

# **A Report on Some Developments in TCP Congestion Control Mechanisms**

By  
Sridevi Polavaram  
Instructor: Dr. Huang Yih  
SPR'04  
CS 756  
George Mason University

# Contents

1. *Motivation*
2. *Related Work*
  - 2.1 *End-to-End approaches*
  - 2.2 *Router-based approaches*
3. *TCP-Friendly protocols*
4. *TCP-Non intrusive protocols*
  - 4.1 *TCP-LP: Problem formulation & design goals*
  - 4.2 *TCP-LP: Approximation of Real Model*
  - 4.3 *Method of estimating early one-way packet delays*
  - 4.4 *Measuring one-way packet delays using time-stamp option*
  - 4.5 *TCP-LP: Congestion avoidance policy*
  - 4.6 *Parameter settings*
5. *Compare and contrast TCP-LP and Sync-TCP*
  - 5.1 *Design goals*
  - 5.2 *Similarities*
  - 5.3 *Congestion detection and avoidance policies*
  - 5.4 *Congestion reaction mechanism of Sync-TCP*
  - 5.5 *Congestion reaction mechanism of TCP-LP*
6. *Performance Evaluation of TCP-LP and Sync-TCP*
  - 6.1 *Performance Evaluation of Sync-TCP*
  - 6.2 *Performance Evaluation of TCP-LP*
  - 6.3 *Differences*
7. *Conclusion*
8. *References*

Several changes have been proposed over the last few years to TCP's congestion control mechanisms. The changes to TCP include modification of basic TCP congestion control algorithms for Congestion detection (Slow start), Congestion avoidance (AIMD) and Congestion recovery (Fast retransmit & Fast recovery) phases. These changes to TCP are designed to avoid unnecessary retransmit timeouts, to correct unnecessary fast retransmits resulting from reordered or delayed packets, and to assist the development of mechanisms that decrease the responsiveness of the web connections. All these modifications claim to have improved throughput, efficient bandwidth utilization and better performance & responsiveness over the existing base-line TCP/Reno protocol. Of course, extensive simulation results for e.g., ns-2 has been provided to prove the claims. Some of the protocols have also been implemented on the Internet (e.g., TCP Vegas). However, a close examination of these protocols adaptiveness provided some interesting insights. The changes in the network (routers) include the ECN and Active Queue Management techniques, whereas the end-point control mechanisms include TCP-LP: an incrementally deployable algorithm, that modifies the congestion avoidance scheme of the basic TCP and helps speed up the existing best-effort service while itself being a non-intrusive and low-priority protocol.

## 1. Motivation

This report aims to investigate the simple ways that would result in drastic improvement in the existing congestion control mechanisms in terms of responsiveness, bandwidth utilization, non-intrusiveness, TCP-friendliness, TCP-fairness and performance issues.

Primarily, the end-system hosts take the responsibility to detect packet losses and interpret them as indicators of congestion. Despite the success of TCP Reno, concerns have been raised about the pure end-to-end approaches to congestion avoidance. As a result, router-based congestion control mechanisms such as active queue management (AQM) and early congestion notification (ECN) have been proposed and deployed. These router-based mechanisms give best performance by sending early congestion notifications messages to the end-hosts, as they are in the best position to know when congestion is occurring. Drawbacks to using AQM methods include the complexity involved and the need to change routers in the networks.

This report investigates about protocols that have common ended goals but different perspectives

- (1) 'TCP-LP: A Distributed Algorithm for Low Priority Data Transfer'.
- (2) 'Sync-TCP: Delay-Based Early Congestion Detection and Adaptation: Impact on web performance'.
- (3) TCP/Vegas and RED mechanism
- (4) A brief overview of TCP-friendly protocols is presented.

## 2. Related Work

There have been two main approaches to detecting congestion before router buffers overflow: end-to-end methods and router-based mechanisms.

End-to-end methods are focused on making changes to TCP Reno. Most of these approaches try to detect and react to congestion earlier than TCP Reno (*i.e.*, before packet loss occurs) by monitoring the network using end-to-end measurements.

Router-based mechanisms, such as AQM, make changes to the routers so that they notify senders when congestion is occurring but before packets are dropped.

### 2.1 End-to-End Approaches

TCP/Vegas:

Vegas have mainly showed that it can achieve 40%-70% better throughput than Reno. However, this throughput is achieved not by aggressive retransmissions, but of efficient use of the available bandwidth. Vegas proposed three important techniques to improve throughput and reduce losses.

(1) New Retransmission Mechanism: Vegas estimate the RTT and variance using a fine-grained timer. Vegas read and record the system clock each time a segment is sent. This helps to estimate when to retransmit a lost packet. Also Vegas only decreases the congestion window if the retransmitted segment was previously sent after the last decrease. Whereas, in Reno it is possible to decrease the congestion window more than once for losses that occurred during one RTT interval.

(2) Congestion avoidance mechanism: The simple idea that Vegas exploits is that number of bytes in transit is directly proportional to the expected throughput, and therefore, as the window size increases the bytes in transit increases resulting in increased throughput of the connection. The goal of Vegas is to maintain the “right” amount of extra data in the network. Vegas congestion avoidance actions are based on changes in the estimated amount of extra data in the network, and not only on the dropped packets.

Expected throughput =  $\text{windowSize} / \text{BaseRTT}$  BaseRTT is set to the minimum of all measured round trip times. WindowSize is the size of the current congestion window

(3) Modified Slow-Start Mechanism: Vegas uses rate control during slow-start, using a rate defined by the current window size and the BaseRTT.

### 2.2 Router-based approaches

Random Early Detection: Random Early Detection (RED) is an AQM mechanism that seeks to reduce the long-term average queue length in routers. In RED, as each packet arrives, routers compute a weighted average queue length that is used to determine when to notify endsystems of incipient congestion. “Marking” a packet performs congestion notification in RED.

While TCP/Vegas also use delay-based congestion control in an effort to increase TCP throughput due to 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.

### 3. TCP-Friendly protocol

The algorithm for TCP congestion control is the main reason we are still using the Internet successfully today, despite largely unpredictable user access patterns and despite resource bottlenecks and limitations. Without TCP congestion control, the Internet could have become history a long time ago. Therefore, there is very necessity that every protocol adapt TCP-Friendliness issue.

Non-TCP flows are defined as TCP-Friendly when “their long-term throughput does not exceed the throughput of a conformant TCP connection under the same conditions”.

Many protocols have been proposed that basically modify the parameters like round-trip time, retransmission timeout value, segment size, and packet loss rate. All these protocols have the common goal of providing a TCP-Friendliness can be broadly classified under the following categories:

- Window-based vs Rate-based
- Unicast vs Multicast
- Single-rate vs Multi-rate
- End-to-End vs Router-supported

Detail description of each of these categories is beyond the scope of this report. However, an overview provides the significance of each of these categories with example protocols.

Window-based vs Rate-based: Algorithms that used window-based category use a congestion window at the sender or at the receiver(s) to ensure TCP-friendliness.

Algorithms that dynamically adapt the transmission rate according to the network feedback mechanism belong to Rate-based protocols.

Example: Rate Adaptation Protocol (RAP) is a simple AIMD scheme for unicast flows, where every data packet is acknowledged by the receiver. RAP achieves rates similar to, but does not take time-outs into consideration.

TCP Emulation At Receivers (TEAR) adapts both window-based and rate-based congestion control mechanisms. The receiver also maintains congestion window, calculates the fair receiving rate and sends that to the sender, which adjusts the sending rate accordingly.

Unicast vs Multicast: when talking about the congestion control mechanisms, we generally deal with the unicast protocols. However, designing a multicast protocol is more difficult than the unicast. Since they should ideally scale to large receiver sets and be able to cope with the heterogeneous network conditions at the receivers.

Example: Random Listening Algorithms (RLA) extends TCP SACK by adding some enhancements for multicast. The sender saves smoothed round-trip time for each sender and the measured congestion probability.

Receiver-driven Layered Multicast (RLM) is used for transmitting video is an example of layered multicast transmission. The sender splits the video into several layers. A receiver starts receiving by subscribing to the first layer. When the receiver does not experience any congestion in the form of packet loss for a certain period of time, it subscribes to the next layer. When the receiver experiences packet loss, it unsubscribes from the highest layer it is currently receiving. However, this protocol does not focus on TCP-friendliness but on how to provide each receiver with the best possible video quality. The mechanism of adding/dropping a single layer based on the packet loss is not TCP-friendly and can result in unfair distribution of bandwidth.

Single-rate vs Multi-rate: In single-rate schemes, data is sent to all receivers at the same rate. This limits the scalability of the mechanism. Whereas, multi-rate congestion control protocols allow for a more flexible allocation of bandwidth along the different network paths. Such schemes scale better to large receiver sets where increased heterogeneity among receivers is expected.

Example: a typical approach to multi-rate congestion control is to use layered mutlicast. Receiver-driven Layered Congestion (RLC)

End-to-end vs Router-supported: Many of the TCP-friendly schemes proposed are designed for best-effort IP networks that do not provide any additional router mechanisms. In end-to-end congestion control mechanisms the sender/receiver adjust the window size or transmission rate according to the network feedback. In the router-based approach, the routers employ some congestion control schemes instead of the sender and receiver hosts.

## 4. TCP Non-intrusive protocols

### 4.1 TCP-LP: Problem Formulation & Design Goals

The basic mechanism unique to TCP-LP congestion control is use of one-way packet delays for congestion indications. The design goals of TCP-LP aim for being non-intrusive to co-existing TCP flows by utilizing only the excess bandwidth available. In effect, TCP-LP provides a two-class service prioritization congestion control mechanism, where TCP flows have the first priority and TCP-LP flows have the second priority.

### 4.2 TCP-LP: Approximation of the Reference model

The objective of TCP-LP is to design end-host-based transport protocol that emulates the low-priority service. To achieve this goal, it is necessary for TCP-LP to infer congestion earlier than TCP. This is achieved by the measuring the one-way packet delays as early indicators for TCP-LP, as compared to packet drops used by TCP.

### 4.3 Method of estimating early one-way packet delays

Denote  $d_i$  as the one-way delay of the packet with sequence number  $i$ , and  $d_{\min}$  and  $d_{\max}$  is an estimate of the minimum and maximum queuing delays. The smoothed one-way delay is computed using the exponentially weighted moving average as:

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

An early congestion indication occurs when the following equation satisfies

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

where  $0 < \delta < 1$  denotes the threshold parameter.

From equation (2)  $d_{\min}$  represents the minimum-observed one-way packet delay, which is close to the propagation delay on the link.  $d_{\max}$  represents the maximum-observed one-way packet delay, which is caused by total queuing delay along the path between the sender and receiver. Therefore, equation (2) satisfies when the smoothed delay exceeds estimated propagation delay and queuing delays. The threshold value  $\delta$  is an important parameter that affects the performance of TCP-LP. The minimum and maximum queuing delays are calculated during the slow-start phase of the connection, and used after the first packet loss (i.e.) during the congestion avoidance phase.

Initially, queuing delay of a path can be estimated by taking the difference between the minimum-observed one-way packet delay and the most recent one-way packet delay. The minimum-observed one-way packet delay from a flow is assumed to be close to the propagation delay on the link. Any additional delay is expected to be caused by queuing.

Since the estimation of the queuing delay involves the difference between one-way packet delays (rather than the exact values of one-way packet delays), synchronized clocks are not required.

One-way packet delays can more accurately reflect queuing delay caused by network congestion than round-trip times (RTTs). Using RTTs there is no way to accurately estimate the forward path queuing delay. If there is an increase in the RTT, the sender cannot distinguish between the cause being congestion on the forward (data) path, congestion on the reverse (ACK) path, or both.

#### 4.4 Measuring one-way packet delays using Time-stamp option

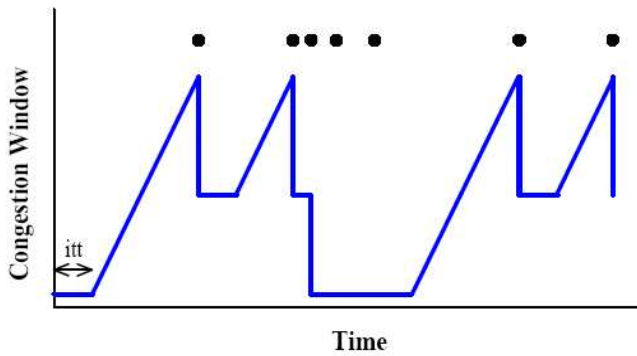
Delay measurement is measured by using timestamp option in the TCP header. End systems can determine the one-way packet delays between them by exchanging timestamps. For TCP-LP, an optional TCP header is added, based on the RFC 1323. This TCP header option includes the one-way packet delay in microseconds calculated for the last segment received (*one-way packet delay*), the current segment's sending time (*timestamp*), and the sending time of the last segment received (*echo reply*). When a receiver sees any segment with the TCP-LP timestamp option, it calculates the segment's one-way packet delay by subtracting the time the packet was sent from the time the packet was received. The one-way packet delay is then inserted into the next segment the receiver sends to the sender (generally an ACK). Upon receiving the ACK, the sender can estimate the current queuing delay by subtracting its minimum-observed one-way packet delay from the one-way packet delay present in the ACK.

#### 4.5 TCP-LP: Congestion Avoidance Policy

Major objective of TCP-LP is to detect the early congestion and reacting immediately to it. Since,  $d_{\min}$  and  $d_{\max}$  have already been estimated before hand. Any indication of congestion, TCP-LP has the first chance to react to it from equation (2), rather than TCP flows. Therefore, there is an additional phase in TCP-LP called

Inference phase. During this phase, the TCP-LP halts its aggressiveness to utilize the excess bandwidth. An inference time-out timer is started during this phase to make sure that the congestion indication is not caused due to any cross-traffic.

The sender host receives an ACK packet from the receiver consisting of the one-way packet delay. The sender estimates the minimum and maximum delay values from these ACK packets. TCP-LP observes the changes in forward-path queuing delay to detect congestion. After receiving the first ACK indicating the congestion, the TCP-LP cuts the window by half and starts the inference timer. During this period the window size is not increased, before the timer goes off, if there is a second indication, then the window size is throttled to one packet. Otherwise, the window again enters additive-increase congestion avoidance phase. The below figure shows the TCP-LP congestion avoidance policy, the dots represent the congestion inferences.



#### 4.6 TCP-LP: Parameter settings

Delay smoothing  $\gamma$  in equation (1) is set to  $1/8$ , the typical value for computing round-trip time for TCP.

Delay threshold  $\delta$  plays an important role in the TCP-LP responsiveness. Small  $\delta$  values increases the throughput of TCP-LP flows, but however setting  $\delta$  to very small values causes TCP-LP to react to the false early congestion indications caused by bursts of cross-traffic. Thus  $\delta$  must be balanced enough to increase TCP-LPs responsiveness while avoiding false early congestion indications. The below figure shows the relationship between the threshold  $\delta$  and the RTT ratio between the TCP-LP and TCP flows.

Similar trade-off between congestion-responsiveness and throughput-aggressiveness holds for the inference time-out timer.

## 5. Compare and Contrast TCP-LP and Sync-TCP

### 5.1 Design Goals

Both the protocols make modifications to the TCP/Reno in the congestion avoidance and recovery phases. However, TCP-LP main objective is to achieve better throughput and performance while itself being a low-priority end-point congestion control protocol. Whereas, the Sync-TCP, is a delay-based early congestion detection and reaction mechanism that uses one-way transit times to estimate forward-path queuing delay.

Though, the statement of the two protocols looks different both have the same point to target, (i.e) measuring one-way packet delays and thus achieving early congestion

notification messages that help react faster than the basic TCP flows. And both the protocols are end-point control protocols, which doesn't need AQM support from the routers.

### 5.2 Similarities

Let's observe the similarities of these two protocols in a tabular form:

TCP-LP	Sync-TCP
(1) Use of one-way packet delays for TCP-transparency	(1) Provide early congestion detection mechanisms using one-way packet delays
(2) Largely non-intrusive to TCP	(2) Performance of TCP is improved by addition of Sync-TCP
(3) Distributed algorithm that is dynamically adaptable	(3) Sync-TCP is incrementally deployable algorithm
(4) Decrease the HTTP-responsiveness of web connections by 90%	(4) Better throughput and HTTP response performance than TCP
(5) Based on the TCP-Reno	(5) Based on TCP-Reno
(6) Computes average queuing delay using weighted moving average with smoothing factor 1/8	(6) Same as TCP-LP
(7) Measures one-way packet delays using the Time-stamp option	(7) Measures one-way transit time by placing Timestamps in TCP headers
(8) Sender and receiver clocks need not be synchronized	(8) Robust to clock-drifts
(9) The information about the available bandwidth is known without having to probe the network	(9) Monitoring queuing delays allows connections to detect when congestion is subsiding and take advantage of the excess capacity in the network by increasing their sending rates more aggressively than Reno.

### 5.3 Congestion detection and avoidance policies of TCP-LP and Sync-TCP

The Sync-TCP approach:

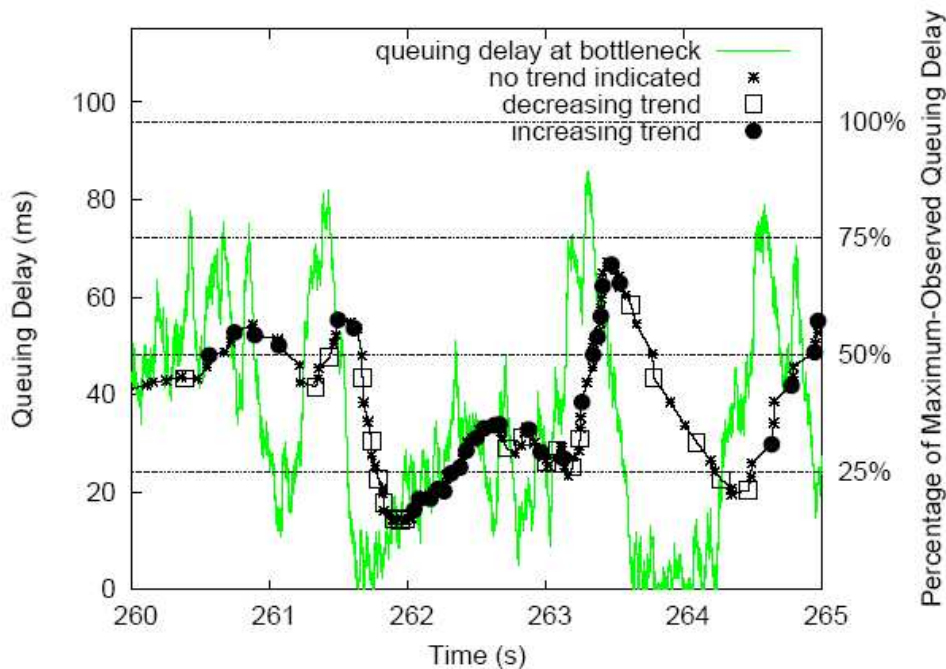
- Uses forward path queuing delays as early congestion indications
- These queuing delays are averaged and the trend of the average delay is estimated
- The estimated trend and average are given to the congestion reaction mechanism

- Computes the weighted average of the estimated queuing delay with smoothing factor 1/8 and compares the resultant magnitude on the scale of four parts 0%-25%, 25%-50%, 50%-75%, 75%-100%
- The trend of the delay is observed using nine-sample trend analysis as follows:  
Sync-TCP first gathers nine average queuing delay samples and splits them into three groups of three samples, in the order of their computation. The median,  $m_i$ , of each of the three groups is computed. Since there are only three medians, the trend is determined to be increasing if  $m_0 < m_2$ . Likewise, the trend is determined to be decreasing if  $m_0 > m_2$ . Sync-TCP computes a new trend every three ACKs by replacing the three oldest samples with the three newest samples for trend analysis. Due to the nine-sample trend analysis, no early congestion detection will occur for connections that receive fewer than nine ACKs.

Each time an ACK is received, the congestion detection mechanism in Sync-TCP reports one of two outcomes:

- 1) Not enough samples have been gathered to compute the trend, or
- 2) The region where the average queuing delay lies in relation to the maximum-observed queuing delay and the direction of the trend.

Below figure shows the Sync-TCP congestion detection algorithm at work.



When there are 20 long-lived FTP connections in each direction over one bottleneck link. The average queuing delay tracks the actual queuing delay relatively well. The lag between the actual queuing delay and the average queuing delay is caused by both the time between the packet experiencing the delay and its ACK returning (delayed ACKs

were used) and the average weighting factor of 1/8. Above figure demonstrates that the trend analysis algorithm can perform well in tracking the trend of the average queuing delay.

#### 5.4 Congestion Reaction mechanism of Sync-TCP

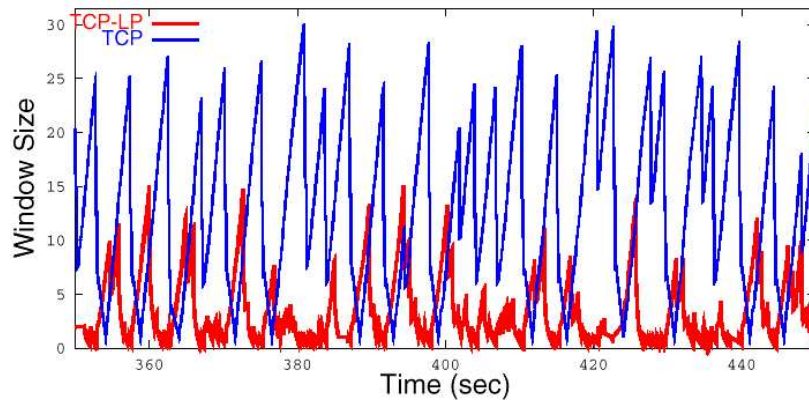
Sync-TCP uses additive increase multiplicative decrease (AIMD) congestion window adjustment but adjusts AIMD's  $\alpha$  and  $\beta$  parameters differently than Reno as follows:

Additive increase is defined as  $w(t+1) = \alpha + w(t)$ , where  $w(t)$  is the size of the current congestion window in segments at time  $t$  and the time unit is one RTT. During Reno congestion avoidance,  $\alpha = 1$ . For every ACK received, the congestion window is increased by  $1/w(t)$ , which results in an increase of at most one segment in one RTT. Multiplicative decrease is defined as  $w(t+1) = \beta w(t)$ . In Reno,  $\beta = 0.5$ .

Following AIMD, when *cwnd* is to be increased, Sync-TCP incrementally increases *cwnd* for every ACK returned, and when *cwnd* is to be decreased, Sync-TCP makes the decrease immediately.

#### 5.5 Congestion Reaction mechanism of TCP-LP

One of the claims made by TCP-LP is that TCP-LP can be able to obtain excess bandwidth even in the presence of greedy TCP flows. Considering 10 FTP flows in TCP and just 1 TCP-LP flow, the below figure depicts the nature of TCP-LP.



While the TCP *cwnd* is oscillating between 0 and 30, the *cwnd* of TCP-LP, also tracks the TCP's oscillations and increases its own size when TCP's window decreases, and via early congestion inference, TCP-LP enters inference phase when the TCP flow's window reaches its maximum 30 packets.

## 6. Performance Evaluation of TCP-LP and Sync-TCP

### 6.1 Performance Evaluation of Sync-TCP

Network-level metrics: Average packet loss percentage at each congested router, the average queue size at each congested router, the average throughput, the average goodput (data arriving to the web clients), and the average link utilization at the link closest to the forward-path HTTP clients.

Application-level metrics: HTTP response times and the average goodput per response. The application-level metrics are measured only for those flows that have completed their request-response pair. Whereas, the network-level metrics are measured for all traffic, including flows that still has data in the network when the experiment ends.

HTTP response time cumulative distribution functions (CDFs) are used as the main metric for evaluating the performance of HTTP traffic.

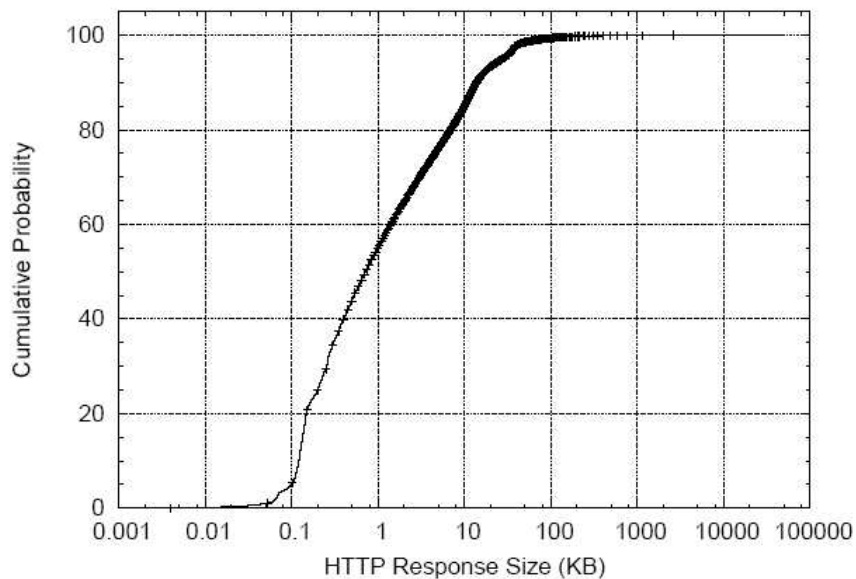


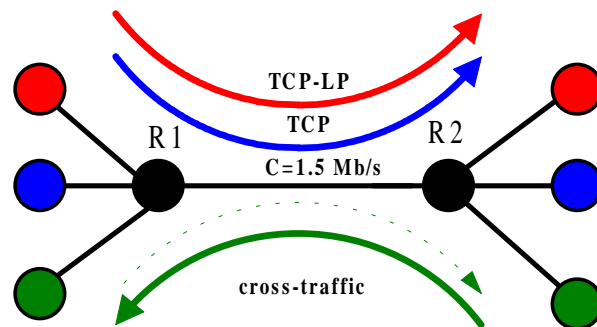
Fig. 4. HTTP Response Sizes

Varying loads of 75%, 90%, and 95% are offered on the available bandwidth of 10 Mbps. At every level, three protocols are simulated under isolation. TCP/Reno, SACK-ARED-ECN and Sync-TCP are subjected to the two-way background HTTP traffic, that's generated by 5 server pools. All the performance metrics are measured against the HTTP cumulative distributed probability.

	TCP Reno	SACK ARED ECN	Sync TCP	100 Mbps
<b>% packet drops at R0</b>	3.9	13.9	2.0	0
<b>avg queue size (pckts)</b>	64.4	19.4	53.9	0
<b>avg completed rspsz (B)</b>	6960	6353	7202	7256
<b>goodput / response (kbps)</b>	89.4	83.9	99.6	226.8
<b>median rsptime (ms)</b>	430	810	390	230
<b>mean rsptime (ms)</b>	1268.1	4654.7	943.6	406.1

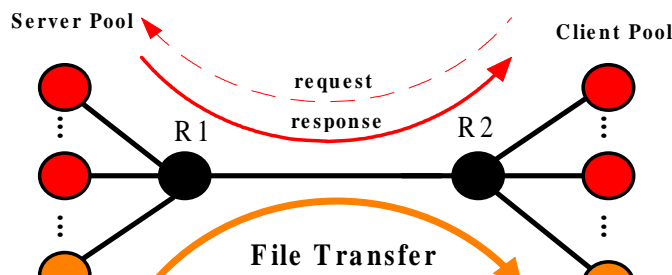
Experimental results show that Sync-TCP doesn't have any cross-over point with TCP/Reno or SACK-ARED-ECN at 75% load, and continues perform better under higher loads 90% and 95%. The TABLE above, show the statistics of the performance metrics at 90% load.

### 6.2 Performance evaluation of TCP-LP



When there is low aggregation, (i.e) one TCP flow, one TCP-LP flow, TCP cannot attain 1.5 Mb/s throughput due to the presence of cross-traffic, this situation definitely leaves some excess bandwidth for TCP-LP. It's known from the experiments that when there is no TCP-LP, the throughput of the forward TCP flow is 49.7%. With the presence of TCP-LP the throughput is 49.3%, indicating that TCP-LP achieves nearly perfect TCP transparency while achieving 7.3% throughput.

During high aggregation with short-lived flows, when Bulk FTP flows use TCP-LP and TCP flow containing web traffic, it's known from the experiments that web response times improved 3-5 times and the FTP throughput for TCP is 58.2%, TCP-LP is 55.1%



Therefore, from the figures obtained from experiments it can be seen that with the presence of TCP-LP flows, the short-lived web connections response times improves and also long-lived bulk FTP transfers achieve nearly same throughput as when transmitted using TCP.

The TCP-LP are non-intrusive to TCP flows even if their round-trip times are much larger.

Also, multiple TCP-LP flows utilize better bandwidth with the co-existing TCP flows than a single one.

### 6.3 Differences

Differences between TCP-LP and Sync-TCP

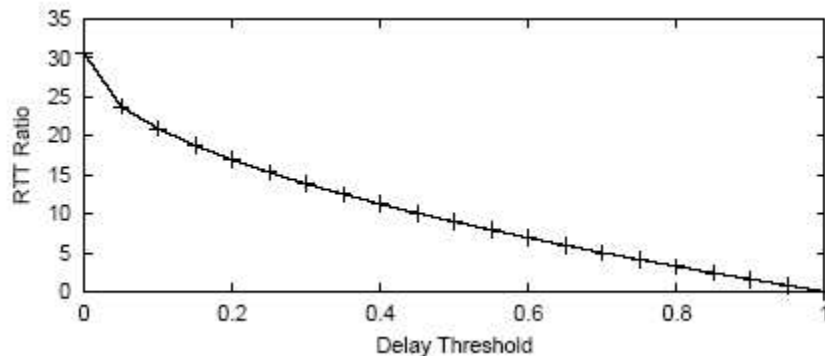
TCP-LP	Sync-TCP
(1) Provided an algorithm for congestion avoidance by adding an additional inference phase. This inference phase backs-off for utmost two RTT's when there is congestion. This reaction of TCP-LP on detection congestion is shown by the experimental results	(1) Provided an empirical analysis for the congestion detection and reaction mechanism. Modifying $\alpha$ and $\beta$ AIMD parameters helps Sync-TCP react quickly to the congestion inferences.
(2) The response time improved nearly by 90% by applying TCP-LP to single bottleneck topology.	(2) The overall improved response times are achieved by applying Sync-TCP to only 5% of the connections.
(3) Estimates the average queuing delay from weighted moving average and the minimum and maximum observed delays values during connection lifetime.	(3) Estimated the average queuing delay from weighted moving average and the trend analysis mechanism, to determine if the delay is increasing or decreasing
(4) Simulation is done on single and multiple bottleneck links of 1.5 Mbps capacity and the propagation delay on each link is 20 ms.	(4) Simulation is done on 10 Mbps link and each link has 1ms of propagation delay
(5) For HTTP traffic generation, the network topology has 3 server and 3 client pools. Each pool is a cloud of 3 servers/clients	(6) For HTTP traffic generation, the network topology consists of 5 HTTP server clouds, 5 HTTP client clouds and 6 router/delay boxes. Each cloud consists of 5 servers/clients.

## 7. Conclusion

Both TCP-LP and Sync-TCP have the same key mechanism of detecting early congestion indications by using the one-way packet delays and thus achieving a better throughput performance. Both the protocols are aiming at developing an end-point congestion control mechanism, which tries to approximate the router-based ARED-ECN congestion control algorithms. The experimental results of both the protocols have

showed that their respective newly proposed protocols perform better than the TCP/Reno and TCP/ARED-ECN.

However, the interesting observation is TCP-LP has proposed a new low-priority end-point congestion control algorithm. By adding a new inference phase in the TCP congestion avoidance policy. The important parameters like delay threshold  $\delta$ , inference time-out timer values make a drastic difference in the performance of the protocol. One of the graphs below shows how varying the delay threshold  $\delta$  effects the response time of the TCP-LP when sharing the bandwidth with other TCP flows, when TCP-LP and TCP have different RTTs



From the figure it can be seen that the point (0.4, 11.25) shows that with delay threshold  $\delta=4$ , 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.

The performance charts show the behavior of the protocol by increasing and decreasing these values in a certain range. Also, the design objectives of TCP-LP stated that

- (a) TCP-LP allows fairness among the multiple flows
- (b) TCP-LP being able to achieve the transparency inspite of having longer RTT values than the co-existing TCP flows

For (b) they have provided a queuing model solution with some mathematical equations to estimate the RTTs of TCP and TCP-LP.

But no formal explanation or mathematical proof has been provided for these features. It looks like these design features are inherently existing for the reason being TCP-LP is basically modification of TCP/Reno.

The Sync-TCP protocol on the other hand, has provided an empirical derivation of how the protocol achieves better performance than the TCP/Reno or TCP-ARED-ECN. It has adapted the trend analysis algorithm to detect the one-way delay. The experimental results or theoretical evaluation is much clearer for the reader.

## 8. References

- [1] TCP-LP: A Distributed Algorithm for Low Priority Data Transfer. *Aleksandar Kuzmanovic and Edward W. Knightly*
- [2] Delay-Based Early Congestion Detection and Adaptation: Impact on web performance. *Michele C. Weigle, Kevin Jeffay, and F. Donelson Smith*
- [3] The Effectiveness of End-to-End Congestion Control Mechanisms  
*J. Bolliger, U. Hengartner and Th. Gross*
- [4] Survey on Fairness Issues in TCP Congestion Control Mechanisms. *Go Hasegawa and Masayuki Murata*
- [5] TCP Vegas: New Techniques for Congestion Detection and Avoidance. *Lawrence S. Brakmo Sean W. O'Malley Larry L. Peterson*
- [6] A Survey on TCP-Friendly Congestion Control. *J. Widmer, R. Denda, and M. Mauve*
- [7] Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Kevin Fall and Sally Floyd*
- [8] Random early detection gateways for congestion avoidance. *Sally Floyd and Van Jacobson.*