^*} = (\beta_n^\pi, \cdots, \beta_1^\pi)$. Clearly, this can be accomplished by sending each of the m largest flow to one of the m links, each of the next m largest to one of the m links, and so on. Note that since the βs are the same for the flows in each m flow group, how exactly these flows are assigned to the m links does not change the overall delay. The flow assignment described above is equivalent to Permutation B and thus the theorem follows. □

6.7.4 Proof of Theorem 6.3.4

The proof is essentially the same as that for Theorem 6.3.3. The only difference is that the values of βs are such that, instead of ks for the kth largest flows on each SRPT link, they are 1s for the largest flows on each FS link, 3s for the second

largest, ..., $(2k-1)$s for the kth largest, and so on. This is due to that the total delay of flows on a FS link can be written as $\sum_{k=1}^{n_l} p_k^l \cdot (2k-1)$, where p_k^l is the size of the kth largest flow on link l. The above difference does not impact the argument provided in the proof for Theorem 6.3.3. Thus Permutation B also minimizes the overall delay for flows on parallel FS links in the transient regime. □

6.7.5 Proof of Theorem 6.3.5

Again the theorem is similar to that for Theorem 6.3.3, but now with the coefficients being such that the β of the kth largest flow on link l is k/c_l where c_l is the capacity of link l. One should easily see that the BTD-optimal routing π^* that finds the cumulatively smallest vector β^{π^*} is one that assigns the kth overall largest flow to the link that has the kth smallest β coefficient. □

6.7.6 Proof of Theorem 6.3.6

The proof is the same as that for Theorem 6.3.5, with the coefficients being such that the β of the kth largest flow on link l is $(2k-1)/c_l$ where c_l is the capacity of link l. □

6.7.7 Proof of Theorem 6.3.7

Through a change of variables, $w_j = \sum_{k=1}^{j} p_k$, we may re-state the 'Optimal Flow Size Vector' problem defined in 6.3.1 as to minimize

$$b_{\text{all}} = \frac{w_1}{w_2 - w_1} + \frac{w_2}{w_3 - w_2} + \cdots + \frac{w_{n-1}}{w_n - w_{n-1}},$$

such that $p_{\min} \leq w_1 \leq \frac{w}{n}$, $kw_1 \leq w_k \leq w$, $\forall\ k:\ 2 \leq k \leq n-1$, and $w_n = w$. Note that since $\frac{\partial b_{\text{all}}}{\partial w_1} > 0$ and $\frac{\partial^2 b_{\text{all}}}{\partial w_1^2} > 0$, we know that b_{all} is strictly increasing in w_1. Thus we may set $w_1^* = p_1 = p_{\min}$. Now the problem becomes an optimization problem of $n-2$ variables, $w_2, \ldots, w_{n-1}$. By examining the Hessian of b_{all} with respective to these $n-2$ variables, one can show that $\nabla^2 b_{\text{all}}$ is positive definite. Hence the minimizer of the problem is the solution of $\nabla b_{\text{all}} = 0$ with respect to $w_2, \ldots, w_{n-1}$, and by simple calculation one can show that it is equivalent to the following condition

$$\frac{w_1^*}{w_2^*} \;=\; \frac{w_2^*}{w_3^*} \;=\; \cdots \;=\; \frac{w_{n-1}^*}{w_n^*}$$

with $w_1^* = p_{\min}$ and $w_n^* = w$. This clearly gives an unique optimal solution to the original problem by letting $w_k^* = p_{\min}(\sqrt[n-1]{\frac{w}{p_{\min}}})^{k-1}$, and the minimum is thus $b_{\text{all}} \;=\; n + (n-1)\left(\frac{p_{\min}}{w}\right)^{\frac{1}{n-1}}$. □

Chapter 7

Conclusion

In this thesis we address the question of how to optimize both network and user perceived performance on best effort networks. Although at first glance this question might seem paradoxical, in fact it is not. A key property of a typical 'best effort' transfer, *e.g.,* mediated by TCP, is that it is malleable in the sense that throughput variations throughout the transfer are tolerable, as long as the overall delay or BTD is adequate. This tolerance to throughput variations, underlies the traditional notion of 'best effort' where it is often stated that the goal is to achieve a 'fair' allocation of available resources among ongoing transfers. However, taking a user centric point of view, a more natural design objective would be to optimize the user perceived performance rather than achieve 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, realizing network mechanisms that achieve enhancements in the average delay, BTD, *etc.*, for best effort networks makes perfect sense.

Our approach to optimizing the average performance seen by users on a network is by differentiation based on the initial or residual size of the transfer. This provides a means to exploit the malleability of transfers in order to benefit

the whole. In fact, for more savvy users who expect longer delays when transferring larger files, our proposed residual size-based differentiation provides a good match of overall user/network 'utility'. Realizing a better match between users needs/wants and network allocations becomes increasingly critical when users are highly sensitive to the performance they are experiencing. Indeed in the context of interactive applications, or even web browsing, users may eventually become impatient leading to transfer interruptions, which in turn can result in poor network resource utilization. Accounting for such behavior in optimizing best effort networks is quite challenging. On the one hand, users that abort presumably have a low utility for the transfer being realized, and free up resources for others. This type of self admission control based on utility provides a good use of network resources that matches users' expectation, and perhaps leads to a higher revenue. On the other hand, when and how users interrupt their transfers depends on how resources are allocated in the network. In Harnessing this complex interaction between bandwidth allocation and user impatience behavior, we found again that exploiting the malleability, *i.e.,* size-based differentiation, offers good middle ground between complexity and enhancing both system and user perceived performance.

Various possibilities, in addition to our proposed transport level and routing mechanisms, exist to realize size-based differentiation. Preserving the spirit of 'simplicity' that has been used to implement control mechanisms on best effort networks, we believe end-system controls of file transfers are more appropriate, *i.e.,* rather than introducing complexity in the core of the network, *e.g.,* per-flow state. For example, one may employ SRPT scheduling for packets of ongoing transfers

on the server side [21], or initiate parallel connections of different numbers through browser implementation, both depending on the file size or even the congestion status. These approaches exhibit various pros and cons, but presumably achieve more or less the same potential benefit we have exhibited through our fluid flow simulations of transport level differentiation. Further extensions of size-based differentiation may account for, say, application level utility, *e.g.,* user might be more sensitive in delay while retrieving an instant message than downloading a backup file. Our study provides an avenue and framework for further investigation of possible implementations of such extensions.

Our thorough investigation of size-based differentiation under various scenarios have qualified its performance benefits, as compared to traditional approaches emphasizing on instantaneous fairness. While the performance gains range from very marginal (2%) to impressively significant (80%), it promises a *robust* performance under various traffic load conditions, peak rate constraints, and bottleneck scenarios. Furthermore, it offers a graceful performance degradation when facing transient overloads, which may be unavoidable on today's and future Internet due to the (multi-scale) burstiness of the traffic load. As long as the global exchange of files is built upon a heterogeneous environment, which is a key to its success, providing a robust and good performance through size-based differentiation is perhaps one of the simplest approaches to better their overall operation.

Bibliography

[1] Network Simulator (NS-2). *available via http://www.isi.edu/nsnam/ns/.*

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

[3] B. Awerbuch, Y. Azar, S. Leonardi, and O. Regev. Minimizing the flow time without migration. *Proc. 31st Annual ACM Symp. Theory of Computing*, pages 198–205, 1999.

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

[5] M. Bender, S. Chakrabarti, and S. Muthukrishnan. Flow and stretch metrics for scheduling continuous job streams. *Proc. 9th Annual ACM-SIAM Symp. Discrete Algorithms*, 1998.

[6] Y. Bhumralkar, J. Lung, and P. Varaiya. Network adaptive TCP slow start. available via http://www.path.berkeley.edu/~varaiya/comm.html.

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

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

[9] S. Oueslati Boulahia and J.W. Roberts. Impact of trunk reservation on elastic flow routing. *Proc. Networking 2000 LNCS 1815.*

[10] T. Bu and D. Towsley. Fixed point approximations for TCP behavior in an AQM network. *Proc. ACM SIGMETRICS*, 29(1):216–225, 2001.

[11] M. Butto, E. Cavallero, and A. Tonietti. Effectiveness of the leaky bucket policing mechanism in ATM networks. *IEEE JSAC*, 9(3):335–342, 1991.

[12] N. Cardwell, S. Savage, and T. Anderson. Modeling the performance of short tcp connections. *Technical Report, Computer Science Department, Washington University*, 1998.

[13] J. W. Cohen. The multiple phase service network with generalized processor sharing. *Acta Informatica*, 12:245–284, 1979.

[14] 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*, pages 3–26, 1998.

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

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

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

[18] J. Du, J. Y. T. Leung, and G. H. Young. Minimizing mean flow time with release time constraint. *Theoretical Computer Science*, 75(3), 1990.

[19] A. Feldmann et.al. Performance of web proxy caching in heterogeneous bandwidth environment. *Proc. IEEE INFOCOM*, 1:107–116, 1999.

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

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

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

[23] 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.

[24] 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*, pages 189–197, 1999.

[25] A. Goel, A. Meyerson, and S. Plotkin. Combining fairness with throughput: Online routing with multiple objectives. *Proc. 32nd Annual ACM Symp. Theory of Computing*, 2000.

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

[27] M. Harchol-Balter, M.E. Crovella, and C.D. Murta. On choosing a task assignment policy for a distributed system. *Proc. Performance Tools*, 1468:231–242, 1998.

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

[29] V. Jacobson and M. Karels. Congestion avoidance and control. In *Proc. ACM SIGCOMM*, pages 314–29, Aug. 1988.

[30] Raj Jain. *The Art of computer Systems Performance Analysis.* John Wiley and Sons, 1992.

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

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

[33] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and scheduling: Algorithms and complexity. *Handbooks in Operations research and management science*, 4:445–522, 1993.

[34] S. Leonardi and D. Raz. Approximating total flow time on parallel machines. *Proc. 31st Annual ACM Symp. Theory of Computing*, pages 110–119, 1999.

[35] Z. Liu and D. Towsley. Optimality of the round robin routing policy. *Journal of the Applied Probabilities*, 31:466–475, 1994.

[36] S.H. Low and D.E. Lapsley. Optimization flow control I: Basic algorithm and convergence. *IEEE/ACM Trans. Networking*, 7(6):861–874, 1999.

[37] Q. Ma and P. Steenkiste. Supporting dynamic interclass resource sharing: a multi-class qos routing algorithm. *Proc. IEEE INFOCOM*, 1999.

[38] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. *ACM Computer Communication Review*, 26(4):206–217, 1996.

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

[40] I. Matta and L. Guo. Differentiated predictive fair service for TCP flows. *Technical Report BU-CS-2000-012, Computer Science Department, Boston University*, 2000.

[41] J. Mo and J. Walrand. Fair end-to-end window based congestion control. *IEEE/ACM Trans. Networking*, 8(5):556–567, 2000.

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

[43] S. Oueslati-Boulahia and E. Oubagha. An approach to routing elastic flows. *Proc. ITC*, 1999.

[44] S. Oueslati-Boulahia and E. Oubagha. A comparative study of routing algorithms for elastic flows in a multiservice network. *Proc. ITC Specialist Seminar on IP Traffic*, 2000.

[45] V. Paxon. End-to-end routing behavior in the Internet. *IEEE/ACM Tran. Networking*, 5(5):601–605, 1997.

[46] V. Paxon and S. Floyd. Difficulties in simulating the Internet. *IEEE/ACM Trans. Networking*, 2001.

[47] R. Perera. The variance of delay time in queueing system M/G/1 with optimal strategy SRPT. *AEU, Archiv fuer Elektronik und Uebertragungstechnik*, 47(2):110–114, 1993.

[48] J. Roberts. Flow aware networking for effective quality of service control. 2000. available via http://www.ima.umn.edu/reactive/fall/networks.html.

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

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

[51] S. Shenker. Fundamental design issues for the future Internet. *IEEE J. Select. Areas Commun.*, 13(7):1176–88, Sept. 1995.

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

[53] X. Su and G. de Veciana. Dynamic multi-path routing: Asymptotic approximation and simulations. *Proc. ACM SIGMETRICS*, 29(1):25–36, 2001.

[54] M. Vojnović, J. Y. Le Boudec, and C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogenous round-trip times. *Proc. IEEE INFOCOM*, 2000.

[55] S. Yilmaz and I. Matta. On class-based isolation of udp, short-lived and long-lived tcp flows. *Proc. MASCOTS*, 2001.

[56] Y. Zhang, L. Qiu, and S. Keshav. Speeding up short data transfers: Theory, architectural support, and simulation results. *Proc. NOSSDAV*, 2000.

Vita

Shan-Chieh Yang was born in Taipei, Taiwan on February 13, 1973, the son of Chie-chong Yang and Betty Yu-Mei Fan. He graduated in 1991 from Taipei Municipal Chien-kuo Senior High School and entered National Chiao-Tung University. He received his B.S. degree in Electronics Engineering in 1995. After graduation, he had been a research assistant for Professor C. Bernard Shung for one year. He then entered the Department of Electrical and Computer Engineering at the University of Texas at Austin and received his M.S. degree in 1998. From Summer 1997 to Fall 2001, Mr. Yang worked with Dr. Gustavo de Veciana on various graduate research projects, and earned his Ph.D. on December 2001.

Permanent address: 5829 Westmont Drive
Plano, Texas 75093

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.