TCP-LP: A Distributed Algorithm for Low Priority Data Transfer

Aleksandar Kuzmanovic and Edward W. Knightly
Department of Electrical and Computer Engineering
Rice University
Houston, TX 77005, USA
Email: {akuzma,knightly}@ece.rice.edu

***Abstract*— Service prioritization among different traffic classes is an important goal for the future 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 congestion indications and a TCP-transparent congestion avoidance policy. Our simulation results show 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 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.**

I. Introduction

MOTIVATED by the diversity of networked applications, a significant effort has been made to provide differentiation mechanisms in the Internet, e.g., [1]. However, despite the availability of simple and scalable solutions (e.g., [2]), 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, its 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.

This research is supported by NSF Grants ANI-0085842 and ANI-0099148, the Department of Energy, and by a Sloan Fellowship.

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 [3], and performance optimization (e.g., to select a mirror server with the highest available bandwidth). Current techniques (e.g., [4], [5], [6]) 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 on one-way packet delays as early indications of network congestion vs. packet losses 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 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.

Finally, we perform an extensive set of *ns-2* simulation experiments and study TCP-LP's characteristics in a variety of scenarios. 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. Thus, our simulation results indicate that TCP-LP is a practically applicable protocol that accurately achieves 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. Finally, in Sections VI and VII 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. 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 acknowledgement. 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.

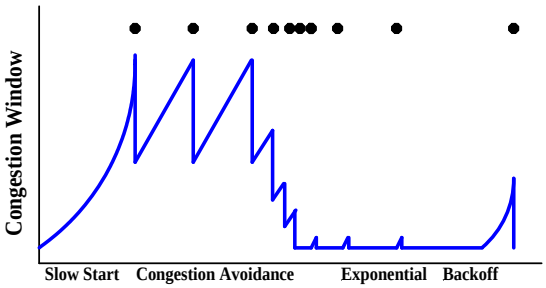


Fig. 1. Behavior of TCP Congestion Control

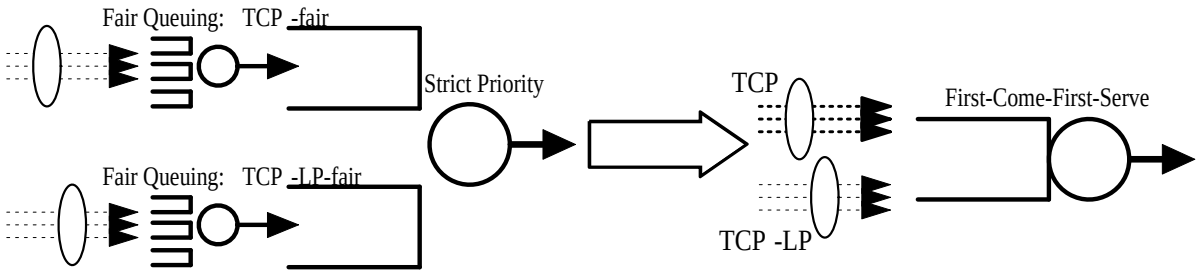
One objective of TCP congestion control is for each flow to transmit at its fair rate at its bottleneck link. While biasing rates in favor of flows with small round-trip times, we nonetheless refer to TCP as "fair" in the discussion below.[1]

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 [8]) 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 other TCP-LP flows.[2]

[1] TCP's fairness properties are studied in depth in [7] for example.

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



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

Fig. 2. Reference Model and TCP-LP Realization

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

The goal of TCP-LP is to provide low priority service in the presence of TCP traffic. To achieve this goal, 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 to inform TCP-LP flows of lesser congestion levels than TCP flows. However, given the absence of such network support, we devise an endpoint 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.

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.[3] 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. A simple exponentially weighted moving average is computed as

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

[3]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.

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

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

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.

2) Delay Measurement: TCP-LP obtains samples of one-way packet delays using the TCP timestamp option from [9]. 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 [10]. 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.

B. Congestion Avoidance Policy

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. A 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 here is to develop a congestion control policy that is able to *approximate* the desired dynamic behavior.

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 once-per round-trip time, as recommended for ECN flows [11], would result in too slow a response to achieve TCP transparency.

To compromise between the two, 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 [11], 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.

3) Pseudo Code: 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.

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

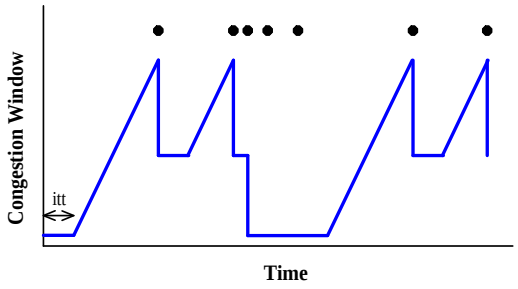


Fig. 4. Behavior of TCP-LP Congestion Avoidance Phase

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.[4]

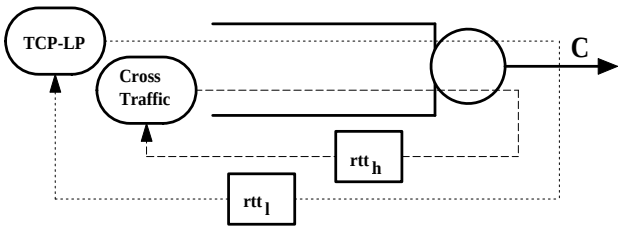


Fig. 5. Simplified Model of Heterogeneous RTT Effects

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, this time is given by the solution to

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

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 the Equation (3), the term $C+(\alpha_l/rtt_l+\alpha_h/rtt_h)t$ denotes instantaneous arrival rate of the two flows at time t, while the term $-C$ denotes 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 $rtt_l < \sqrt{\frac{2Q}{\alpha_l/rtt_l+\alpha_h/rtt_h}}(1-\sqrt{\delta})$.

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

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

Inequality (4) 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 (4) for $rtt_l = rtt_h$ and $\alpha_l \neq \alpha_h$.

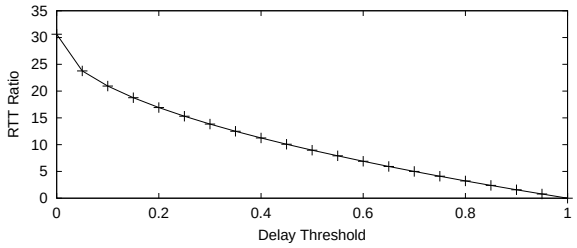


Fig. 6. Relationship between the RTT Ratio n and Threshold δ

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.

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.

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.

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

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

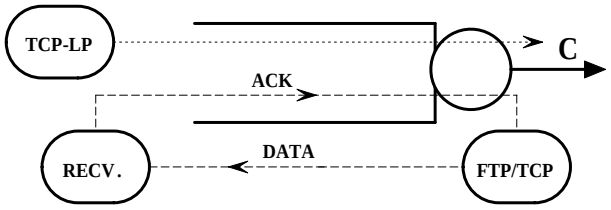


Fig. 7. Scenario with Reverse ACK Traffic

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.

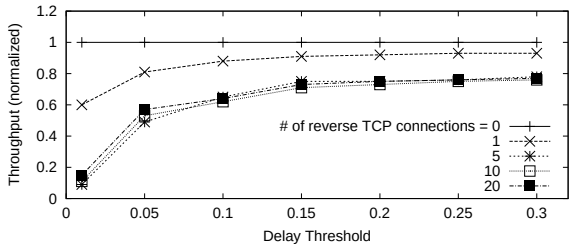


Fig. 8. Throughput vs. Threshold δ

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. [5]

3) Inference Time-out itt*:* Finally, a similar trade-off 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.

[5]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.

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.

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.)

We simulate TCP-LP/ECN by modifying the implementation of RED [12] 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. 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. 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.

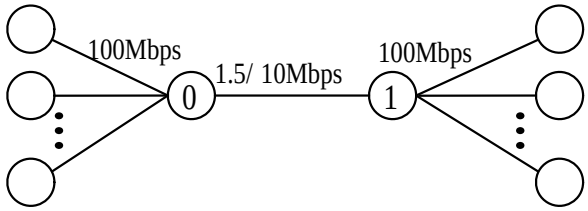


Fig. 9. Single Bottleneck Simulation Scenario

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 [13]. 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 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 [14]. 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. Experimental Results

We now use simulation to evaluate the performance of TCP-LP in a variety of scenarios. 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.

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.

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

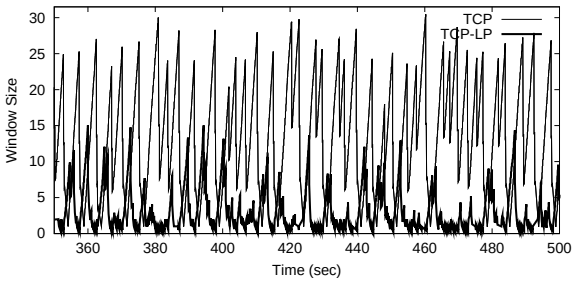


Fig. 10. TCP and TCP-LP's Congestion Window

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.

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. For comparison, we also depict the normalized average available bandwidth curve.

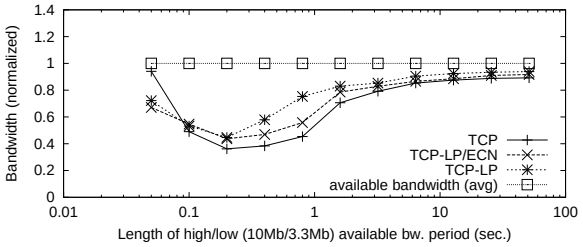


Fig. 11. Utilized Available Bandwidth vs. Square Wave Period

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.

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

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

its window accordingly. However, TCP-LP overcomes this problem with a larger number of multiplexed flows.

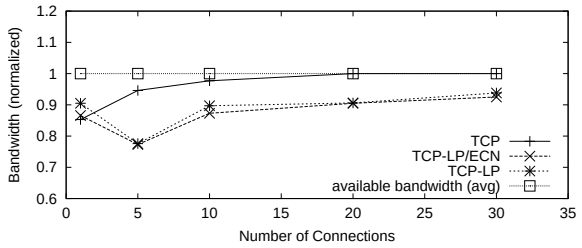


Fig. 12. Utilized Available Bandwidth vs. Number of Flows

3) Fairness: Here we study fairness among TCP-LP flows using Jain's fairness index [15]. The index, always between 0 and 1, is 1 if all flow throughputs are the same. 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.

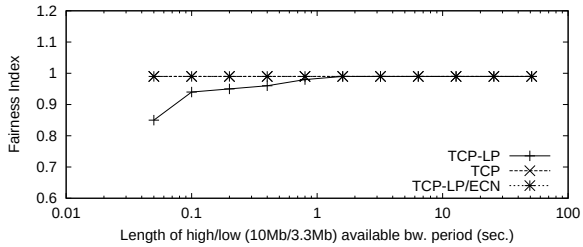


Fig. 13. Fairness Index vs. Square Wave Period

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 overestimates 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.

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.

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. 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.

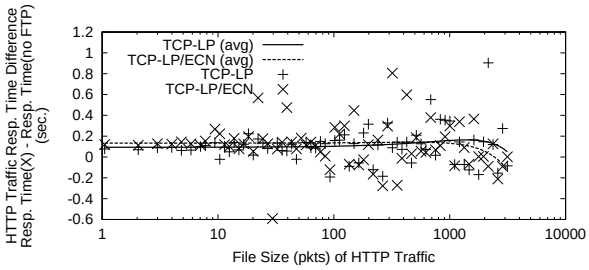


Fig. 14. Resp. Time Diff. (sec.) vs. File Size (kB) for HTTP Traffic

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.

Observe that use of TCP-LP for bulk data transfer reduces the web traffic response times by approximately 80% compared to TCP bulk 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

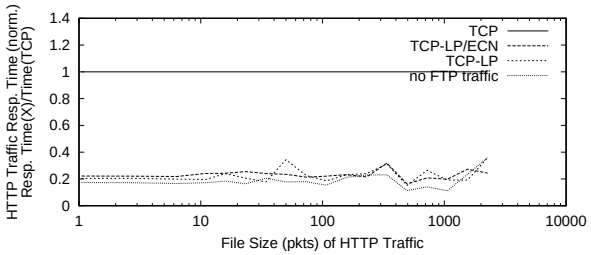


Fig. 15. Norm. Resp. Time vs. File Size (kB) for HTTP Traffic

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.

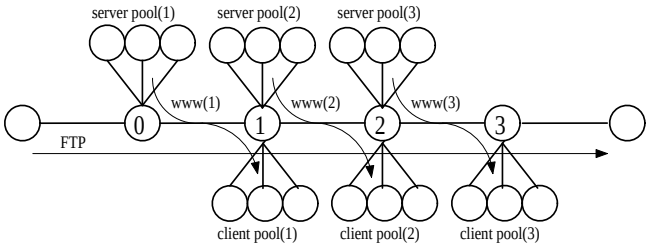


Fig. 16. First Topology for Multiple Bottlenecks

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.

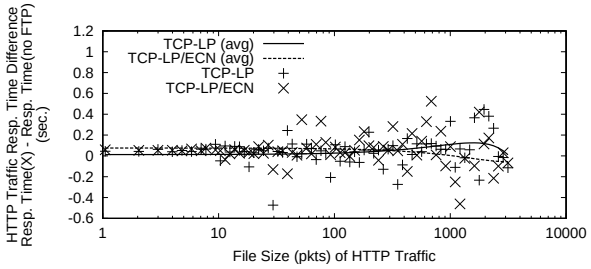


Fig. 17. Resp. Time Diff. (sec.) vs. File Size (kB) for HTTP Traffic

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 a 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.

2) Impact of High vs. Low Priority 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.

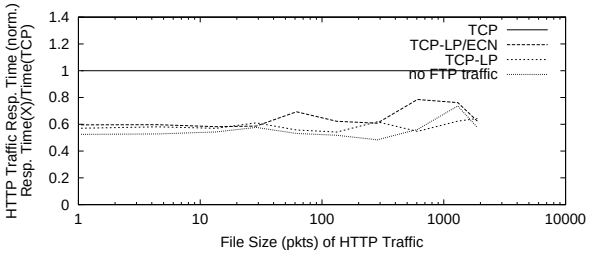


Fig. 18. Norm. Resp. Time vs. File Size (kB) for HTTP Traffic

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 [16].

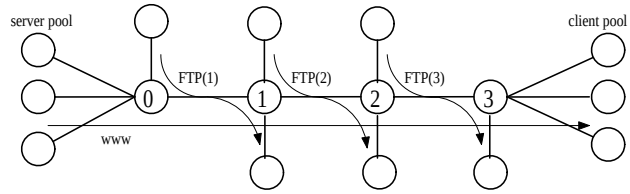


Fig. 19. Second Topology for Multiple Bottlenecks

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 improve 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 left unchanged.

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.

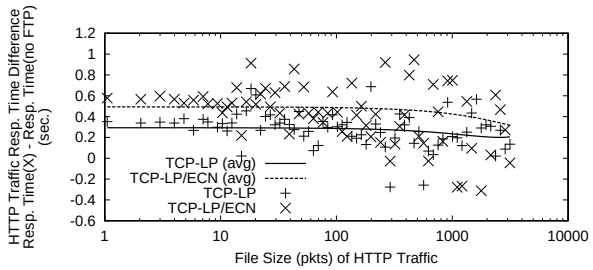


Fig. 20. Resp. Time Diff. (sec.) vs. File Size (kB) for HTTP Traffic

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. 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%.

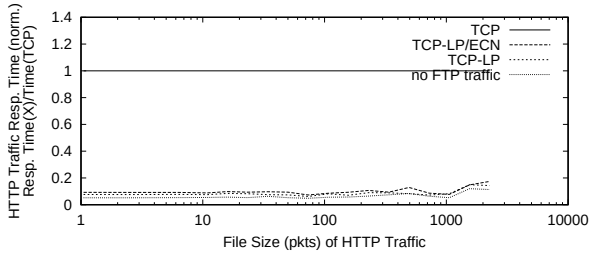


Fig. 21. Norm. Resp. Time vs. File Size (kB) for HTTP Traffic

VI. Related Work

While no protocols other than TCP-LP 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 [15], Wang *et al.'s* TCP/Dual [17], Brakmo *et al.'s* TCP/Vegas [18] 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. In contrast, TCP-LP uses *one-way* delay measurements vs. round-trip delays. Moreover, 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.* [19] 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.

Second, TCP-LP uses early congestion indication (earlier than TCP) as a basis for achieving class differentiation. Clark and Feng [2] 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.

Next, TCP-LP relates to adaptive bandwidth allocation schemes that aim to minimize file-transmission times using file-size-based service differentiation. Guo and Matta [20] 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 [21] 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.

Finally, 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 [6] for a thorough review of such algorithms). For example, Ribeiro *et al.* [4] and Alouf *et al.* [5] 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 [6] 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.

VII. 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. We performed an extensive set of *ns-2* simulations and showed that 1) TCP-LP is largely non-intrusive to TCP traffic 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 the 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. A linux implementation of TCP-LP is available at

http://www.ece.rice.edu/networks/TCP-LP. In future work, we plan to validate the above findings using experiments in a controlled network testbed as well as on the Internet.

ACKNOWLEDGMENTS

We would like to thank Sally Floyd for helpful comments on TCP-LP, and Liang Guo for providing valuable pieces of *ns* code.

REFERENCES

[1] S. Blake et al., "An architecture for differentiated services," 1998, Internet RFC 2475.

[2] D. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, Aug. 1998.

[3] 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, Aug. 2000.

[4] 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.

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

[6] 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.

[7] M. Vojnovic, J. 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, Mar. 2000.

[8] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, Aug. 1995.

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

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

[11] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Comm. Review*, vol. 24, no. 5, pp. 10–23, 1994.

[12] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[13] 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, Aug. 2001.

[14] 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 *Proceedings of ACM SIGCOMM '99*, Vancouver, British Columbia, Sept. 1999.

[15] R. Jain, "A delay based approach for congestion avoidance in interconnected heterogeneous computer networks," *ACM Computer Comm. Review*, vol. 19, no. 5, pp. 56–71, Oct. 1989.

[16] 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.

[17] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm," *ACM Computer Comm. Review*, vol. 22, no. 2, pp. 9–16, Apr. 1992.

[18] L. Brakmo and L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[19] 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.

[20] L. Guo and I. Matta, "The war between mice and elephants," in *Proceedings of IEEE ICNP '01*, Riverside, CA, Nov. 2001.

[21] 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.