TCP-LP: Low-Priority Service via End-Point Congestion Control

Aleksandar Kuzmanovic and Edward W. Knightly

Abstract—**Service prioritization among different traffic classes is an important goal for the Internet. Conventional approaches to solving this problem consider the existing best-effort class as the low-priority class, and attempt to develop mechanisms that provide "better-than-best-effort" service. In this paper, we explore the opposite approach, and devise a new distributed algorithm to realize a low-priority service (as compared to the existing best effort) from the network endpoints. To this end, we develop TCP Low Priority (TCP-LP), a distributed algorithm whose goal is to utilize only the excess network bandwidth as compared to the "fair share" of bandwidth as targeted by TCP. The key mechanisms unique to TCP-LP congestion control are the use of one-way packet delays for early congestion indications and a TCP-transparent congestion avoidance policy. The results of our simulation and Internet experiments show that that: (1) TCP-LP is largely non-intrusive to TCP traffic; (2) both single and aggregate TCP-LP flows are able to successfully utilize excess network bandwidth; moreover, multiple TCP-LP flows share excess bandwidth fairly; (3) substantial amounts of excess bandwidth are available to the low-priority class, even in the presence of "greedy" TCP flows; (4) the response times of web connections in the best-effort class decrease by up to 90% when long-lived bulk data transfers use TCP-LP rather than TCP; (5) despite their low-priority nature, TCP-LP flows are able to utilize significant amounts of available bandwidth in a wide-area network environment.**

Keywords— **TCP-LP, TCP, available bandwidth, service prioritization, TCP-transparency.**

I. Introduction

MOTIVATED by the diversity of networked applications, a significant effort has been made to provide differentiation mechanisms in the Internet, e.g., [7]. However, despite the availability of simple and scalable solutions (e.g., [5]), deployment has not been forthcoming. A key reason is the heterogeneity of the Internet itself: with vastly different link capacities, congestion levels, etc., a single mechanism is unlikely to be uniformly applicable to all network elements.

In this paper, we devise TCP-LP (Low Priority), an end-point protocol that achieves two-class service prioritization without any support from the network. The key observation is that end-to-end differentiation can be achieved by having different end-host applications employ different congestion control algorithms as dictated by their performance objectives. Since TCP is the dominant protocol for best-effort traffic, we design TCP-LP to realize a low-priority service as compared to the existing best effort service. Namely, the objective is for TCP-LP flows to utilize the bandwidth left unused by TCP flows in a non-intrusive, or TCP-transparent, fashion. Moreover, TCP-LP is a distributed algorithm that is realized as a sender-side modification of the TCP protocol.

One class of applications of TCP-LP is low-priority file transfer over the Internet. For network clients on low-speed access links, TCP-LP provides a mechanism to retain faster response times for interactive applications using TCP, while simultaneously making progress on background file transfers using TCP-LP. Similarly, in enterprise networks, TCP-LP enables large file backups to proceed without impeding interactive applications, a functionality that would otherwise require a multi-priority or separate network. Finally, institutions often rate-limit certain applications (e.g., peer-to-peer file sharing applications) such that they do not degrade the performance of other applications. In contrast, TCP-LP allows low priority applications to use all excess capacity while also remaining transparent to TCP flows.

A second class of applications of TCP-LP is inference of available bandwidth for network monitoring, end-point admission control [4], and performance optimization (e.g., to select a mirror server with the highest available bandwidth). Current techniques (e.g., [20], [1], [13]) estimate available bandwidth by making statistical inferences on measurements of the delay or loss characteristics of a sequence of transmitted probe packets. In contrast, TCP-LP is algorithmic with the goal of transmitting at the rate of the available bandwidth. Consequently, competing TCP-LP flows obtain their *fair share* of the available bandwidth, as opposed to probing flows which infer the *total* available bandwidth, overestimating the fraction actually available individually when many flows are simultaneously probing. Moreover, as the available bandwidth changes over time, TCP-LP provides a mechanism to continuously adapt to changing network conditions.

Our methodology for developing TCP-LP is as follows. First, we develop a reference model to formalize the two design objectives: TCP-LP transparency to TCP, and (TCP-like) fairness among multiple TCP-LP flows competing to share the excess bandwidth. The reference model consists of a two level hierarchical scheduler in which the first level provides TCP packets with strict priority over TCP-LP packets and the second level provides fairness among microflows within each class. TCP-LP aims to achieve this behavior in networks with non-differentiated (first-come-first-serve) service.

Next, to approximate the reference model from a distributed end-point protocol, TCP-LP employs two new mechanisms. First, in order to provide TCP-transparent low-priority service, TCP-LP flows must detect oncoming congestion prior to TCP flows. Consequently, TCP-LP uses inferences of one-way packet delays as early indications of network congestion rather than packet losses as used by TCP. We develop a simple analytical model to show that due to the non-linear relationship between throughput and round-trip time, TCP-LP can maintain TCP-transparency even if TCP-LP flows have larger round-trip times than TCP flows. Moreover, a desirable consequence of early congestion inferences via *one-way* delay measurements is that they detect congestion only on the forward path (from the

A. Kuzmanovic is with the EECS Department at Northwestern University. E. Knightly is with the ECE/CS Departments at Rice University.

This research is supported by the National Science Foundation.

A subset of this work appears in the Proceedings of IEEE Infocom '03 [16].

source to the destination) and prevent false early congestion indications from reverse cross-traffic.

TCP-LP's second mechanism is a novel congestion avoidance policy with three objectives: (1) quickly back off in the presence of congestion from TCP flows, (2) quickly utilize the available excess bandwidth in the absence of sufficient TCP traffic, and (3) achieve fairness among TCP-LP flows. To achieve these objectives, TCP-LP's congestion avoidance policy modifies the additive-increase multiplicative-decrease policy of TCP via the addition of an inference phase and use of a modified back-off policy.

Furthermore, we perform an extensive set of *ns-2* simulation experiments and study TCP-LP's characteristics in a variety of scenarios (single and multiple bottlenecks, short- and long-lived TCP flows, etc.). First, in our experiments with greedy TCP flows (FTP downloads), we show that TCP-LP is largely non-intrusive to TCP traffic, and that TCP flows achieve approximately the same throughput whether or not TCP-LP flows are present. Second, we explore TCP-LP's dynamic behavior using experiments with artificial "square-wave" background traffic. We show that single and aggregate TCP-LP flows can successfully track and utilize the excess network bandwidth. Finally, in our experiments with HTTP background traffic, we show that flows in the best-effort class can benefit significantly from the two-class service prioritization scheme. For example, the response times of web connections in the best-effort class decrease by up to 90% when long-lived bulk data transfers use TCP-LP rather than TCP.

Finally, we implement TCP-LP in Linux and evaluate it both in a testbed as well as on the Internet. In the testbed, we perform experiments with many TCP and TCP-LP flows and show that TCP-LP remains its TCP-transparent property even in such large-aggregation regimes. Likewise, our Internet experiments show that TCP-LP remains non-intrusive in a wide-area network environment, while being able to utilize substantial amounts of the available spare network bandwidth. For example, when compared to TCP, TCP-LP is able to utilize approximately 45% of the TCP throughput on average during working-hours (8 a.m. to 5 p.m.), and as much as 75% outside this interval. Thus, the results from both our simulation and Internet experiments confirm that TCP-LP is a practically applicable protocol that accurately approximates the functionality of the reference model.

The reminder of this paper is organized as follows. In Section II, we present the reference model to describe TCP-LP's design objectives and in Section III we present the TCP-LP protocol. Sections IV and V present simulation preliminaries and experimental results. In Section VI we present protocol implementation details and results from the real-network experiments. Finally, in Sections VII and VIII we discuss related work and conclude.

II. Problem Formulation

In this section, we provide a brief review of TCP congestion control and present a reference model to describe TCP-LP's design objectives.

A. TCP Congestion Control

Figure 1 shows a temporal view of the TCP/Reno congestion window behavior at different stages with points on the top indicating packet losses.[1] Data transfer begins with the *slow-start* phase in which TCP increases its sending rate exponentially until it encounters the first loss or maximum window size. From this point on, TCP enters the *congestion-avoidance* phase and uses an additive-increase multiplicative-decrease policy to adapt to congestion. Losses are detected via either time-out from non-receipt of an acknowledgment, or by receipt of a triple-duplicate acknowledgment. If loss occurs and less than three duplicate ACKs are received, TCP reduces its congestion window to one segment and waits for a period of retransmission time out (RTO), after which the packet is resent. In the case that another time out occurs before successfully retransmitting the packet, TCP enters the *exponential-backoff* phase and doubles RTO until the packet is successfully acknowledged.

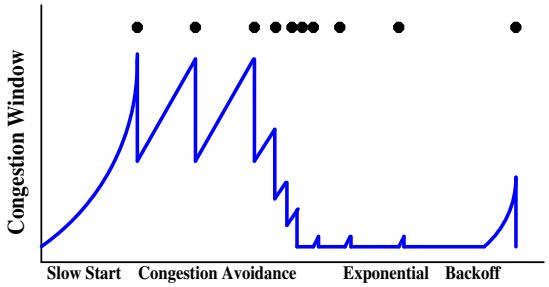


Fig. 1. Behavior of TCP Congestion Control

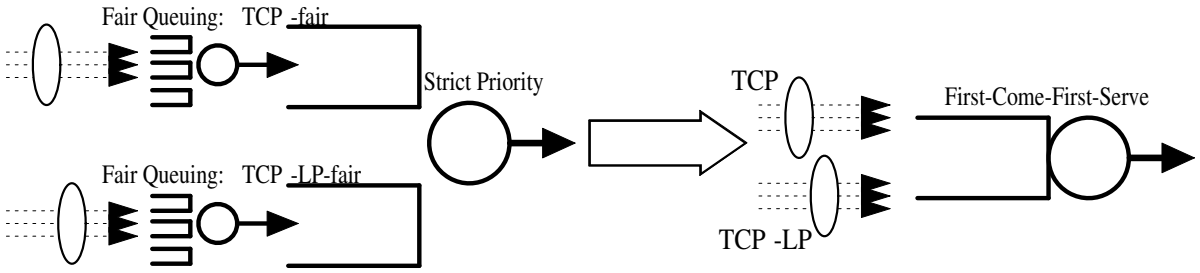
One objective of TCP congestion control is for each flow to transmit at its fair rate at its bottleneck link. While biasing rates in favor of flows with small round-trip times, we none-the-less refer to TCP as "fair" in the discussion below.[2]

B. Reference Model and Design Objectives

The objective of TCP-LP is to use excess network bandwidth left unutilized by non TCP-LP flows thereby making TCP-LP flows transparent to TCP and UDP flows. This design objective is formalized in Figure 2(a) which depicts a two-class hierarchical scheduling model (see [10]) that achieves the idealized system functionality. In the reference system, there is a high-priority and low-priority class, with the former obtaining strict priority service over the latter. Within each class, service is fair among competing flow-controlled flows. As networks do not typically employ such scheduling mechanisms, the objective of TCP-LP is to obtain an approximation to the reference model's behavior via an end-point congestion control algorithm. As depicted in Figure 2(b), in the actual system, all flows (high and low priority) are multiplexed into a single first-come-first-serve queue and service approximating that of the reference model is obtained via the use of two different congestion control protocols, TCP and TCP-LP. In other words, TCP flows should obtain strict priority service over TCP-LP flows, and competing TCP-LP flows should each obtain a fair bandwidth share compared to

[1] A detailed description of TCP can be found in [15].

[2] TCP's fairness properties are studied in depth in [23] for example.



(a) Reference Model (b) TCP-LP Realization

Fig. 2. Reference Model and TCP-LP Realization

other TCP-LP flows.[3]

To further illustrate, consider again the system shown in Figure 2(b). Denote C as the link capacity, D as the aggregate rate demanded by all non-TCP-LP flows (high priority), and n as the number of TCP-LP flows in the system, with all TCP-LP flows having infinite demand and identical round trip times. Since the excess network bandwidth is $(C-D)^+$, the goal is for each TCP-LP flow to utilize bandwidth given by $(C-D)^+/n$.

III. TCP-LP Protocol: Mechanisms and Deployment

In this section we develop TCP-LP, a low-priority congestion control protocol that uses the excess bandwidth on an end-to-end path, versus the fair-rate utilized by TCP. We first devise a mechanism for early congestion indication via inferences of one-way packet delays. Next, we present TCP-LP's congestion avoidance policy to exploit available bandwidth while being sensitive to early congestion indicators. We then develop a simple queueing model to study the feasibility of TCP-transparent congestion control under heterogeneous round trip times. Finally, we provide guidelines for TCP-LP parameter settings.

A. Early Congestion Indication

To achieve low priority service in the presence of TCP traffic, it is necessary for TCP-LP to infer congestion earlier than TCP. In principle, the network could provide such early congestion indicators. For example, TCP-LP flows could use a type-of-service bit to indicate low priority, and routers could use Early Congestion Notification (ECN) messages [19] to inform TCP-LP flows of lesser congestion levels than TCP flows. However, given the absence of such network support, we devise an end-point realization of this functionality by using packet delays as early indicators for TCP-LP, as compared to packet drops used by TCP. In this way, TCP-LP and TCP implicitly coordinate in a distributed manner to provide the desired priority levels.

A.1 Delay Threshold

TCP-LP measures one-way packet delays and employs a simple delay threshold-based method for early inference of congestion. Denote d_i as the one-way delay of the packet with sequence number i, and d_{min} and d_{max} as the minimum and maximum one-way packet delays experienced throughout the connection's lifetime.[4] Thus, d_{min} is an estimate of the one-way propagation delay and $d_{max}-d_{min}$ is an estimate of the maximum queueing delay.

Next, denote γ as the delay smoothing parameter, and sd_i as the smoothed one-way delay. An exponentially weighted moving average is computed as

$$sd_i = (1-\gamma)sd_{i-1} + \gamma d_i. \tag{1}$$

An early indication of congestion is inferred by a TCP-LP flow whenever the smoothed one-way delay exceeds a threshold within the range of the minimum and maximum delay. In other words, the early congestion indication condition is

$$sd_i > d_{min} + (d_{max} - d_{min})\delta. \tag{2}$$

where $0 < \delta < 1$ denotes the threshold parameter (we discuss the setting of parameters δ and γ in detail in Section III-D). Thus, analogous to the way ECN uses increasing queue sizes to alert flows of congestion before loss occurs, the above scheme infers forthcoming congestion from the end points' delay measurements so that TCP-LP flows can be non-intrusive to TCP flows.

A.2 Delay Measurement

TCP-LP obtains samples of one-way packet delays using the TCP timestamp option [12]. Each TCP packet carries two four-byte timestamp fields. A TCP-LP sender timestamps one of these fields with its current clock value when it sends a data packet. On the other side, the receiver echoes back this timestamp value and in addition timestamps the ACK packet with its own current time. In this way, the TCP-LP sender measures one-way packet delays. Note that the sender and receiver clocks do not have to be synchronized since we are only interested in the relative time difference. Moreover, a drift between the two clocks is not significant here as resets of d_{min} and d_{max} on time-scales of minutes can be applied [18]. Finally, we note that by using *one-way* packet delay measurements instead of round-trip times, cross-traffic in the reverse direction does not influence TCP-LP's inference of early congestion.

[3] As UDP flows are non-responsive, they would also be considered high priority and multiplexed with the TCP flows.

[4] Minimum and maximum one-way packet delays are initially estimated during the slow-start phase and are used after the first packet loss, i.e., in the congestion avoidance phase.

B. Congestion Avoidance Policy

B.1 Objectives

TCP-LP is an end-point algorithm that aims to emulate the functionality of the reference-scheduling model depicted in Figure 2. Consider for simplicity a scenario with one TCP-LP and one TCP flow. The reference strict priority scheduler serves TCP-LP packets only when there are no TCP packets in the system. However, whenever TCP packets arrive, the scheduler immediately begins service of higher priority TCP packets.

Similarly, after serving the last packet from the TCP class, the strict priority scheduler immediately starts serving TCP-LP packets. Note that it is impossible to exactly achieve this behavior from the network endpoints as TCP-LP operates on time-scales of round-trip times, while the reference scheduling model operates on time-scales of packet transmission times. Thus, our goal is to develop a congestion control policy that is able to *approximate* the desired dynamic behavior.

B.2 Reacting to Early Congestion Indicators

TCP-LP must react quickly to early congestion indicators to achieve TCP-transparency. However, simply decreasing the congestion window promptly to zero packets after the receipt of an early congestion indication (as implied by the reference scheduling model) unnecessarily inhibits the throughput of TCP-LP flows. This is because a single early congestion indication cannot be considered as a reliable indication of network congestion given the complex dynamics of cross traffic. On the other hand, halving the congestion window of TCP-LP flows upon the congestion indication, as recommended for ECN flows [8], would result in too slow a response to achieve TCP transparency.

To compromise between the two extremes, TCP-LP employs the following algorithm. After receipt of the initial early congestion indication, TCP-LP halves its congestion window and enters an *inference phase* by starting an *inference time-out timer*. During this inference period, TCP-LP only observes responses from the network, without increasing its congestion window. If it receives another early congestion indication before the inference timer expires, this indicates the activity of cross traffic, and TCP-LP decreases its congestion window to one packet. Thus, with persistent congestion, it takes two round-trip times for a TCP-LP flow to decrease its window to 1. Otherwise, after expiration of the inference timer, TCP-LP enters the additive-increase congestion avoidance phase and increases its congestion window by one per round-trip time (as with TCP flows in this phase).

We observe that as with router-assisted early congestion indication [8], consecutive packets from the same flow often experience similar network congestion state. Consequently, as suggested for ECN flows, TCP-LP also reacts to a congestion indication event at most once per round-trip time. Thus, in order to prevent TCP-LP from over-reacting to bursts of congestion indicated packets, TCP-LP ignores succeeding congestion indications if the source has reacted to a previous delay-based congestion indication or to a dropped packet in the last round-trip time.

Finally, the minimum congestion window for TCP-LP flows in the inference phase is set to 1. In this way, TCP-LP flows conservatively ensure that an excess bandwidth of at least one packet per round-trip time is available before probing for additional bandwidth.

B.3 Pseudo Code

```
Variables
  new-ACK:  indication that ACK packet has arrived
  cong_ind:   congestion indication
  itti:         inference time-out timer indication
  cwnd:       congestion window
Pseudocode
1.    if (new_ACK == 1)
2.      if (cong_ind == 1)
3.        if (itti == 1)
4.          cwnd = 1;
5.        else
6.          cwnd = cwnd/2;
7.        endif
8.        itt = 1;
9.      else
10.       if (itti != 1)
11.         cwnd += 1/cwnd;
12.       endif
13.     endif
14.   endif
```

Fig. 3. TCP-LP Congestion Avoidance Policy

Figure 3 shows the pseudo code for TCP-LP's congestion avoidance policy. We denote *cwnd* as congestion window size and *itti* as the inference time-out timer state indicator. It is set to one when the timer is initiated and to zero when the timer expires. Further, Figure 4 illustrates a schematic view of TCP-LP's congestion window behavior at different stages, where points on the top mark early congestion indications and the inference timer period is labeled *itt*. For example, with the first early congestion indicator, this flow enters the inference phase. It later successfully exits the inference phase into additive increase as no further early congestion indicators occur. On the other hand, the second early congestion indicator is followed by a second indicator within the inference phase such that the congestion window is subsequently set to one.

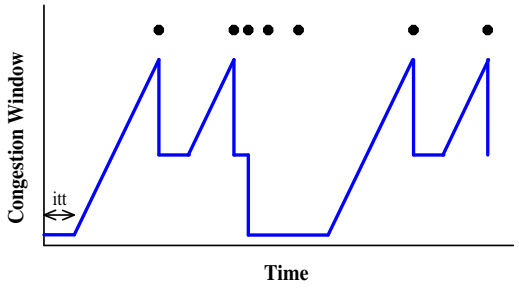


Fig. 4. Behavior of TCP-LP Congestion Avoidance Phase

B.4 Preserving TCP-Transparency in Large Aggregation Regimes

A key goal of TCP-LP is to achieve non-intrusiveness to TCP flows. Thus, as explained above, TCP-LP reduces its window size to one packet per RTT in the presence of TCP flows. However, in scenarios with many TCP-LP flows, it becomes increasingly possible for TCP-LP aggregates to impact TCP flows. For example, consider a scenario with a hundred TCP-LP flows competing with TCP flows on a 10 Mb/s link. If the round-trip time of the TCP-LP flows is 100 ms and the packet size is 1500 Bytes, then this TCP-LP aggregate utilizes 12% of the bandwidth, despite the fact that each flow sends only a single packet per RTT.[5] To mitigate this problem, TCP-LP decreases the packet size to 64 Bytes whenever the window size drops below 5 packets. In this way, TCP-LP significantly decreases its impact on TCP flows in high-aggregation regimes, yet it is still able to quickly react (after RTT) to changes in congestion. In the above example, a hundred TCP-LP flows would then utilize only 0.5% of the bandwidth in the presence of TCP flows.

C. Modeling TCP and TCP-LP Interactions

As described above, TCP-LP must detect congestion earlier than TCP. However, in a heterogeneous networking environment, different flows can have different round-trip times ranging from several *msec* to several *sec*. Here we address to what extent TCP-LP flows with large round-trip times can still infer congestion prior to TCP flows with smaller round-trip times. Such behavior is required such that TCP-LP flows with large round-trip times can still utilize excess network bandwidth without hindering TCP flows with small round-trip times.

Our approach is to develop a simple queueing model that characterizes TCP-LP's non-intrusiveness in the presence of TCP cross-traffic, and quantifies it with respect to the threshold parameter δ. The model, illustrated in Figure 5, consists of a bottleneck queue with capacity C driven by traffic from one TCP-LP connection with round-trip time rtt_l. Moreover, the queue services (high priority) TCP cross traffic with round-trip time denoted by rtt_h. For simplicity, the cross traffic is also modeled as originating from a single TCP connection.

Denoting the queue's total buffer space by Q, the early congestion indication condition is satisfied whenever the queue length is greater than $Q\delta$ packets, which is equivalent to condition (2) with $\gamma = 1$ in this idealistic scenario. Further consider that without congestion, the two flows are increasing their rates linearly with constants α_l and α_h packets per second respectively.[6]

In such a scenario and under a fluid flow model, we can quantify the conditions in which the TCP-LP flow will decrease its sending rate before the TCP cross-traffic will experience packet loss. We assume that the queue is initially empty and consider that the aggregate rate of the two flows is C at $t = 0$. Denote t_l and t_h as the respective times when the TCP-LP and TCP cross-traffic flow determine that the queue is congested. For TCP-LP,

[5]We disregard the effects of the exponential-backoff phase that may actually decrease this percentage.

[6]An increase in congestion window of α packets is considered to be equal to an increase in bandwidth of α packets per second.

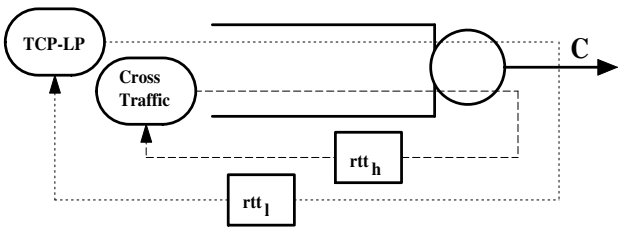


Fig. 5. Simplified Model of Heterogeneous RTT Effects

this time is given by the solution to

$$Q\delta = \int_0^{t_l} (C + (\alpha_l/rtt_l + \alpha_h/rtt_h)t - C)dt, \qquad (3)$$

so that $t_l = \sqrt{\frac{2Q\delta}{\alpha_l/rtt_l+\alpha_h/rtt_h}}$. Similarly, $t_h = \sqrt{\frac{2Q}{\alpha_l/rtt_l+\alpha_h/rtt_h}}$. In Equation (3), the term $C + (\alpha_l/rtt_l + \alpha_h/rtt_h)t$ denotes the instantaneous arrival rate of the two flows at time t, whereas C denotes the service rate. For the TCP-LP flow to decrease its rate before the cross traffic experiences packet loss, it is necessary that $t_l + rtt_l < t_h$, which is equivalent to

$$rtt_l < \sqrt{\frac{2Q}{\alpha_l/rtt_l + \alpha_h/rtt_h}}(1 - \sqrt{\delta}). \qquad (4)$$

To interpret this result, consider that $\alpha_l/rtt_l = n\alpha_h/rtt_h$. For $\alpha_l = \alpha_h$, this means that the TCP-LP flow's round-trip time is n times larger than the competing TCP flow's round-trip. In this case, the above condition is equivalent to

$$n(n+1) < \frac{\alpha_h}{\alpha_l^2 rtt_h} 2Q(1 - \sqrt{\delta})^2. \qquad (5)$$

Inequality (5) gives an upper bound on n as a function of the cross traffic's round-trip time rtt_h, the queue size Q (in packets) and the delay threshold δ. To interpret this result, consider a typical queue size of $Q = 2.5Crtt_h$ and increase parameters $\alpha_l = \alpha_h = 1$ packet/RTT. With the approximation that $n(n+1) \approx n^2$, we have that

$$n < \sqrt{5C}(1 - \sqrt{\delta}).$$

Figure 6 depicts the relationship between the ratios of the round-trip times n and the delay threshold δ for capacity $C = 1.5$ Mb/s and average packet size of 1 kB. Observe that TCP-LP's responsiveness rapidly decreases with increasing delay threshold δ. Moreover, the figure indicates TCP-LP's potential to achieve TCP transparency. For example, the point (0.4, 11.25) shows that with delay threshold $\delta = 0.4$, a single TCP-LP connection infers congestion before the competing TCP incurs loss, even if the TCP-LP flow's round-trip time is 11 times larger than that of the TCP flow. Similar conclusions can be drawn from Equation (5) for $rtt_l = rtt_h$ and $\alpha_l \neq \alpha_h$.

D. Guidelines for Parameter Settings

Here, we propose guidelines for setting TCP-LP's parameters given that the receipt of a single packet whose smoothed one-way delay is greater than a prespecified threshold serves as an early notification of congestion to a TCP-LP flow.

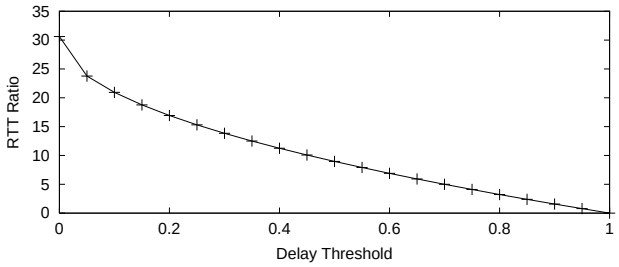


Fig. 6. Relationship between the RTT Ratio n and Threshold δ

D.1 Delay Smoothing γ

First, we consider the delay smoothing parameter γ of Equation (1). With large variations in network delay due to bursty cross traffic, smoothing one-way packet delays is essential for preventing false early congestion indications. On the other hand smoothing over excessively long time intervals (corresponding to small vales for γ) can substantially degrade TCP-LP's ability to detect congestion in its early stages. To balance these two requirements, TCP-LP uses smoothing parameter $\gamma = 1/8$, the value typically used for computing the smoothed round-trip time for TCP.

D.2 Delay Threshold δ

Next, we consider the early-congestion-indication delay threshold δ of Equation (2). The example from Figure 6 illustrates the advantages of small values for the threshold δ as TCP-LP's responsiveness decreases when δ increases. However, the use of very small thresholds can substantially degrade TCP-LP's throughput in realistic scenarios. This is because even very small (and frequent) bursts of cross-traffic can cause queueing delays on a bottleneck link. TCP-LP senses these delays from the edge, and if it uses small thresholds, frequent delay oscillations can be misinterpreted as congestion indications, even in a lightly loaded network. In turn, false early congestion indications would cause a TCP-LP flow to unnecessarily decrease its sending rate.

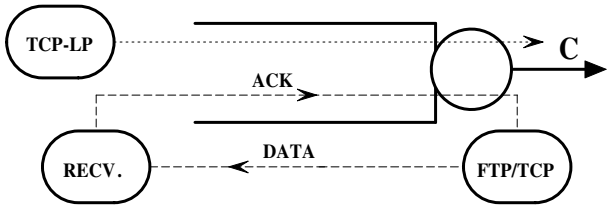


Fig. 7. Scenario with Reverse ACK Traffic

Thus, δ must be set to balance increased protocol responsiveness with avoiding false early congestion indications. To obtain the smallest value of δ capable of avoiding false indications, we devise the following experiment with reverse traffic. We consider a single TCP-LP flow in a single-bottleneck scenario, where different numbers of long-lived FTP/TCP flows operate in the reverse direction, as depicted in Figure 7. Thus, the ACK packets of the TCP flow form a cross-traffic stream that multiplexes with TCP-LP's data traffic. The objective is to set the threshold δ such that TCP-LP's throughput does not degrade in the presence of this reference ACK stream.

Figure 8 depicts TCP-LP's normalized throughput for different values of the threshold parameter δ. Observe that even this low bit-rate cross-traffic reference stream, which consists solely of ACK packets, can degrade TCP-LP's throughput substantially if the threshold is set too low. For example, as depicted in Figure 8, TCP-LP's throughput can drop to as low as 10% of the link bandwidth if the threshold δ is set to 0.01. However, the figure also indicates that the throughput improves with increasing δ, since for larger values of δ TCP-LP becomes non-sensitive to pure ACK bursts.

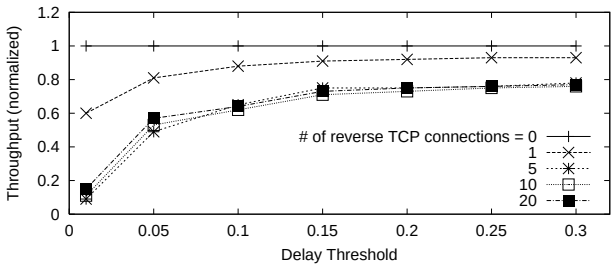


Fig. 8. Throughput vs. Threshold δ

Thus, while necessarily not comprehensive, we find that setting the threshold δ to the value of 0.15 is able to accurately decouple the influence of ACK cross-traffic streams from data cross-traffic streams. In other words, while being robust in utilizing available bandwidth in the presence of pure ACK streams, TCP-LP retains its responsive nature in the presence of pure data or aggregation of data and ACK streams.[7]

D.3 Inference Time-out *itt*

Finally, a similar tradeoff between congestion-responsiveness and throughput-aggressiveness holds for the inference time-out timer parameter. With a longer inference time-out timer, TCP-LP becomes more responsive to congestion whereas a smaller inference time-out timer causes TCP-LP to switch sooner to the more aggressive additive-increase phase. To compromise between the two, we set *itt* to three round-trip times, thereby giving enough space for a TCP-LP flow to rapidly decrease its window size in periods of persistent congestion, while at the same time allowing TCP-LP to probe the network aggressively enough.

IV. Simulation Preliminaries

In this section, we describe TCL-LP/ECN, a benchmark algorithm that uses network ECN instead of end-point delay thresholds to infer congestion. This provides means to evaluate the early-congestion-inference aspect of TCP-LP separately from its congestion-control policy. We also present the baseline simulation scenario and describe the "square-wave" and web-like background traffic patterns.

[7] Numerous additional simulations (not shown) including scenarios with hundreds of flows, heterogeneous link capacities and multiple bottlenecks corroborate that this value represents a high performance compromise between TCP-LP's responsiveness and ability to prevent false congestion indications.

A. TCP-LP/ECN Benchmark Algorithm

Here, we describe TCP-LP/ECN, a variant of TCP-LP that uses ECN for detecting congestion instead of one-way packet delays. (Recall that one of our basic design goals is to develop an end-point protocol that is able to operate without any support from the network.) Use of router-supported early congestion indication allows us to study the effectiveness of inferences from one-way packet delay to provide early inference of congestion.

We simulate TCP-LP/ECN by modifying the implementation of RED [9] in *ns-2* as follows. First, we set the minimum and the maximum RED thresholds to the value of δQ packets. Second, we configure the RED gateways to set the ECN bit in the TCP-LP packet header when the average queue size exceeds δQ as an early indication of congestion. When a TCP-LP receiver receives a data packet with the ECN bit set in the packet header, the receiver sets the ECN bit in the next outgoing ACK packet. On the other hand, packets belonging to TCP flows are neither marked nor dropped when the queue size exceeds δQ, and TCP packets are dropped only when the queue overflows. In this way, TCP-LP/ECN emulates the distributed TCP-LP protocol with the former using router queue measurements and the latter using end-point delay measurements.

B. Topology and Background Traffic

As a baseline topology, we consider many flows sharing a single congested link as shown in Figure 9. The bandwidth of this link is either 1.5 Mb/s or 10 Mb/s and it has propagation delay 20 ms. The access links have capacity 100 Mb/s and delay 2 ms, so that the minimum round-trip time for flows is approximately 50 ms. The queue size is set to 2.5 times the delay-bandwidth product. For each data point, we perform 50 simulation runs and report averages. Each simulation run lasts 1000 sec. Our *ns-2* implementation of TCP-LP is derived by modifying TCP/Reno.

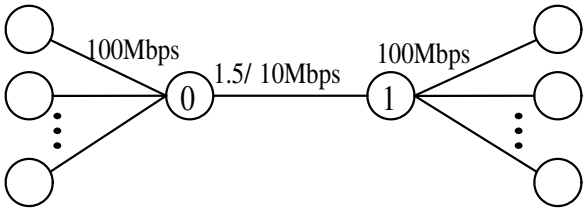


Fig. 9. Single Bottleneck Simulation Scenario

To explore the dynamics of TCP-LP, we use on-off constant-rate flows with equal on and off times, giving periodic "square-wave" patterns of available bandwidth as in reference [2]. While not representative of actual traffic patterns, this scenario is motivated by the need to systematically explore TCP-LP's ability to utilize the excess bandwidth and to study its transparency and fairness properties in the presence of dynamic background traffic. In these experiments, the available bandwidth alternates between the full link capacity of 10 Mb/s and 3.3 Mb/s when the periodic source is idle and active respectively. The period of oscillations is changed from one to 1000 round-trip times, i.e., from 50 ms to 50 sec.

Next, to explore TCP-LP's behavior with web traffic, we adopt the model developed in [6]. In this model, clients initiate sessions from randomly chosen web sites with several web pages downloaded from each site. Each page contains several objects, each of which requires a TCP connection for delivery (i.e., HTTP 1.0). The inter-page and inter-object time distributions are exponential with means of one sec and one msec, respectively. Each page consists of ten objects and the object size is distributed according to a Pareto distribution with shape parameter 1.2.

V. Simulation Experiments

We now use simulation to evaluate the performance of TCP-LP in a variety of scenarios, including FTP, "square-wave", and HTTP background traffic patterns, with long and short-lived TCP flows and both single and multiple-bottleneck network topologies. Our goal is to explore TCP-LP's behavior in both artificial and realistic network environments. We evaluate TCP-LP's impact on both the throughput and delay characteristics of competing cross-traffic. Moreover, we explore TCP-LP's ability to utilize the excess network bandwidth and to achieve fairness among competing TCP-LP flows. The TCP-LP *ns* code and simulation scripts are available at *http://www.ece.rice.edu/networks/TCP-LP*.

A. FTP and Reverse Background Traffic

We first consider simultaneous FTP downloads, where one flow uses TCP-LP and the other uses TCP. Our objectives are to examine to what extent TCP-LP can utilize excess bandwidth in the presence of greedy long-lived TCP traffic, and to investigate the extent to which TCP-LP flows perturb TCP traffic. In addition to this scenario, we also measure the throughput in simulations without TCP-LP consisting of one and two TCP flows. The results are summarized in the first row of Table I. In this scenario, there is no excess capacity available for TCP-LP, and TCP-LP slightly perturbs the TCP flows and receives a throughput of 2.7% of the link capacity for both TCP-LP and TCP-LP/ECN.

With ten FTP/TCP flows in the reverse direction, the ACKs of the forward-direction TCP flows are delayed thereby increasing their round-trip time and ACK losses, and decreasing their throughput. Thus, excess capacity is indeed available for TCP-LP flows. In particular, the second row of Table I illustrates that the throughput of the (forward) TCP flow in this case is 49.7%. With the presence of a TCP-LP flow, the TCP flow's throughput is only marginally reduced to 49.3%, indicating that TCP-LP achieves nearly perfect TCP transparency while achieving 7.3% throughput.

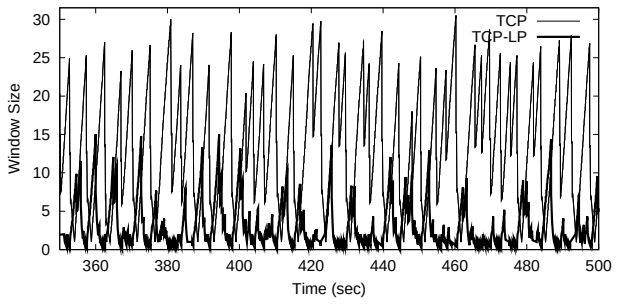


Fig. 10. TCP and TCP-LP's Congestion Window

TABLE I
NORMALIZED THROUGHPUT (%)

scenario	TCP	TCP vs. TCP-LP	TCP vs. TCP-LP/ECN	TCP vs. TCP
no reverse TCP traffic	100	96.8 vs. 2.7	96.8 vs. 2.7	50 vs. 50
reverse TCP traffic	49.7	49.3 vs. 7.3	49.1 vs. 8	32 vs. 32

Figure 10 depicts the temporal dynamics of this scenario and illustrates that TCP's congestion window widely oscillates in the range between zero and 30 packets. The window of the TCP-LP flow, also depicted, is able to track TCP's oscillation and increases its own window size when TCP's window decreases, and via early congestion inference, TCP-LP quickly backs off when the TCP flow ramps up its window size. By the time the TCP flow's window reaches its maximum of 30 packets, TCP-LP is in the inference phase, waiting for the next opportunity to utilize excess bandwidth.

B. Square-wave Background Traffic

Next, we explore TCP-LP's performance in the presence of square-wave background traffic as described in Section IV-B.

B.1 Square Wave Period

Our first experiments investigate TCP-LP's ability to utilize excess bandwidth remaining from periodic on-off flows that transmit at constant rate when "on". Figure 11 depicts the bandwidth utilized by TCP, TCP-LP and TCP-LP/ECN, normalized to 6.6 Mb/s, the average excess bandwidth left unused by the square-wave background traffic. Each point in the figure represents the normalized bandwidth, utilized by the respective protocol, for a given period of the square-wave's oscillation. For comparison, we also depict the normalized average available bandwidth curve.

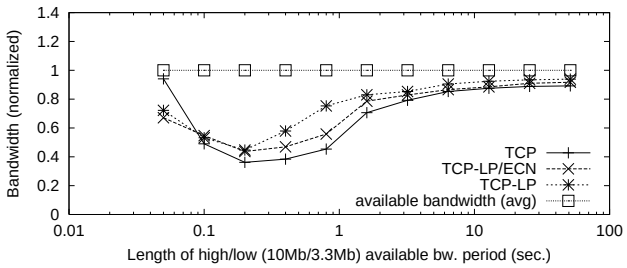


Fig. 11. Utilized Available Bandwidth vs. Square Wave Period

Observe that all three curves in Figure 11 have similar shape, and all three protocols utilize approximately only 50% of the available bandwidth when the square-wave period is too small (e.g., 0.2 seconds). Surprisingly, in this regime, both TCP-LP and TCP-LP/ECN utilize more available bandwidth than TCP. This is due to the early congestion indication and responsive congestion avoidance policy of the TCP-LP protocol, which is able to defer access to the cross-traffic bursts (from 0 to 2/3 C in this case) while avoiding entering the exponential-backoff phase.

B.2 Aggregation Level

Next, we explore the impact of the number of flows under a fixed square wave period of 6.4 sec. Figure 12 illustrates that with higher levels of aggregation consisting of even 5 flows, TCP flows quickly overcome the performance problem of Figure 11. On the other hand, for TCP-LP utilization increases more slowly with aggregation level, as with a small number of flows, TCP-LP is not able to develop large congestion windows because it senses the existence of other competing TCP-LP flows and decreases its window accordingly. However, TCP-LP overcomes this problem with a larger number of multiplexed flows.

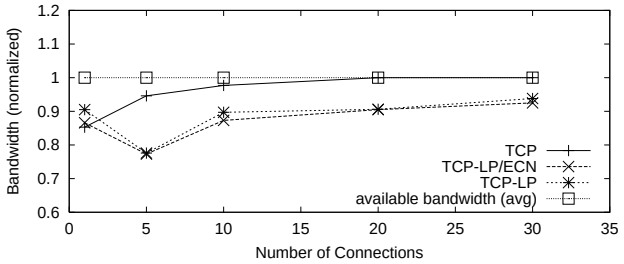


Fig. 12. Utilized Available Bandwidth vs. Number of Flows

B.3 Fairness

Here we study fairness among TCP-LP flows using Jain's fairness index [14]. The index, always between 0 and 1, is 1 if all flow throughputs are the same. If only k of the n users receive equal throughput and the remaining n - k users receive zero throughput, the fairness index is k/n. Our experiments include ten flows of the same type (TCP, TCP-LP or TCP-LP/ECN) that compete with the same non-responsive square wave background traffic.

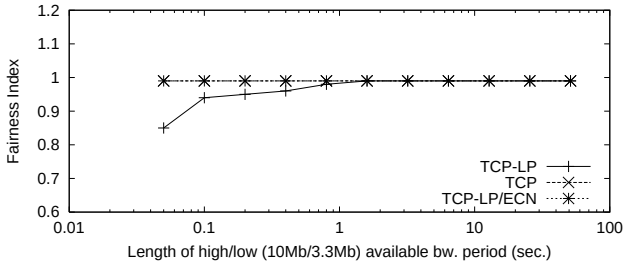


Fig. 13. Fairness Index vs. Square Wave Period

Figure 13 depicts the fairness indexes of three protocols for different periods of square wave oscillations. First, observe that for both TCP and TCP-LP/ECN, the fairness index is approximately equal to 1 for all periods. However, TCP-LP's fairness

index is slightly below one for time scales of up to 400 ms. Examining the traces, we conclude that this originates from inaccurate estimates of the minimum and maximum delays. In most cases, one TCP-LP flow over-estimates the minimum delay value d_{min} due to wide and frequent oscillations of the background traffic. For this reason, it sends more than its fair share and the fairness index drops slightly. However, as the oscillation period increases, all flows use periods of low cross-traffic rate to accurately estimate the minimum one-way delay.

C. HTTP Background Traffic

Here, we explore TCP-LP's behavior in an environment dominated by web-like transactions in the scenario described in Section IV-B. The performance measure of interest is the web-file retrieval (response) time, and we investigate TCP-LP's impact on this measure. As a standard of idealized performance, we use the measured retrieval times in a scenario with only web-traffic present in the system. Further, we perform multi-node experiments to study issues such as heterogeneous round-trip times and early congestion inference with multiple bottlenecks.

We run four experiments for the topology of Figure 9 with a link capacity of 1.5 Mb/s. In addition to web traffic between nodes zero and one, there is one FTP connection that operates in the same direction as the web-traffic. This connection is a long-lived bulk transfer and is a candidate for low-priority service. In the first three experiments, the FTP connection uses TCP-LP, TCP-LP/ECN, and TCP. Finally, to measure web-traffic response times without any cross-traffic, we perform a fourth experiment in which no FTP traffic is generated. For the web transactions, we measure and average the response times for different sized objects.[8]

C.1 Impact on HTTP Response Times

To explore TCP-LP's impact on web traffic, we compare HTTP file retrieval times with and without background TCP-LP bulk transfers.[9] Figure 14 depicts the averaged *difference* between the two transfer times. For example, when TCP-LP is used for a long-lived file transfer, the mean retrieval time for a 10 kB web-file is 0.49 sec. On the other hand, this retrieval time is 0.43 sec when there is no TCP-LP file transfer, hence the point (10, 0.06) in the figure. These experiments illustrate the non-intrusive aspect of TCP, as the long-lived TCP-LP bulk transfer flow only slightly increases the mean web-traffic response time, with increasing transparency achieved with larger HTTP file sizes.

C.2 Impact of High vs. Low Priority Bulk Transfer

We next show that if the bulk transfer flow uses TCP rather than TCP-LP, then the web response times are significantly degraded. Figure 15 depicts web-file response times normalized by the response times obtained when the background file transfer uses TCP. Because of this normalization, the curve labeled "TCP" in Figure 15 is a straight line with a value of one.

[8] As in [11], file sizes are grouped into 85 bins, each of which spans an interval [x,1.1x], and the average is taken over each bin.

[9] Throughput the paper, the HTTP response-time simulations use the same sample path of file sizes and think times.

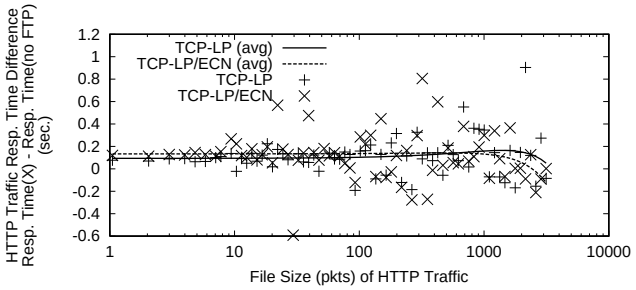


Fig. 14. Resp. Time Diff. (sec.) vs. File Size (kB) for HTTP Traffic

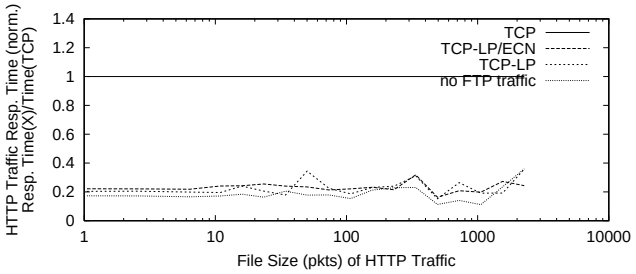


Fig. 15. Norm. Resp. Time vs. File Size (kB) for HTTP Traffic

Observe that use of TCP-LP for bulk data transfer reduces the web traffic response times by approximately 80% compared to TCP bulk transfer. For example, the average response time for the 10 kB file from the web-traffic stream is 2.46 sec when web traffic multiplexes with a TCP bulk-transfer background flow. This time is considerably larger than the 0.49 sec response time when TCP-LP is used for the bulk data transfer. TCP-LP's reduction in response time for web traffic occurs because without it, the TCP bulk-transfer demands its fair share of network bandwidth when competing with web-traffic. On the other hand, the bulk-transfer flow itself utilizes 61% of the bandwidth when TCP is used, only 10% more than when TCP-LP is used. This result emphasizes the benefits of low prioritization of bulk data transfers over web-traffic, which TCP-LP achieves in a distributed manner.

D. Multiple Bottlenecks

We next consider a more realistic multiple bottleneck scenario using the topologies of Figures 16 and 19. In all experiments, links 0-1, 1-2 and 2-3 have capacity of 1.5 Mb/s, while all the others have capacity of 100 Mb/s.

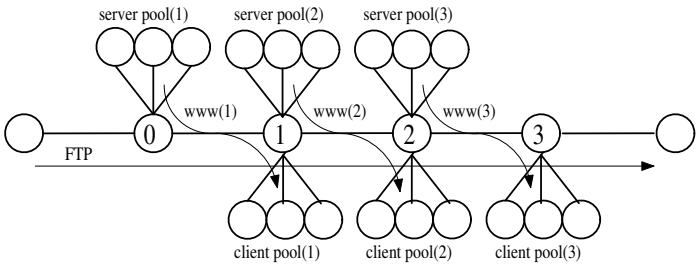


Fig. 16. First Topology for Multiple Bottlenecks

D.1 RTT Heterogeneity

To study TCP-LP when its round-trip time increases compared to round-trip times of competing HTTP flows, we consider the scenario in which the bulk file-transfer flow traverses multiple bottlenecks as shown in Figure 16. There are three server and client pools, each of which generates cross-traffic on different bottleneck links.

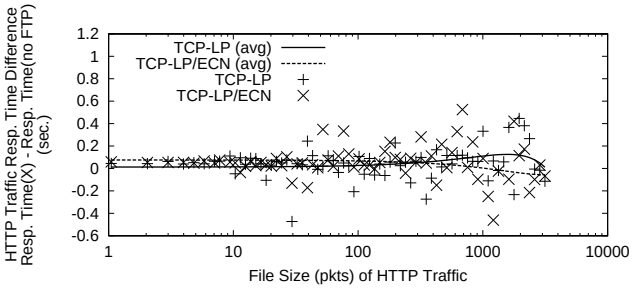


Fig. 17. Resp. Time Diff. (sec.) vs. File Size (kB) for HTTP Traffic

Figure 17 depicts the averaged *difference* between HTTP file response times with and without the presence of a bulk-transfer TCP-LP flow. Observe that despite having the average round-trip time three times as large, TCP-LP retains its non-intrusiveness to the HTTP/TCP flows. This confirms the modeling result from Section III-C, which states that TCP-LP flows are non-intrusive to TCP flows even if their round-trip times are much larger. Also, we do not observe any substantial difference between TCP-LP and TCP-LP/ECN, except that TCP-LP is slightly more responsive for large files.

D.2 Multi-hop Bulk Transfer

Figure 18 depicts the response times for different sized objects from all three pools normalized by the response times obtained when background FTP transfer uses TCP. We observe that the benefit of prioritization observed in the single bottleneck scenario still holds in this multiple-bottleneck scenario, although less pronounced. The difference is because the long-lived TCP flow is now less intrusive to web traffic due to its larger round-trip time.

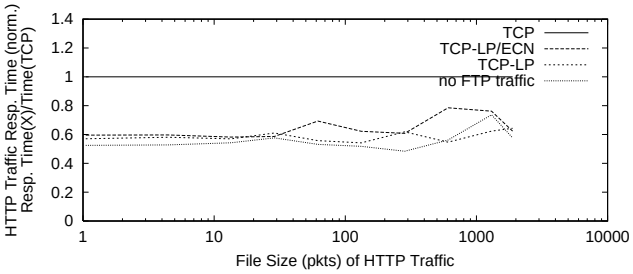


Fig. 18. Norm. Resp. Time vs. File Size (kB) for HTTP Traffic

D.3 Multi-hop Web Traffic

Next, we consider the scenario in which web traffic traverses multiple hops and three FTP connections each traverse a single hop as depicted in Figure 19. Thus, the FTP flows in this scenario play the role of "fast elephants", a term for long-lived flows with short round-trip times [21].

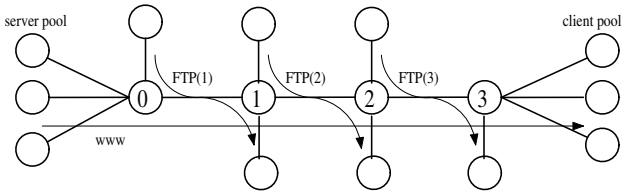


Fig. 19. Second Topology for Multiple Bottlenecks

Figure 20 depicts the averaged difference between web file response times with and without the three TCP-LP bulk transfers. In this scenario, the small TCP-LP round-trip time only improves its responsiveness and non-intrusiveness to competing web-traffic such that it becomes fully transparent to TCP. For example, the mean response time for the 10 kB file is 0.98 sec, while it is 0.74 sec in the idealized scenario when there are no FTP downloads in the system. This is revealed as the point (10, 0.24) for TCP-LP in Figure 20. Observe that the absolute difference in response times increases three times in this scenario when compared to the single-node scenario simply because the HTTP traffic now traverses three congested hops. However, the *per-node* impact of the bulk-transfer TCP-LP flows is approximately unchanged.

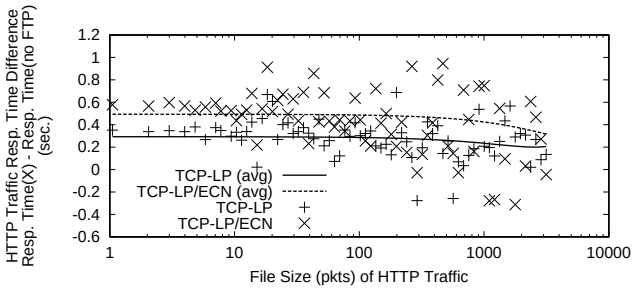


Fig. 20. Resp. Time Diff. (sec.) vs. File Size (kB) for HTTP Traffic

Finally, for comparison, we again explore the system behavior when TCP is used for bulk data transfers. Figure 21 depicts the normalized response times for HTTP file retrievals. The figure indicates that "fast TCP elephants" severely impede the performance of web traffic that traverses multiple hops. For example, in this scenario, the average response time for a 10 kB file from the HTTP traffic stream is 14.27 sec.

This poor performance is because many web-traffic flows experience loss of their first packet which requires waiting for a default time-out interval of 3 sec before resending. According to our results, each TCP flow from the web stream experiences four to five such timeout intervals on average. An interested reader can find more details on this problem in [11]. On the other hand, the results from Figure 21 indicate that simple two-class prioritization achieved by TCP-LP can successfully provide a desirable system behavior. While TCP-LP attains 52% of the bandwidth (10% less than TCP), it improves web-traffic response times by more than 90%.

VI. Protocol Implementation and Internet Experiments

In this section, we first describe the implementation details of TCP-LP in Linux and explain the specific use of the TCP

TABLE II
NORMALIZED TCP THROUGHPUT (%) VS. NUMBER OF FLOWS

Number of TCP and TCP-LP flows	1	2	5	10	15	20
Normalized TCP throughput	99.49	99.25	99.50	99.02	98.29	99.15

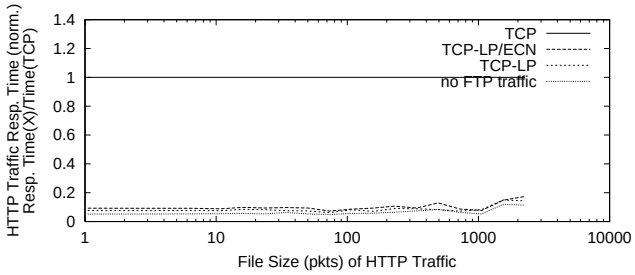


Fig. 21. Norm. Resp. Time vs. File Size (kB) for HTTP Traffic

window-scaling and timestamping options from [12]. Next, we examine the protocol performance on a testbed and evaluate three important TCP-LP features. The first is non-intrusiveness to TCP traffic in large aggregation regimes; the second is the ability of both a single TCP-LP flow and aggregates to utilize the available bandwidth in the presence of highly dynamic cross traffic; the third is fairness among TCP-LP flows. Furthermore, we perform Internet measurements to evaluate the extent to which TCP-LP can utilize excess bandwidth in the presence of greedy TCP traffic. Finally, we measure TCP-LP's performance during a 24-hour period in a wide-area network and explore the impact of time-of-day effects on TCP-LP's throughput.

A. Implementation

Our implementation of TCP-LP is derived by modifying the Linux-2.4.19-web100 kernel, which applies TCP Sack, and the TCP-LP source code is available at *http://www.ece.rice.edu /networks/TCP-LP*. Besides having monitoring and debugging features, the above kernel supports the window-scaling option [12], which allows the use of larger window sizes (about 1 GByte) and enables TCP-LP to fully utilize the available bandwidth. Note that many TCP stacks do not support this option by default, such that the TCP header allocates only 16 bits for window advertisement, which limits maximum window size to 64 kBytes.

TCP-LP also uses the TCP timestamping option from [12] for the one-way delay measurements as explained in Section III. In short, each TCP packet carries two four-byte timestamp fields. A TCP-LP sender timestamps one of these fields with its current clock value when it sends a data packet. On the other side, the receiver echoes back this timestamp value and in addition timestamps the ACK packet with its own current time. Thus, using these two values, the TCP-LP sender may measure one-way packet delays. However, each end-system measures time (and timestamps packet fields) in the "local unit" that corresponds to the number of clock ticks elapsed since a reference point in time. Since the clock granularity[10] may be different for the TCP-LP sender and receiver, the sender has to first estimate the receiver's clock granularity in order to use its timestamps for one-way delay measurements. The TCP-LP sender performs this task by monitoring the ACK packet's timestamp field and measuring the number of remote ticks elapsed during the one-second period after the connection establishment. Finally, the sender accurately estimates the receiver's clock granularity by choosing a value from the set of possible values (e.g., 100, 512 or 1024) that is closest to the measured number of remote clock ticks per second.

To the best of our knowledge, TCP-LP is the first TCP stack that uses timestamping option for computing one-way delays. This option is originally developed to alleviate computation of round-trip times. We emphasize the fact that the use of one-way delays is an essential requirement for low-priority transport protocols in the Internet, as will be discussed in detail in Section VII. Finally, while we do not observe any problems with a drift between the sender and the receiver clocks, TCP-LP nevertheless applies resets of d_{min} and d_{max} on three minute intervals by default.

B. Testbed Experiments

Here, we report the results obtained on a testbed at Rice University. The testbed consists of two Linux clusters, as shown in Figure 9, with the difference that the bottleneck capacity is 100 Mb/s. Regular TCP (non-TCP-LP) flows apply the Linux-2.4.19-web100 kernel using TCP Sack.

B.1 TCP Transparency in Large Aggregation Regimes

To achieve TCP-transparent behavior in large aggregation regimes, TCP-LP reduces its window size to one packet per RTT in the presence of TCP flows and decreases packet size to 64 Bytes. Here, we consider scenarios with many flows to evaluate whether TCP-LP remains non-intrusive in such regimes.

We perform experiments with simultaneous TCP and TCP-LP file transfers where the number of flows in the system (both TCP and TCP-LP) increases from one to 20, as shown in Table II. The second row of the table shows aggregate TCP throughput normalized to the throughput obtained when there are no TCP-LP flows in the network. Observe that the influence of TCP-LP flows is indeed marginal and that TCP throughput degrades by less than 1% on average. More importantly, observe that as the number of TCP-LP flows increases, the TCP throughput does not degrade. According to the theoretical computations from Section III-B.4, we would require more than 390 TCP-LP flows in the above experiment to degrade TCP's throughput by more than 2%.

B.2 Available Bandwidth Utilization

Next, we evaluate TCP-LP's ability to utilize the available bandwidth in the presence of extremely dynamic cross-traffic.

[10]Typically 100, 512 or 1024 ticks per second.

To this end, we perform an experiment similar to the one from Section V-B with square-wave UDP cross-traffic oscillating between 0 and 2/3 of the link capacity, and where the period of oscillation changes from 50 ms up to 51.2 sec. We generate the cross-traffic stream using active probing software from [18]. Figure 22 depicts the bandwidth utilized by TCP-LP and TCP, normalized to the average excess bandwidth left unused by the square-wave's oscillation.

Observe that the curves in Figure 22 are somewhat different from the curves in Figure 11 since here we do not notice significant degradation of TCP and TCP-LP throughputs on shorter time-scales of the background traffic. We believe that his is due to shorter round-trip times in this scenario (the minimum RTT is 2 ms) that enable both TCP and TCP-LP flows to apply a more robust control and avoid entering the exponential-backoff phase. Furthermore, observe that again both protocols have similar behavior, with TCP-LP having slightly better performance on longer time-scales.

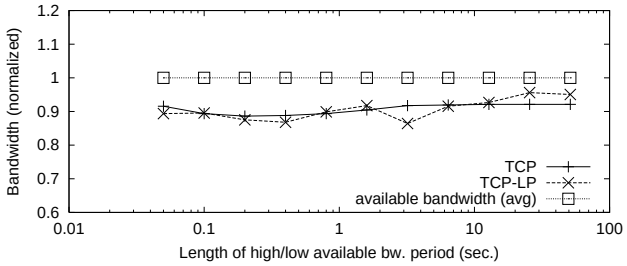


Fig. 22. Utilized Available Bandwidth vs. Square Wave Period

B.3 Aggregation Level and Fairness

Finally, we repeat the simulation experiment from Section V-B.2 in our testbed and explore the impact of the number of flows under a fixed square wave period of 6.4 sec. Figure 23 shows that TCP-LP utilization increases very quickly (quicker than TCP) with aggregation level in this scenario. Also, we measure the fairness index of ten TCP-LP flows and our results (not shown) confirm that TCP-LP flows achieve inter-TCP-LP fairness.

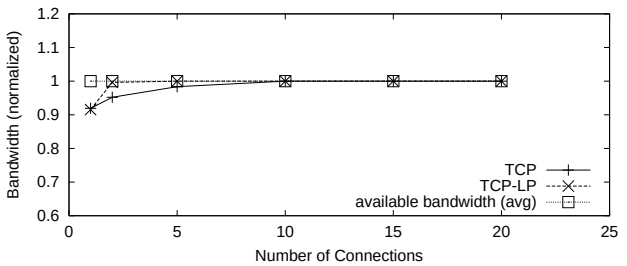


Fig. 23. Utilized Available Bandwidth vs. Number of Flows

C. Internet Experiments

Next, we evaluate TCP-LP's performance in the Internet. All flows in the experiments (both TCP and TCP-LP) are originated from Rice University (Houston, TX) and the receivers are located 14 hops away, at SLAC (Stanford, CA). The flows traverse the Rice campus network, local and regional providers networks, and finally the Stanford campus network.

C.1 TCP-LP and a Greedy Long-Lived TCP Flow

We consider simultaneous FTP downloads, where one flow uses TCP-LP and the other uses TCP. Our objective is two-fold. First, we want to check TCP-LP's non-intrusiveness property in a WAN environment. Second, we want to investigate to what extent can TCP-LP utilize excess bandwidth in the presence of a greedy long-lived TCP traffic. To obtain a time-dependent function of the utilized bandwidth, we perform simultaneous downloads each five minutes, and thus obtain 12 throughput samples per hour for each of the flows.

Figure 24 depicts the TCP and TCP-LP throughput over a 12-hour (720 minutes) period.[11] First, observe that whenever TCP throughput is high (around 60 Mb/s), TCP-LP throughput remains low, thus confirming its TCP-transparent property. On the other hand, when the cross-traffic activity is strong enough (see Figure 24 when the x-axis is less than 200 minutes), TCP throughput drops down significantly, yet TCP-LP does not utilize substantially more bandwidth because it also gives priority to cross-traffic flows. However, observe that there are times (e.g., 200 min, 300 min, 440 min), when cross-traffic can simply hinder TCP from fully utilizing the available bandwidth (by forcing it to enter exponential backoff), yet leaving some bandwidth unused. TCP-LP detects such moments and successfully fills these gaps in TCP throughput. Thus, while these events are much less pronounced than in Figure 10 (due to the lack of reverse cross-traffic), TCP-LP demonstrates an important ability to detect and exploit such events.

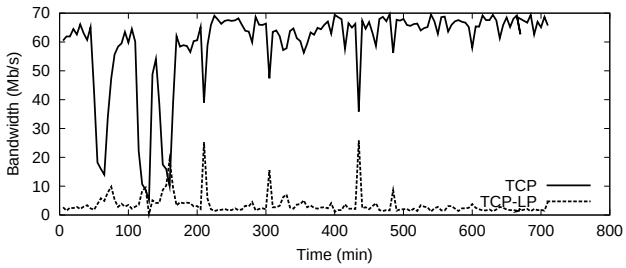


Fig. 24. TCP and TCP-LP Throughput vs. Time

C.2 Time-of-Day Effects

Finally, we explore the impact of time-of-day effects on TCP-LP's throughput. Naturally, our hypothesis is that more bandwidth is available during nights when the network is less utilized. Furthermore, our goal is to quantify the amount of bandwidth available to a TCP-LP flow, and to compare it to TCP throughput. To obtain the desired time-dependent functions of TCP and TCP-LP throughputs, while mitigating the above simultaneous file transfers effects (where TCP utilizes the entire available bandwidth), we *interchangeably* measure TCP and TCP-LP throughputs in the 5-minute intervals over a 24-hour period. In this way, we obtain totally 12 samples per hour, six

[11] The experiments took place during a weekend, and that is why the figure lacks time-of-day effects.

for each of the flows. While necessarily not comprehensive (due to traffic fluctuations over short time intervals), this methodology provides a reasonable way to independently measure TCP and TCP-LP throughputs on the same network path.

Figure 25 depicts the TCP and TCP-LP throughputs measured over a 24-hour period, starting at midnight. Note first that the time-of-day effects are clearly observable for both TCP and TCP-LP. As expected, the effects are more pronounced for the TCP-LP flow, which gives priority to *all* flows on the end-to-end path, and utilizes only the bandwidth that is left unused. On the other hand, TCP competes for resources with the cross traffic flows, and eventually utilizes its share of bandwidth. Figure 25 indicates that in the after-midnight hours (midnight to 8 a.m.) as well as in the after-working hours (5 p.m. to midnight), TCP-LP throughput fluctuates between 50% and 100% of TCP's throughput at the same time, utilizing approximately 75% of the TCP bandwidth on average. On the other hand, during working hours (8 a.m. - 5 p.m.), TCP-LP throughput fluctuates between 0% and 100% of TCP's throughput, utilizing approximately 45% of the TCP bandwidth in this interval on average. Thus, the experiment illustrates that despite its low-priority nature, a TCP-LP flow is able to utilize significant amounts of available bandwidth in a wide-area network environment.

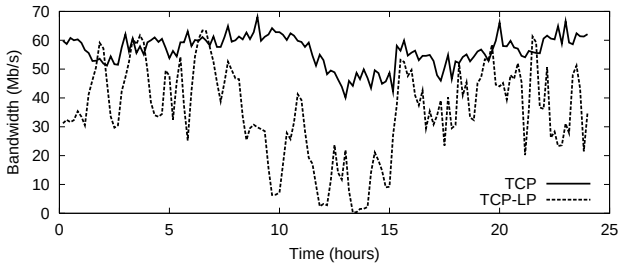


Fig. 25. TCP and TCP-LP Throughput vs. Time

VII. Related Work

The most related protocol to TCP-LP is TCP-Nice [22], which aims to provide a system support for background file replication. TCP-LP [16] and TCP-Nice are developed in parallel and completely independently from each other. TCP-Nice is designed as an extension to TCP-Vegas [3], with a more sensitive congestion detector. It uses RTT-threshold-based congestion indication scheme where the congestion is indicated if more than 50% packets encounter RTT-delay threshold. On the other hand, recall that TCP-LP reacts to one-way-delay threshold-based congestion indications as recommended in [8], and more aggressively decreases window size in times of persistent congestion. The key difference between TCP-LP and TCP-Nice is the use of *one-way* delay measurements (TCP-LP) vs. round-trip delays (TCP-Nice) for congestion indication. The use of one-way delays is fundamentally required for low-priority transport protocols because the cross-traffic in the direction from the receiver to the sender may significantly prevent TCP-Nice from utilizing the excess network bandwidth. More precisely, if TCP-Nice's ACK packets are persistently delayed on the reverse path (such that the round-trip times are beyond the round-trip threshold), TCP-Nice may achieve near *zero* throughput, *independently* from the actual amount of the excess bandwidth in the network. On the other hand, TCP-LP does not have this problem as it suppresses the influence of the reverse cross-traffic by using one-way delay measurements.

While no protocols other than TCP-LP and TCP-Nice provide an end-point realization of a low priority service, there are related efforts in several areas. First, one of the key TCP-LP mechanisms is the use of packet delay measurements for early congestion indications. Jain's delay-based congestion avoidance protocol [14], Wang *et al.'s* TCP/Dual [24], Brakmo *et al.'s* TCP/Vegas [3] all use delay-based congestion control in an effort to increase TCP throughput due to a reduced number of packet losses and timeouts, and a reduced level of congestion over the path. The key difference between TCP-LP and RTT-based congestion control protocols is in their primary objective. While the former aim to achieve *fair-share* rate allocations, TCP-LP aims to utilize only excess bandwidth. In this context, we also note that Martin *et al.* [17] suggest that RTT-based congestion avoidance is problematic to incrementally deploy in the Internet due to degraded throughput as compared to TCP/Reno flows. Observe that TCP-LP does not suffer from this problem again due to its different objective: TCP-LP targets the excess-capacity rate vs. the fair-share rate. Thus, TCP-LP is incrementally deployable and could be successfully used by *any* subset of Internet users.

Second, TCP-LP uses early congestion indication (earlier than TCP) as a basis for achieving class differentiation. Clark and Feng [5] proposed RIO (RED with In and Out) in which routers apply different marking/dropping functions for different classes of flows, thereby providing service differentiation. While similar in philosophy to TCP-LP, TCP-LP develops an *end-point* realization of early congestion indication for the purpose of low-priority transfer. Consequently, TCP-LP is applicable over routers and switches that provide no active queue management or service differentiation.

Third, TCP-LP relates to adaptive bandwidth allocation schemes that aim to minimize file-transmission times using file-size-based service differentiation. Guo and Matta [11] use RIO in core routers and a packet classifier at the edge to distinguish between long- and short-lived TCP flows. Yang and de Veciana [25] develop TCP/SAReno in which the AIMD parameters dynamically depend on the remaining file size. While TCP-LP also substantially improves file-transmission times in the best-effort class, the key difference between TCP-LP and the above schemes is that it provides *strict* low-priority service, independent of the file size.

Next, as TCP-LP targets transmitting at the rate of available bandwidth, it is related to cross-traffic estimation algorithms which attempt to infer the available bandwidth via probing (see reference [13] for a thorough review of such algorithms). For example, Ribeiro *et al.* [20] and Alouf *et al.* [1] provide algorithms for estimation of parameters of competing cross-traffic under multifractal and Poisson models of cross traffic. In contrast, TCP-LP provides an adaptive estimation of available bandwidth by continually monitoring one-way delays and dynamically tracking the excess capacity. Similarly, Jain and Dovrolis [13] develop *pathload*, a delay-based rate-adaptive probing scheme for estimating available bandwidth. The key difference

between *pathload* and TCP-LP is that the latter aims to *utilize* the available bandwidth, while the former only estimates it. Moreover, TCP-LP addresses the case of multiple flows *simultaneously* inferring the available bandwidth by providing each with a fair share (according to TCP fairness), an objective that is problematic to achieve with probes.

Finally, end-point admission control algorithms also use probes to detect if sufficient bandwidth is available for real-time flows [4]. Unfortunately, such techniques have a "thrashing" problem when many users probe simultaneously and none can be admitted. While TCP-LP targets a low rather than high priority class, its basic ideas of adaptive and transparent bandwidth estimation could be applied to end-point admission control and alleviate the thrashing condition.

VIII. Conclusions

This paper presents TCP-LP, a protocol designed to achieve low-priority service (as compared to the existing best-effort class) from the network endpoints. TCP-LP allows low-priority applications such as bulk data transfer to utilize excess bandwidth without significantly perturbing non-TCP-LP flows. TCP-LP is realized as a sender-side modification of the TCP congestion control protocol and requires no functionality from the network routers nor any other protocol changes. Moreover, TCP-LP is incrementally deployable in the Internet. We performed an extensive set of *ns-2* simulations and Internet experiments and showed that 1) TCP-LP is largely non-intrusive to TCP traffic (including very large aggregation regimes) while at the same time, TCP-LP flows can successfully utilize a large portion of the excess network bandwidth. 2) In practice, significant excess capacity is available even in the presence of "greedy" long-lived FTP/TCP flows due to factors such as ACK delays from reverse traffic. 3) Competing TCP-LP flows share excess bandwidth fairly. 4) File transfer times of best-effort web traffic are significantly reduced when long-lived bulk data transfers use TCP-LP rather than TCP. 5) Despite their low-priority nature, even longer-RTT TCP-LP flows are able to utilize substantial amounts of spare available bandwidth in a wide-area network environment.

References

[1] S. Alouf, P. Nain, and D. Towsley. Inferring network characteristics via moment-based estimators. In *Proceedings of IEEE INFOCOM '01*, Anchorage, Alaska, April 2001.

[2] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. Dynamic behavior of slowly-responsive congestion control algorithms. In *Proceedings of ACM SIGCOMM '01*, San Diego, CA, August 2001.

[3] L. Brakmo and L. Peterson. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, Oct. 1995.

[4] L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: Architectural issues and performance. In *Proceedings of ACM SIGCOMM '00*, Stockholm, Sweden, August 2000.

[5] D. D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, Aug. 1998.

[6] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *Proceedings of ACM SIGCOMM '99*, Vancouver, British Columbia, September 1999.

[7] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Z.-L. Zhang. Theories and models for Internet quality of service. *IEEE*, 90(9):1565–1591, Sept. 2002.

[8] S. Floyd. TCP and explicit congestion notification. *ACM Computer Comm. Review*, 24(5):10–23, 1994.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[10] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, Aug. 1995.

[11] L. Guo and I. Matta. The war between mice and elephants. In *Proceedings of IEEE ICNP '01*, Riverside, CA, November 2001.

[12] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance, May 1992. Internet RFC 1323.

[13] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, Aug. 2002.

[14] R. Jain. A delay based approach for congestion avoidance in interconnected heterogeneous computer networks. *ACM Computer Comm. Review*, 19(5):56–71, Oct. 1989.

[15] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley, 2000.

[16] A. Kuzmanovic and E. Knightly. TCP-LP: A distributed algorithm for low priority data transfer. In *Proceedings of IEEE INFOCOM '03*, San Francisco, CA, April 2003.

[17] J. Martin, A. Nilsson, and I. Rhee. The incremental deployability of RTT-based congestion avoidance for high speed TCP Internet connections. In *Proceedings of ACM SIGMETRICS '00*, Santa Clara, CA, June 2000.

[18] A. Pasztor and D. Veitch. High precision active probing for Internet measurement. In *Proceedings of INET '01*, Stockholm, Sweden, 2001.

[19] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification to IP, January 1999. Internet RFC 2481.

[20] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, and R. Baraniuk. Multifractal cross-traffic estimation. In *Proceedings of ITC '00*, Monterey, CA, Sept. 2000.

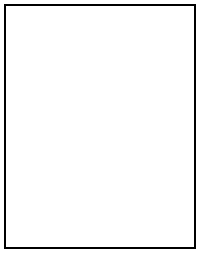
[21] S. Sarvotham, R. Riedi, and R. Baraniuk. Connection-level analysis and modeling of network traffic. In *Proceedings of IEEE/ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, Nov. 2001.

[22] A. Venkataramani, R. Kokku, and M. Dahlin. TCP Nice: A mechanism for background transfers. In *Proceedings of OSDI '02*, Boston, MA, December 2002.

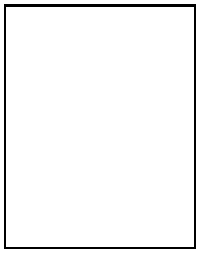
[23] M. Vojnovic, J.-Y. Le Boudec, and C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times. In *Proceedings of IEEE INFOCOM '00*, Tel Aviv, Israel, March 2000.

[24] Z. Wang and J. Crowcroft. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. *ACM Computer Comm. Review*, 22(2):9–16, Apr. 1992.

[25] S. Yang and G. de Veciana. Size-based adaptive bandwidth allocation: Optimizing the average QoS for elastic flows. In *Proceedings of IEEE INFOCOM '02*, New York, NY, June 2002.



Aleksandar Kuzmanovic is an assistant professor in the Department of Electrical Engineering and Computer Science at Northwestern University. He received his B.S. and M.S. degrees from the University of Belgrade, Serbia, in 1996 and 1999 respectively. He received the Ph.D. degree from the Rice University in 2004. His research interests are in the area of computer networking with emphasis on design, security, analysis, theory, and prototype implementation of protocols and algorithms for the wired and wireless Internet.



Edward W. Knightly (SM 04) is an associate professor of Electrical and Computer Engineering at Rice University. He received the B.S. degree from Auburn University in 1991 and the M.S. and Ph.D. degrees from the University of California at Berkeley in 1992 and 1996 respectively. He is an associate editor of IEEE/ACM Transactions on Networking. He served as technical co-chair of IEEE IWQoS 1998 and IEEE INFOCOM 2005 and served on the program committee for numerous networking conferences including ICNP, INFOCOM, IWQoS, MobiCom, and SIGMETRICS. He received the National Science Foundation CAREER Award in 1997 and the Sloan Fellowship in 2001. His research interests are in the areas of mobile and wireless networks and high-performance and denial-of-service resilient protocol design.