# Differentiated Predictive Fair Service for TCP Flows*

IBRAHIM MATTA    LIANG GUO
Computer Science Department
Boston University
Boston, MA 02215, USA
{matta, guol}@cs.bu.edu

## Abstract

*The majority of the traffic (bytes) flowing over the Internet today have been attributed to the Transmission Control Protocol (TCP). This strong presence of TCP has recently spurred further investigations into its congestion control mechanism and its effect on the performance of short and long data transfers. In this paper, we investigate the* interaction *among short and long TCP flows, and how TCP service can be improved by employing a low-cost service-differentiation scheme. Through control-theoretic arguments and extensive simulations, we show the utility of* isolating *TCP flows into two classes based on their lifetime/size, namely one class of short flows and another of long flows. With such class-based isolation, short and long TCP flows have separate service queues at routers. This protects each class of flows from the other as they possess different characteristics, such as burstiness of arrivals/departures and congestion/sending window dynamics. We show the benefits of isolation, in terms of better predictability and fairness, over traditional shared queueing systems with both tail-drop and Random-Early-Drop (RED) packet dropping policies.*

## 1 Introduction

In order to improve Internet services, enhancements to the Internet's basic best-effort architecture have been recently proposed, most notably, the Intserv and Diffserv architectures. In the Intserv architecture [31], routers are stateful, i.e. they maintain state information about each communication flow of packets. Thus, per-flow delivery guarantees can be provided at the expense of complexity at routers. The Diffserv architecture [4, 10, 34], however, requires only access (border) routers at the edge of the network to maintain per-flow information, whereas routers in the core of the network are kept essentially stateless. In this paper, we concern ourselves with Diffserv-like architectures and how they can best manage TCP flows[1], which constitute the majority of the traffic volume (80-90%) on the Internet today [35].

Recent measurements of Internet traffic [29] show that the length (in terms of both lifetime and transfer size) of TCP flows follows a heavy-tailed distribution, i.e. only a small percentage (e.g. less than 20%) of flows are long-lived (e.g. more than 20 packets), but they carry a large percentage (e.g. 85%) of the total traffic (in bytes). This calls for a careful design of the network so that these few long-lived TCP flows are managed well. Furthermore, many recent proposals (e.g. [10, 15]) have attempted to provide long-lived TCP flows with predictive (or controlled-load) service, where a TCP flow is statistically allocated a certain rate/bandwidth. In these proposals, it has been shown that a TCP flow has to operate in a predictable mode, otherwise it may not be able to take advantage of its reservation. This predictability is hard to achieve in the presence of the many short-lived TCP flows, which have a more bursty flow arrival/departure process [14, 35] and more drastic sending window dynamics.

In this paper, we argue for an architecture that *isolates* short-lived and long-lived TCP flows into two classes. This can be implemented, for example, by using a class-based queueing (CBQ) scheme [17], or by routing these two classes of flows on (logically) separate communication paths [34]. See Figure 1 for an illustration of isolation control. Edge routers will be responsible for classifying flows and marking packets as belonging to long-lived or short-lived flow. Once a flow is classified into a long flow (e.g. after a certain number of packets from that flow are observed), edge routers will be able to direct the recognized flow to a new path, for example, by establishing a label-switched path using MPLS [6, 2, 34].

We show using control-theoretic arguments and extensive simulations the utility of such class-based isolation

*This work was supported in part by NSF grants CAREER ANI-0096045 and MRI EIA-9871022.

[1]We use the terms "flow" and "connection" interchangeably throughout the paper. A flow can be generally defined as a sequence of packets which share some common properties. In this paper, we define a flow as packet sequence between the same source host/network and destination host/network pair.

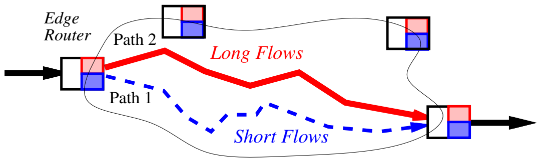



**Figure 1.** Edge routers perform isolation control

scheme. We show that fairness and predictability are improved for both short and long TCP flows. Furthermore, by allocating more bandwidth to the class of short flows, which are usually interactive and delay-sensitive, we can significantly reduce their response time. Our work differs from previous work (reviewed in Section 5) in that we study the benefits of class-based isolation to not only long TCP transfers[2], but also to short TCP transfers, which constitute a large percentage of TCP flows on the Internet today.

The rest of the paper is organized as follows. Section 2 uses control-theoretic arguments (similar to [21, 30]) to establish the benefits of isolation (or individual) control over traditional aggregate control, where flows of different characteristics (e.g. size) are mixed together to share the same resources (e.g. buffer, transmission capacity). Section 3 describes our simulation model and experiments. Section 4 supports our claims by extensive simulations of TCP-reno flows. Section 5 discusses related work, and Section 6 concludes the paper. Due to space limitations, we refer the reader to [23, 22] for the derivation of stability conditions in Section 2, and for more simulation results.

## 2 Aggregate versus Individual Control

In this section, we formulate two discrete-time control models. In the first model (Section 2.1), each flow controls its input traffic rate based on the *aggregate* state of the network due to all $N$ flows. In the second model (Section 2.2), each flow (or class of homogeneous flows) controls its rate based on its own *individual* state within the network. Here we do not model TCP flows whose control generally leads to oscillatory behavior [8], rather we assume PID control[3] for which a stable solution exists under certain conditions [27, 30, 21]. The motivation behind this assumption is to keep the models simple while still gaining insights into the fundamental differences between aggregate and individual controls, and how these differences can affect the performance of TCP flows of different size.

In the aggregate control model, the number of new packets that a flow $i$ sends into the network at time step $k$, denoted by $w_i^k$, is adapted based on the difference between a target *total* buffer space, denoted by $B$, and the current *total* number of outstanding packets, denoted by $q^k$. In the individual control model, $w_i^k$ is adapted based on flow (or class) $i$'s target, denoted by $B_i$, and its current number of outstanding packets, denoted by $q_i^k$. We denote by $c^k$ the total number of packets acknowledged at time step $k$, and by $c_i^k$ the number of flow/class $i$ packets acknowledged at step $k$. In what follows, for each control model, we determine conditions under which the system stabilizes. We then solve for the values of the state variables at equilibrium, and show whether fairness (or a form of weighted resource sharing) can be achieved. Table 1 lists all system variables along with their meanings.

[2] Many TCP studies assume *infinite* transfers.

[3] PID control is a controller where the control signal is a linear combination of the error, its integral, and derivative [27].

**Table 1. Table defining system variables**

| Variable | Meaning |
|---|---|
| $N$ | total number of flows (or classes of homogeneous flows) |
| $w_i^k$ | number of new packets of flow/class $i$ at step $k$ |
| $q_i^k$ | number of outstanding packets of flow/class $i$ at step $k$ |
| $c_i^k$ | number of flow/class $i$ packets acknowledged (i.e. no longer outstanding) at step $k$ |
| $q^k$ | total number of outstanding packets at step $k$ |
| $c^k$ | total number of packets acknowledged (served) at step $k$ |
| $B$ | target total buffer space |
| $B_i$ | target buffer space allocated to flow/class $i$ |
| $\alpha_i, \beta_i$ | parameters controlling increase and decrease rates of $w_i^k$ |

### 2.1 Aggregate Control or Sharing

Under aggregate PID control, the evolution of the system state is described by the following difference equations:

$$\begin{aligned} w_i^k &= w_i^{k-1} + \alpha_i(B - q^k) + \beta_i[(B - q^k) - (B - q^{k-1})] \\ q^k &= q^{k-1} + \sum_{i=1}^{N} w_i^{k-1} - c^{k-1} \end{aligned} \quad (1)$$

For simplicity, assume a constant acknowledgment (service) rate, i.e. $c^k = C$ for all $k$. Let $\alpha$ and $\beta$ be the maximum of $\alpha_i$ and $\beta_i$, respectively. Then, it can be shown that this system is stable if the following condition is satisfied [23]:

$$\alpha + 2\beta < \frac{4}{N} \quad (2)$$

Otherwise, the system does not converge to a stable state. This indicates that the existence of flows which rapidly change their sending rates through high values of $\alpha_i$ and/or $\beta_i$ can cause the system to become unstable. This suggests that short TCP flows, which aggressively change their sending windows in slow start phase, may affect the stability of long TCP flows, which change their sending windows cautiously in congestion avoidance mode, in a system that mixes both kinds of TCP flows. Furthermore, in such a system, the value of $N$ may be high so as to cause instability.

We now derive the values of the state variables at equilibrium given that the system is stable, i.e. the system satisfies equation (2). Denote by $(w_i)_s$ and $q_s$ the steady-state values of $w_i^k$ and $q^k$, respectively. Then, at equilibrium, we have from equations (1):

$$\begin{aligned}(w_i)_s &= (w_i)_s + \alpha_i(B - q_s) + \beta_i[(B - q_s) - (B - q_s)] \\ q_s &= q_s + \sum_{i=1}^{N}(w_i)_s - C \end{aligned} \quad (3)$$

Thus, at equilibrium, $q_s = B$ and $\sum_{i=1}^{N}(w_i)_s = C$. In other words, the system converges to a state where the total input rate matches the total service rate, and the total buffer space is full.

We note that if $\alpha_i = \alpha$ and $\beta_i = \beta$ for all $i$, then $w_i^k$ changes by the same amounts for every flow $i$. Consequently, if we start the evolution of the system with $w_i^0$ being the same for all flows, only then we have equal sharing of the network at steady-state, i.e. $(w_i)_s = \frac{C}{N}$, regardless of the initial values of $q_i^0$. However, in general, when $w_i^0$ are not equal for all flows, the system converges to an *unfair* state, more precisely, to a state where

$$(w_i)_s = w_i^0 + \frac{C - \sum_{j=1}^{N} w_j^0}{N} \quad (4)$$

To summarize, controlling several flows by observing the resulting aggregate state of the network may lead to instability due to either the existence of flows which are rapidly adjusting their sending rates, or a high number of flows competing for the same *shared* resource. Furthermore, even if the system is stable, the system is highly likely to converge to an unfair state where flows receive unequal shares of the resource.

### 2.2 Individual Control or Isolation

Under individual PID control, the evolution of the system state is described by the following difference equations:

$$\begin{aligned}w_i^k &= w_i^{k-1} + \alpha_i(B_i - q_i^k) + \\ &\qquad \beta_i[(B_i - q_i^k) - (B_i - q_i^{k-1})] \\ q_i^k &= q_i^{k-1} + w_i^{k-1} - c_i^{k-1}\end{aligned} \quad (5)$$

Recall that under individual control, flow/class $i$ regulates its input, $w_i^k$, based on its *own* number of outstanding and acknowledged packets. For simplicity, assume a constant acknowledgment (service) rate, i.e. $c_i^k = C_i$ for all $k$. Following the same stability analysis to derive equation (2) [23], it is easy to see that flow/class $i$ stabilizes if the following condition is satisfied:[4]

$$\alpha_i + 2\beta_i < 4 \quad (6)$$

[4]We set $N = 1$ in equation (2).

Observe that, unlike aggregate control, flows/classes are *isolated* from each other. Therefore, the existence of flows/classes, which rapidly change their sending rates through high values of $\alpha_i$ and/or $\beta_i$ (e.g. short TCP flows), does not affect the stability of other flows (e.g. long TCP flows). This isolation can be implemented using, for example, a class-based queueing (CBQ) discipline [17]. In such a CBQ system, each class of homogeneous flows can be allocated its own buffer space and service capacity.

We now derive the values of the state variables of flow/class $i$ at equilibrium given that it stabilizes, i.e. flow/class $i$ satisfies equation (6). Denote by $(w_i)_s$ and $(q_i)_s$ the steady-state values of $w_i^k$ and $q_i^k$, respectively. Then, at equilibrium, we have from equations (5):

$$\begin{aligned}(w_i)_s &= (w_i)_s + \alpha_i(B_i - (q_i)_s) + \\ &\qquad \beta_i[(B_i - (q_i)_s) - (B_i - (q_i)_s)] \\ (q_i)_s &= (q_i)_s + (w_i)_s - C_i\end{aligned} \quad (7)$$

Thus, at equilibrium, $(q_i)_s = B_i$ and $(w_i)_s = C_i$. In other words, each flow/class $i$ converges to a state where its input rate matches its allocated service rate, and its allocated buffer space is full. We note that if the allocated buffers $B_i$ and service capacities $C_i$ are equal, then every flow receives an equal share of the resources, *regardless of* the initial values of $w_i^0$ and $q_i^0$. One can also achieve a weighted resource sharing by assigning different $B_i$ and $C_i$ allocations. Thus, a flow/class with higher priority (e.g. short interactive TCP flows) can be allocated more resources, so as to receive better throughput/delay service.

To summarize, controlling each flow (or class of homogeneous flows) by observing its own individual state within the network provides isolation between them. Thus, stability can be achieved for a flow/class regardless of the behavior and number of other flows/classes. Furthermore, the system can converge to a fair state where flows/classes receive a weighted share of the resources.

## 3 Simulation Model and Experiments

As pointed out in Section 1, recent measurements of Internet traffic [29] show that the length (in terms of both lifetime and transfer size) of TCP flows follows a heavy-tailed distribution, where few long-lived TCP flows carry most of the bytes. Short flows behave very differently compared to long flows. First, some measurement studies [14, 35] suggest that short-lived flows arrive to the network in a more bursty fashion than long-lived flows. In addition, for TCP traffic, which contributes most of the Internet traffic today, long-lived flows, which typically acquire enough knowledge about the congestion state of the network, spend most of their time in *congestion avoidance* phase, while short-lived flows mainly transmit in *slow start* phase [7]. In other words, short flows generally finish their transmission before they can adapt to their fair share of the bandwidth. Generally speaking, the TCP congestion window changes more drastically during slow start than during congestion avoidance. However, because short TCP flows generally attain smaller window sizes, they generate smaller packet bursts (albeit more variable in size) than long TCP flows.

These differences in the behavior of short and long TCP flows suggest that it may be beneficial to treat network flows differently based on their size. More precisely, isolating short flows from long ones promises to reduce the volatility (seen by long flows) of the state of congestion in the network created by the burstier arrivals of short TCP flows and their more drastic window dynamics (i.e. variation in packet burst sizes). Thus, the few well-behaving (more stable) long-lived TCP flows, which are carrying most of the bytes, can be protected and provided better service.

Furthermore, with isolation, short TCP flows are also protected from long TCP flows, which generally attain larger window sizes (or generate larger packet bursts). This reduces the chances that short TCP flows are completely shut off by long flows, which would increase their response time. This is clearly undesirable for short interactive/delay-sensitive flows. This undesirable situation has been observed when both shared tail-drop queues and Random-Early-Drop (RED) queues are deployed [16, 5], where sources generating smaller packet bursts (i.e. short flows with smaller window sizes) are penalized.

Our objective is to investigate the effect of isolating (or splitting) a set of short and long flows into two (size-homogeneous) classes, namely short class and long class, in the absence and presence of background traffic. We then measure various performance metrics, including fairness within each class. We obtain our simulation results using the UCB/LBNL/VINT Network Simulator—*ns* (version 2) [36]. All simulations are for the TCP-reno version [32, 33]. Simulations with TCP-tahoe support the same conclusions. In our experiments, we consider links with FIFO queues employing either tail-drop or RED packet dropping policy. Unless otherwise specified, packet queues are assumed to deploy a tail-drop policy.

### 3.1 Flow Assignment Policies

We compare three traffic management schemes employed at an edge router distributing flows over two parallel paths (cf. Figure 1): (1) a *load-balanced strategy* where incoming flows are randomly distributed with equal probability over the two paths; (2) a *size-based splitting strategy* where short flows, whose transfer volume in bytes is below a certain size threshold, are routed on one path (Path 1) and long flows on the other (Path 2); and (3) a *lifetime-based splitting strategy* where each flow is first routed on Path 1, then if the flow is still active after some time threshold or a certain number of packets (e.g. 30 packets are observed from that flow), that flow is considered long and is routed on Path 2.

When a splitting (isolation) policy is used, unless otherwise specified, a size-based splitting strategy is used. Note that size-based splitting assumes that TCP flows are classified based on the exact size of the transfer, rather than by classifying a flow as long-lived after observing it for some time or for a certain number of packets. The latter mechanism, used in lifetime-based splitting, would likely be the one used in practice [29]. However, it is not straightforward to quantify the effect of a re-routed long flow on the ongoing group of long flows. Thus, in this paper, we also use size-based splitting to exclude these effects and examine isolation if we indeed have a "clean" split among short and long flows.

### 3.2 Performance Measures

We consider the following performance measures: (1) *effective throughput (or goodput)* to measure the rate of successfully transmitted (i.e. acknowledged) packets; and (2) *fairness* within each class (i.e. among size-homogeneous flows). To measure fairness, we use Chiu and Jain's fairness index [8], which is defined as follows: if there are $N$ connections competing for a bottleneck resource, and the goodput achieved by connection $i$ is $x_i$, then the fairness index $f$ is given by:

$$f = \frac{(\sum_{i=1}^{N} x_i)^2}{N \sum_{i=1}^{N} x_i^2}$$

The closer the value of $f$ is to 1, the more fair the resource allocation is (i.e. the values of $x_i$'s are closer to each other). Throughout simulation lifetime, we measure the goodput and fairness index over 20-second intervals, and either plot these instantaneous values as a function of time, or plot the average value.

## 4 Simulation Results

Simulation results in Sections 4.2 and 4.3 compare isolation and sharing for varying network pipe and buffer sizes in the presence of telnet background traffic. Section 4.4 presents performance results with web background traffic.

### 4.1 General Observations

Before presenting the details of our simulation experiments, we summarize our observations:

- Isolating short flows from long ones can reduce the load variation on the route taken by long flows. This is because long flows are then protected from the more bursty arrivals/departures of short flows and their more drastic sending window dynamics. Thus isolation can improve the predictability of the service provided to long flows as well as fairness among them.
- Class-based isolation of short and long TCP flows also protects short flows from being completely shut off

by long TCP flows, which generally generate larger packet bursts due to their larger sending windows. Thus isolation can also improve the predictability of the service provided to short flows as well as fairness among them.

- Class-based isolation provides short TCP flows with much better fairness and response time than that provided in a shared RED queueing system, since RED usually penalizes flows generating smaller packet bursts.
- Class-based isolation can improve service predictability and fairness without sacrificing the overall goodput and utilization of the network. This is especially true when network resources are relatively scarce—generally when the share of each connection is less than 10 packets per round-trip time.
- Class-based isolation can allow a service provider to more accurately predict performance and thus provide reliable guarantees to users, including throughput/delay guarantees to short interactive transfers.

### 4.2 Effect of Pipe Size

Figure 2 shows the topology used for this set of experiments. The link between nodes 0 and 1 is the bottleneck link. Each pair of sender $S_i$ and receiver $R_i$ sets up a session in which infinite amount of data is sent, however, the session data is spread over several connections. Each connection has a limited size and classified as either long if its size is 1000 packets or short if its size is 10 packets. Each session starts at a randomly chosen time in the first 5 seconds of simulation time. Whenever a connection finishes its transmission, a new connection belonging to the same session starts. A total of six low-bandwidth telnet sessions (using tcplib in *ns* [36]) are used as background traffic to avoid deterministic (or synchronization) behavior. All the topology and protocol related parameters are listed in Table 2.

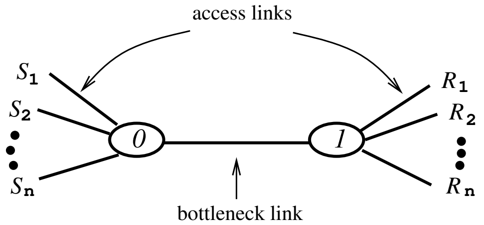



**Figure 2.** Topology to study effects of pipe and buffer size

We use $k_c$ and $k_b$ as scaling factors to control the size of the network pipe. From the simulation parameters, we can

**Table 2. Simulation parameters for studying effects of pipe and buffer size**

| Description | Value |
|---|---|
| Packet size | 576 bytes |
| Maximum TCP window size | 128 packets |
| TCP timeout granularity | 0.5 seconds |
| Bottleneck link delay | 40 ms |
| Access link delay | 5 ms |
| Capacity unit $C_u$ | 2.5 Mbps = 542 packets/s |
| Bottleneck capacity $C$ | $k_c \times C_u$ |
| Buffer unit $B_u$ | 50 packets $\approx C_u \times RTT$ |
| Bottleneck link buffer $B$ | $k_b \times B_u$ |
| Access link capacity | $4 \times C_u$ |
| Access link buffer | 4000 packets |
| Long connection size | 1000 packets |
| Short connection size | 10 packets |

roughly compute the size of the network pipe and the average share that a connection should get from the network. We use the two-way propagation delay of 100 mseconds as an estimate of the round-trip time RTT. Therefore, the network pipe size, $P$, is computed as:

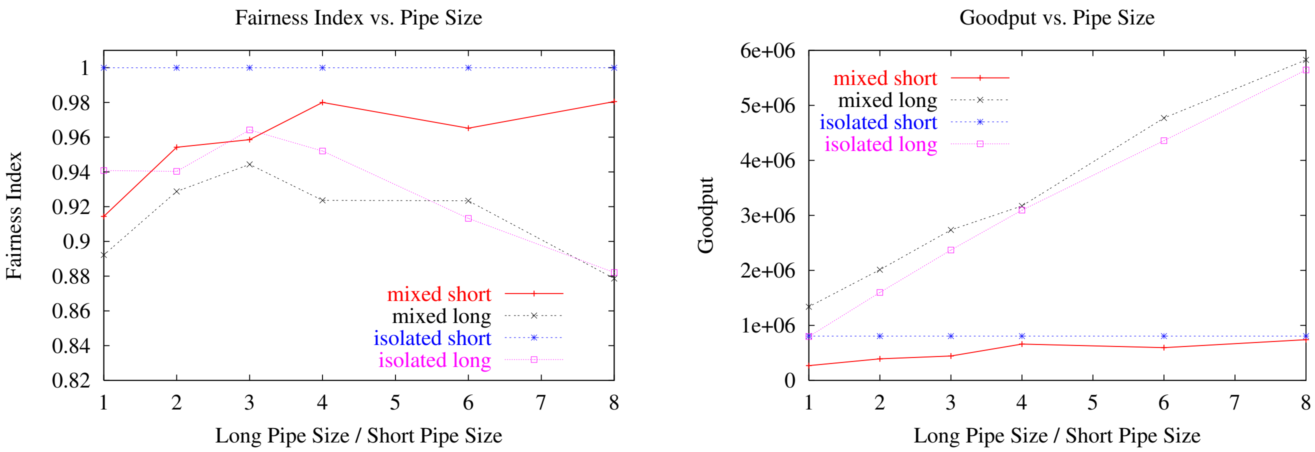
$$\begin{aligned} P &= C \times RTT + B \\ &= k_c \times C_u \times RTT + k_b \times B_u \end{aligned}$$

Assume there are $N$ concurrent connections, then each connection $i$ should be able to transmit at rate $r_i = \frac{P}{N}$ per RTT. In this section, we set $k_c = k_b = k$. If $k = 2$ and $N = 20$, we can compute $P$ to be 208 packets and $r_i$ = 10.4 packets/RTT.

We vary the value of $N$ and $k$ to study per-class fairness under different network congestion conditions for both isolation and sharing. In the *isolation case*, we assume class-based queueing is employed at the bottleneck link to essentially create two separate pipes: the first pipe, allocated to $N$ sessions of short TCP flows, has one queue with $k = 1$ (and hence capacity of $C_u$ and buffer space of $B_u$). The second pipe, allocated to $N$ sessions of long TCP flows, has another queue with $k$ ranging from 1 to 8 (and hence capacity ranging from $C_u$ to $8C_u$ and corresponding buffer space ranging from $B_u$ to $8B_u$). Each queue (pipe) is also used by three of the six telnet sessions. In the *sharing case*, all $2N$ sessions, in addition to the six telnet background sessions, share a common queue (pipe) with combined capacity ranging from $2C_u$ to $9C_u$ and corresponding buffer space ranging from $2B_u$ to $9B_u$.

#### 4.2.1 Large Pipe Case

In these experiments, we set $N = 30$, *i.e.* 30 sessions of long flows and 30 sessions of short flows. In this case, the size of the pipe allocated to short flows is 104 packets, thus each short connection can roughly transmit at a rate




(a) (b)

**Figure 3.** Large pipe case ($N = 30$). Performance versus long-to-short pipe size ratio ($k$): (a) Fairness index, and (b) Goodput

of 104/30=3.5 packets/RTT. Since to transmit 10 packets, at least 4 round-trip times are needed, the maximum rate at which a short connection can transmit is 2.5 packets/RTT.[5] Thus, this setup provides enough resources to short connections to transmit at the maximum rate. Figure 3(a) shows the fairness index for isolated short flows, isolated long flows, and for flows in the sharing (mixed) case. Figure 3(b) shows the goodput for each class of connections.

In the isolation case, since enough bandwidth is allocated to short connections that would allow them to transmit at maximum rate, short connections do not suffer any packet loss. Thus the fairness index for short connections is very close to 1. On the other hand, long connections start to experience packet loss once the window size of a connection approaches its fair share of 3.5 $k$ (unless 3.5 $k >$ the maximum window size of 128), resulting in unfairness among these long connections (consistent with [16]).

In the sharing case, when short and long connections are multiplexed in a common queue, short connections experience packet losses. Because long connections send larger windows (packet bursts), they can grab more bandwidth than their fair share (c.f. Figure 3(b)). Some of the short connections are also forced into timeout (due to insufficient number of duplicate ACKs to fast retransmit), resulting in unfairness among short connections as well. Therefore, with isolation in a well-engineered class-based network, each class of flows can enjoy a more friendly (fair) transmission environment without sacrificing the overall goodput and utilization of the network.

[5] Recall that the sending window of a short TCP connection, operating in slow-start phase, doubles every round-trip time, i.e. takes the values $1, 2, 4, 8, \cdots$. Thus, to send 10 packets, in the absence of packet losses, a TCP connection can send 1, 2, 4, and finally 3 packets over 4 round-trip times and finish its transmission.

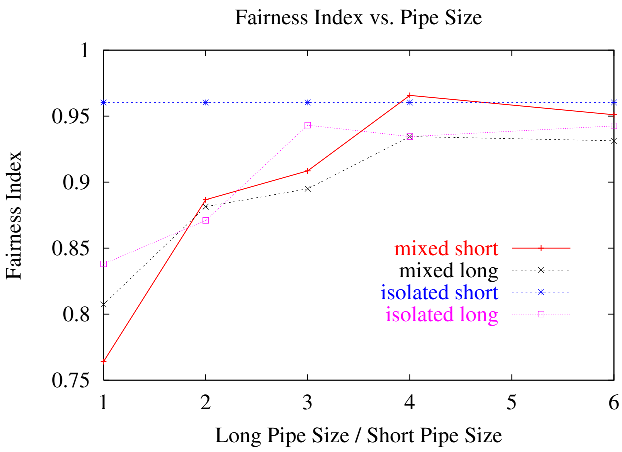
#### 4.2.2 Small Pipe Case

In this set of experiments, we set $N = 60$, *i.e.* 60 sessions of long flows and 60 sessions of short flows. The results are shown in Figure 4. In this case, the share of each short connection becomes 1.75 packets/RTT. This results in short connections experiencing packet losses and timing out. Thus fairness among short connections is lower than in the previous large pipe case.

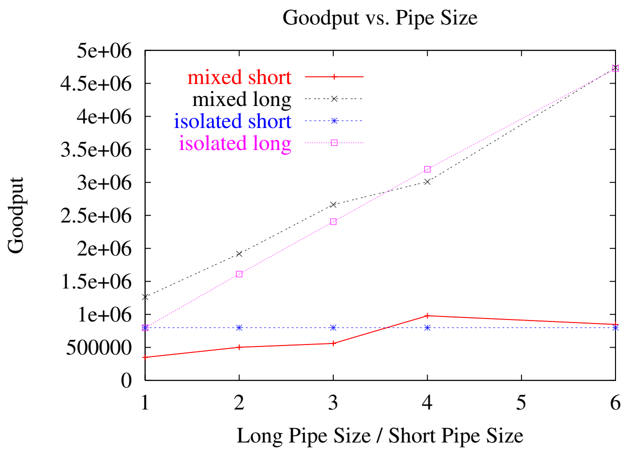
However, with isolation, long connections do not impede short connections, resulting in relatively high fairness (around 96%). On the contrary, when short and long connections are mixed, short connections suffer from severe unfairness ($f < 0.8$) when the network pipe is small (total capacity of $2C_u$). For long connections, isolation also results in improved fairness, yet less significant than that for short connections. Observe that isolation provides an overall goodput that is comparable to that obtained by sharing. More importantly, from Figure 3(b) and Figure 4(b), we observe a (almost) perfectly linear relationship between goodput and the amount of resources allocated to long connections. This makes it possible for a service provider to reliably predict performance for each class of connections, which is hard to do when resources are completely shared.

### 4.3 Effect of Buffer Size

Buffer space provisioning is an important factor in TCP's performance [24, 25]. Using the same setup as in Section 4.2, we investigate the effect of buffer space on isolation and sharing. We fix $N$ at 30, *i.e.* 30 sessions of long flows and 30 sessions of short flows. Under isolation, we set $k_c = 1$ for the short class pipe and $k_c = 4$ for the long class pipe. Thus, $k_c = 5$ in the sharing case, i.e. a total capacity of $5C_u$. We control the ratio of the buffer space




(a)




(b)

**Figure 4.** Small pipe case ($N = 60$). Performance versus long-to-short pipe size ratio ($k$): (a) Fairness index, and (b) Goodput

relative to the capacity of the pipe by varying the value of $k_b$ from $0.2k_c$ to $3k_c$. Figure 5 shows the results.

In the sharing (mixed) case, as the buffer size increases, long connections are allowed to send bigger windows (packet bursts), thus they grab more bandwidth from short connections, and more short connections are forced to timeout. As a result, the overall goodput for short connections decreases. A side effect of this unfair treatment to short connections is an improvement in fairness among them! (They all start to equally see worse performance, consistent with [5, 1].) However, even with such a high "fairness" value for short connections, it's very hard to predict the goodput of short connections, contrary to the isolation scenario where goodput is not affected very much by varying buffer space.

To summarize, when network resources are relatively scarce—generally when the share of each connection is less than 10 packets per round-trip time—isolation provides a more fair and more predictive service, as only "similar" connections (of the same class) compete for their allocated resources.

### 4.4 In The Presence of Web Background Traffic

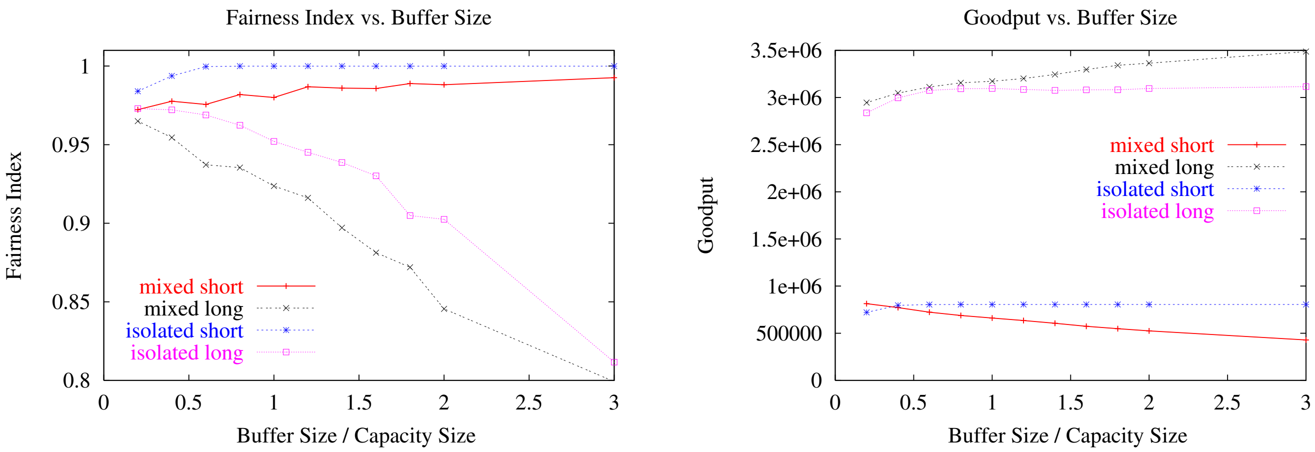
In this set of experiments, we compare the three traffic management schemes, described in Section 3.1, distributing flows over two parallel paths (cf. Figure 1), in the presence of Web background traffic on the topology shown in Figure 6.

This model is similar to the "FLEXBELL" model used in [13]. Here, node 0 connects to the servers pool, while node 1 connects to the clients pool. There is a diamond topology between nodes 0 and 1, with two separate paths between nodes 4 and 6. This diamond topology represents an administrative domain where the edge node 4 is responsible for detecting and classifying flows. This diamond topology is also comprised of the bottleneck links/paths. We assume a Web session model, so a client from the clients pool sets up a *session* with a server. Within each session, the client can grab a *page*, which could contain several *objects*. For each object, a TCP connection is established for transmission. All parameters are given in Table 3. The thresholds used for size-based splitting and lifetime-based splitting schemes were chosen so as to roughly divide the overall traffic equally among the two separate paths.

Both foreground TCP connections and the Web background connections are routed according to the routing scheme employed at node 4. Performance measures are collected for the foreground TCP connections, and results are shown in Figure 7.

Results are also shown when RED shared queues are deployed in the routers (with a load-balanced strategy at node 4). The two isolation schemes, namely size-based and lifetime-based splitting, can achieve fairness comparable to that of RED. However, with RED queues, fairness is achieved by sacrificing the goodput for short connections with smaller packet bursts (window sizes). On the contrary, with isolation (size-based splitting), short connections encounter less packet losses and achieve both high fairness and high goodput.

Without either RED or isolation, i.e. with tail-drop shared queues (with a load-balanced strategy at node 4), long connections can receive a much higher goodput. However, fairness among long connections is very low (as low as 20%) compared to those achieved by using RED shared queues or isolation (all above 80%). Low fairness values imply more variability in achieved goodput, which makes it very hard to predict the behavior of connections. In this sense, isolation provides an important engineering tool to service providers to more accurately predict performance and thus provide reliable guarantees to users. Furthermore,




(a) (b)

**Figure 5.** Performance versus buffer-to-capacity size ratio ($k_b/k_c$): (a) Fairness index, and (b) Goodput

unlike RED shared queues, by allocating enough resources to short interactive TCP transfers, they can be provided throughput/delay guarantees [23].

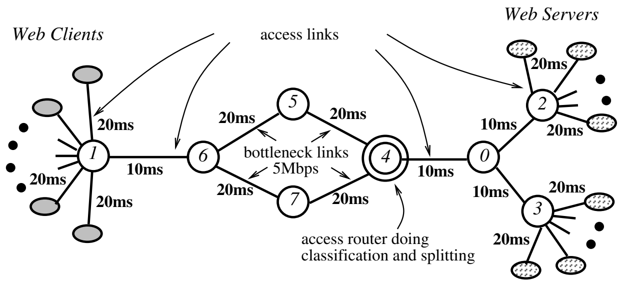



**Figure 6.** Topology with Web background traffic

## 5 Related Work

A number of studies (e.g. [9, 11]) examined isolation and sharing of network resources in the context of Intserv per-flow architectures. We concerned ourselves here with Diffserv-like architectures, where per-flow information is only maintained at access routers. In Section 2, we formulated discrete-time control models along the same lines as in [21]. However, Li and Nahrstedt in [21] do not address the benefits of isolation over sharing, particularly as related to fairness and stability of TCP flows.

A number of studies have proposed protocols that differentiate among flows based on their size or lifetime. For example, Shaikh *et al.* [29] investigate load-sensitive routing of only long-lived IP flows in order to improve routing stability and reduce overhead. However, the effect on

**Table 3. Simulation parameters for Web model**

| **Description** | **Value** |
|---|---|
| **Foreground Traffic** | |
| Number of long connections | 10 |
| Number of short connections | 10 |
| Long (foreground) connection size | 1000 packets |
| Short (foreground) connection size | 10 packets |
| **Web Background Traffic** | |
| Number of sessions | 200 |
| Number of connections per session | 1900 |
| Number of pages per connection | 1 |
| Number of objects per page | 1 |
| Interarrival of connections | exponential 2.4 seconds |
| Connection size distribution | Bounded Pareto [4:$2\times10^5$], skew parameter of 1.2 |
| **Network Parameters** | |
| Threshold for size-based split | 75 packets |
| Threshold for lifetime-based split | 32 packets |
| Packet size | 576 bytes |
| Maximum window size | 128 packets |
| Bottleneck link capacity ($C$) | 5 Mbps |
| Bottleneck link bandwidth$\times$delay | 238 packets |
| Bottleneck link buffer size | 250 packets |
| RED queue min_threshold | 20 packets |
| RED queue max_threshold | 245 packets |
| RED queue max drop rate | 0.2 % |
| Access link capacity | $\geq 3C$ |
| Access link buffer size | 4000 packets |

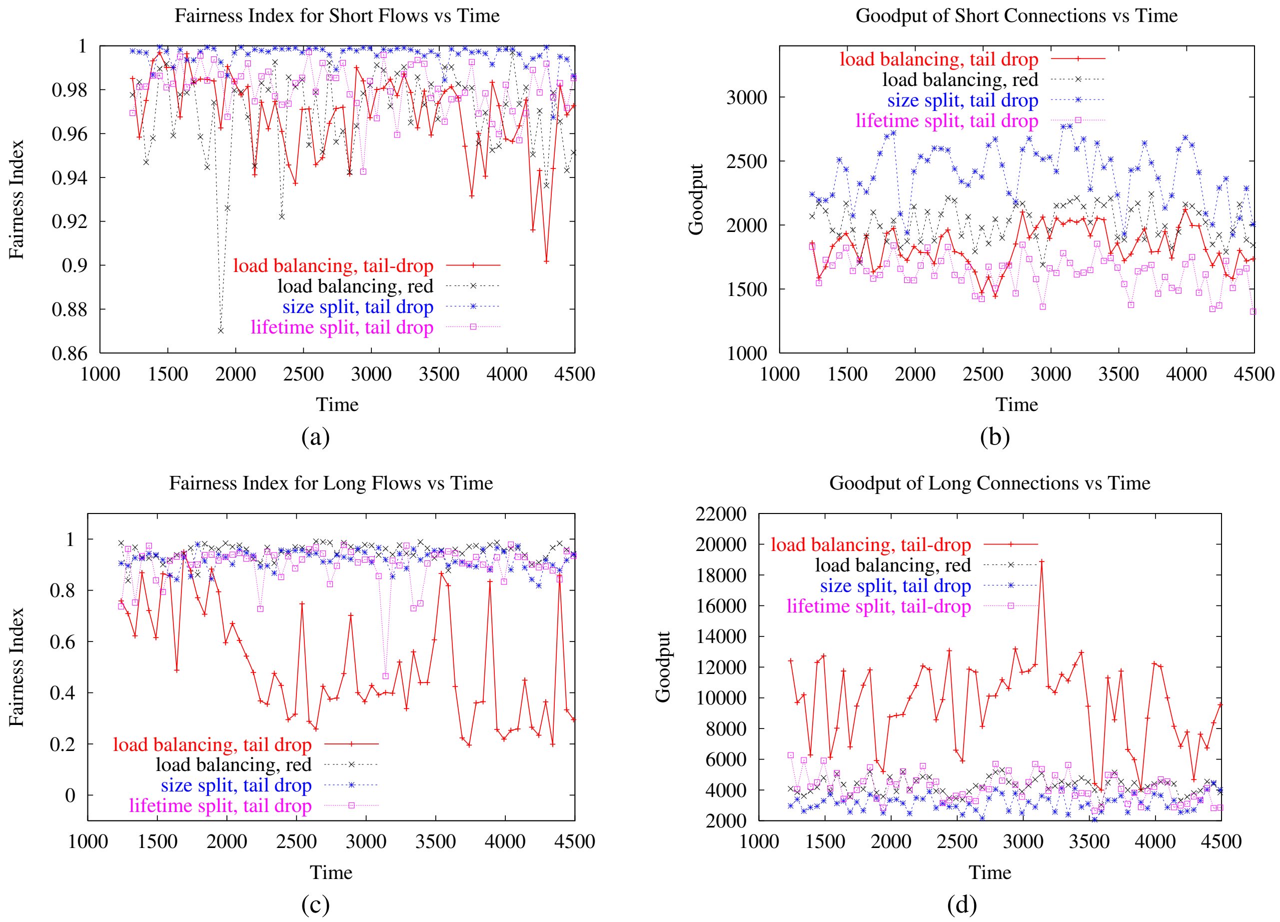



**Figure 7.** Performance with Web background traffic: (a) Fairness index of short connections, (b) Goodput of short connections, (c) Fairness index of long connections, and (d) Goodput of long connections

flow-controlled sources (such as TCP) has not been studied. Other studies have considered the cut-through switching of long-lived IP flows (e.g. using IP/Tag Switching) [35, 14]. These studies only consider the reduced overhead due to switching, but not the performance of flows.

Other studies have attempted to improve TCP fairness by either modifying TCP itself or by employing non-tail-drop buffer management at routers (e.g. [19, 18, 20, 16, 25, 30]). In particular, Morris [25] proposes solutions that require per-flow information at all routers. In this paper, we advocate the use of a (less costly) class-based solution. Bonald *et al.* [5] and Nandy *et al.* [26] have shown that the well-known RED buffer management policy may have a strong loss bias against smooth UDP (e.g. audio) flows. Seddigh *et al.* [12, 28] propose a packet dropping mechanism to improve fairness among UDP and TCP flows. Our focus here has been on using isolation to improve predictability of the service and fairness of TCP flows of different size.

Most TCP studies consider very long (or *infinite*) TCP connections, and focus on characterizing the steady-state transfer throughput. Only few recent TCP studies [7, 3] have started to investigate short flows, which comprise most of the current Internet flows [35]. However, to our knowledge, the interaction among short and long TCP flows has not been studied.

## 6 Conclusions and Future Work

Using control-theoretic arguments and extensive simulations, we have shown that service predictability and fairness can be much improved by isolating TCP flows based on their size. In this paper, we classified TCP flows into two classes: one of short flows and another of long flows. Each class is (logically) allocated separate resources (buffer space and capacity), for which size-homogeneous flows compete. Flow classification can be done only by an access (border) router, thus the complexity of implementing such a class-based solution is low. By appropriately allocating resources to each class, the many short (usually interactive/delay-sensitive) flows can not only enjoy better predictability and fairness, but also faster service. This is in sharp contrast to traditional shared systems that do not differentiate among TCP flows with different characteristics, such as burstiness of arrivals/departures and congestion/sending window dynamics.

Future work remains on how to dynamically allocate resources to various classes, how to determine the best threshold values used to classify flows into classes, and how to account for characteristics other than flow size (e.g. various round-trip times).

**Acknowledgment:** Thanks to Azer Bestavros for various discussions on aspects of this work.

## References


[1] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In *IEEE/INFOCOM'2000*, Tel-Aviv, Israel, March 2000.

[2] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering over MPLS. RFC 2702, Network Working Group, September 1999.

[3] C. Barakat and E. Altman. Performance of Short TCP Transfers. In *Networking 2000*, Paris, May 2000.

[4] Y. Bernet and et al. A Framework for Differentiated Services. Internet Draft draft-ietf-diffserv-framework-02.txt, IETF, Feburary 1999.

[5] T. Bonald, M. May, and J-C. Bolot. Analytic Evaluation of RED Performance. In *IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 2000.

[6] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A Framework for MPLS. IETF Draft, Network Working Group, September 1999.

[7] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP Latency. In *IEEE/INFOCOM'2000*, Tel-Aviv, Israel, March 2000.

[8] D.M. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, pages 1–14, 1989.

[9] D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *Proceedings of ACM SIGCOMM '92*, pages 14–26, August 1992.

[10] D.D. Clark and W. Fang. Explicit Allocation of Best-Effort Packet Delivery Service. *IEEE/ACM Transactions on Networking*, pages 362–373, August 1998.

[11] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *ACM SIGCOMM '89*, pages 1–12, Austin, Texas, September 1989.

[12] W. Fang, N. Seddigh, and B. Nandy. A Time Sliding Window Three Colour Marker (TSWTCM). Internet Draft, draft-fang-diffserv-tc-tswtcm-00.txt, October 1999.

[13] A. Feldmann, A.C. Gilbert, P. Huang, and W. Willinger. Dynamics of IP traffic: A Study of the Role of Variability and the Impact of Control. In *Proceeding of ACM/SIGCOMM'99*, Boston, Mass, September 1999.

[14] A. Feldmann, J. Rexford, and R. Caceres. Efficient Policies for Carrying Web Traffic over Flow-Switched Networks. *IEEE/ACM Transactions on Networking*, pages 673–685, December 1998.

[15] W. Feng, D. Kandlur, D. Saha, and K. Shin. Understanding TCP Dynamics in an Integrated Services Internet. In *NOSSDAV '97*, 1997.

[16] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, pages 397–413, August 1993.

[17] S. Floyd and V. Jacobson. Link-sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, pages 365–386, August 1995.

[18] F.P.Kelly, A.K.Maullo, and D.K.H. Tan. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 1998.

[19] The PSC Networking group. The TCP-Friendly Website. http://www.psc.edu/networking/tcp_friendly.html.

[20] P. Hurley, J.L. Boudec, and P. Thiran. A Note on the Fairness of Additive Increase and Multiplicative Decrease. In *16th International Teletraffic Congress (ITC-16)*, Edinburgh, Scotland, June 1999.

[21] B. Li and K. Nahrstedt. A Control Theoretical Model for Quality of Service Adaptations. In *IEEE International Workshop on Quality of Service (IWQoS 98)*, Napa, California, May 1998.

[22] Ibrahim Matta and Azer Bestavros. QoS Controllers for the Internet. In *Proceedings of the NSF Workshop on Information Technology*, Cairo, Egypt, March 2000.

[23] Ibrahim Matta and Liang Guo. Differentiated Predictive Fair Service for TCP Flows. Technical Report BU-CS-2000-012, Boston University, Computer Science Department, Boston, MA 02215, May 2000.

[24] R.T. Morris. TCP Behavior with Many Flows. In *IEEE International Conference on Network Protocols (ICNP'97)*, Atlanta, Georgia, October 1997.

[25] R.T. Morris. Scalable TCP Congestion Control. In *IEEE/INFOCOM'2000*, Tel-Aviv, Israel, March 2000.

[26] B. Nandy, N. Seddigh, and P. Pieda. Diffserv's Assured Forwarding PHB: What Assurance does the Customer Have? In *Proceedings of NOSSDAV'99*, July 1999.

[27] K. Ogata. *Discrete-Time Control Systems*. Prentice-Hall, Inc., 1987.

[28] N. Seddigh, B. Nandy, and P. Pieda. Study of TCP and UDP Interaction for the AF PHB. Internet Draft, draft-nsbnpp-diffserv-tcpudpaf-01.pdf, September 1999.

[29] A. Shaikh, J. Rexford, and K.G. Shin. Load-Sensitive Routing of Long-Lived IP Flows. In *SIGCOMM'99*, Boston, MA, September 1999.

[30] S. Shenker. A Theoretical Analysis of Feedback Flow Control. In *ACM/SIGCOMM'90*, 1990.

[31] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. Internet RFC 2212, Sepetember 1997.

[32] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. Internet RFC 2001, January 1997.

[33] W. Stevens. *TCP/IP Illustrated, Vol. 1: The Protocols*. Addison-Wesley, 1997.

[34] I. Stoica and H .Zhang. LIRA: An Approach for Service Differentiation in the Internet. In *NOSSDAV'98*, London, UK, July 1998.

[35] K. Thompson, G.J. Miller, and R. Wilder. Wide-Area Internet Traffic Patterns and Characteristics. *IEEE/ACM Transactions on Networking*, pages 10–23, November 1997.

[36] UCB, LBNL, and VINT. Network simulator - ns (version 2). http://www-mash.cs.berkeley.edu/ns/, 1999.