Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control

Sumitha Bhandarkar, Saurabh Jain and A. L. Narasimha Reddy
Dept. of Electrical Engineering,
Texas A & M University
{sumitha,saurabhj,reddy}@ee.tamu.edu

***Abstract*— In this paper, we propose Layered TCP (LTCP for short), a simple layering technique for the congestion window response of TCP to make it more scalable in highspeed, high RTT networks. LTCP is a two dimensional congestion control framework - the macroscopic control uses the concept of layering to quickly and efficiently make use of the available bandwidth whereas microscopic control extends the existing AIMD algorithms of TCP to determine the per-ack behavior. We provide the general intuition and framework for the LTCP protocol modifications in this paper. Then, using a simple design, we illustrate the effectiveness of using layering for improving the efficiency, without sacrificing the convergence properties of TCP. We assess the RTT unfairness with the chosen design and show that by using a simple RTT compensation factor, the RTT unfairness can be assured to be no worse than that of unmodified TCP. Evaluation of the specified design is based on analyses and ns-2 based simulations. We show that LTCP has promising convergence properties, is about an order of magnitude faster than TCP in utilizing high bandwidth links, employs few parameters and is easy to understand. The choice of parameters can be influenced to reduce the RTT unfairness, compared to TCP or other highspeed solutions. The flexible framework opens a whole class of design options for improving the performance of TCP in highspeed networks.**

I. Introduction

The ever-increasing availability of network bandwidth and the deployment of these high-speed links for high-delay transatlantic communication have posed a serious challenge for the AIMD algorithms used for congestion control in TCP. Over the past few years, several solutions have been put forth for solving the problem. These solution can be classified into four main categories - a) Tuning the network stack b) Opening parallel TCP connections between the end hosts c) Modifications to the TCP congestion control d) Modifications to the network infrastructure or use of non-TCP transport protocol.

The traditional solution to improve the performance of TCP on high-capacity networks has been to tune some of the TCP parameters. [2], [3], [4] and [5] are some of the examples. Tuning the stack improves the performance significantly and is best used in conjunction with the other solutions mentioned below.

Significant research effort has focussed on using *network striping* for mitigating the problem. Irrespective of whether the application opens parallel connections ([6], [7], [8], [9], [10]) or the TCP flow behaves as a collection of several virtual flows ([11], [12], [13]), the problem of choosing the *optimal* number of flows to maximise the performance without effecting the fairness properties is non-trivial, limiting the scope of these solutions.

The third category of research for improving the performance of TCP in highspeed networks has been to modify the congestion response function of TCP itself. HighSpeed TCP [14], Scalable TCP [15], FAST TCP [16], Bic-TCP [17] and H-TCP [18] are some of the examples in this area.

Several other schemes that go beyond modifications only to TCP and use either UDP or rely on support from the network infrastructure have also been proposed. XCP[19], Tsunami[20], RBUDP[21], SABUL/UDT [22] and GTP[23] are some of the examples.

In this paper, we propose the Layered TCP scheme which modifies the congestion response function of TCP at the sender-side and requires no additional support from the network infrastructure or the receivers. We focus on improving the TCP performance on high-bandwidth links and provide a simple extension for controlling the unfairness when flows with different RTTs are multiplexed over the high-bandwidth link. This scheme can be thought of as an emulation of multiple flows at the transport level, with the key contribution *that the number of virtual flows adapt to the dynamic network conditions*. Layering schemes for probing the available bandwidth have been studied earlier in the context of multicast and video transfer, for example [24], [25]. LTCP, in contrast to this earlier body of work, uses layering *within the congestion control algorithm of TCP with per-ack window adaptation* to provide efficient bandwidth probing in high bandwidth links and compensation mechanism for controlling the fairness between flows with different RTTs.

The rest of the paper is organised as follows - In Section II we provide the general framework for the LTCP scheme. Section III discusses one possible design choice and presents analyses pertaining to it. Results of the evaluation of the chosen design using ns-2 simulations are presented in Section IV. We conclude the paper in Section V by summarising our experiences and taking a look at the future work.

II. Layered TCP : The Framework

The congestion window response of the LTCP protocol is defined in two dimensions - (a) At the macroscopic level,

This work is supported in part by a grant from The Texas Higher Education Board, by an NSF grant ANI-0087372 and by Intel Corp.

LTCP uses the concept of layering. If congestion is not observed over a period of time, then number of layers is increased. (b) At the microscopic level, it extends the existing AIMD algorithms of TCP to determine the per-ack behavior. When operating at a higher layer, a flow increases its congestion window faster than when operating at a lower layer. In this section, we present the intuition for the layering framework.

The primary goal behind designing the LTCP protocol is to make the congestion response function scale in highspeed networks under the following constraints: (a) the LTCP flows should be fair to each other when they have same RTT (b) the unfairness of LTCP flows with different RTTs should be no worse than the unfairness between unmodified TCP flows with similar RTTs (c) the LTCP flows should be fair to TCP flows when the window is below a predefined threshold W_T. The threshold W_T defines the regime in which LTCP is friendly to standard implementations of TCP. We choose a value of 50 packets for W_T. This value has been chosen to maintain proportional fairness between LTCP and unmodified TCP flows in slow networks which do not require the window scale option [26] to be turned on. Arguments aimed at keeping new protocols fair to TCP below a predefined window threshold, while allowing more aggressive behavior beyond the threshold have been put forth in [14], [15] as well.

All new LTCP connections start with one layer and behave in all respects the same as TCP. The congestion window response is modified only if the congestion window increases beyond the threshold W_T. Just like the standard implementations of TCP, the LTCP protocol is ack-clocked and the congestion window of an LTCP flow changes with each incoming ack. However, an LTCP flow increases the congestion window more aggressively than the standard implementation of TCP depending on the layer at which it is operating. When operating at layer K, the LTCP protocol increases the congestion window as if it were emulating K virtual flows. That is, the congestion window is increased by $K/cwnd$ for each incoming ack, or equivalently, it is increased by K on the successful receipt of one window of acknowledgements. This is similar to the increase behaviour explored in [11].

Layers, on the other hand, are added if congestion is not observed over an extended period of time. To do this, a simple layering scheme is used. Suppose, each layer K is associated with a step-size δ_K. When the current congestion window exceeds the window corresponding to the last addition of a layer (W_K) by the step-size δ_K, a new layer is added. Thus,

$$W_1 = 0, \quad W_2 = W_1 + \delta_1, \quad ..... \quad W_K = W_{K-1} + \delta_{K-1} \quad (1)$$

and the number of layers $= K$, when $W_K \leq W < W_{K+1}$. Figure 1 shows this graphically. The step size δ_K associated with the layer K should be chosen such that convergence is possible when several flows (with similar RTT) share the bandwidth. Consider the simple case when the link is to be shared by two LTCP flows with same RTT. Say, the flow that started earlier operates at a higher layer K_1 (with a larger window) compared to the later-starting flow operating at a smaller layer K_2 (with the smaller window). In the absence of network congestion, the first flow increases the congestion window by K_1 packets per RTT, whereas the second flow increases by K_2 packets per RTT. In order to ensure that the first flow does not continue to increase at a rate faster than the second flow, it is essential that the first flow adds layers slower than the second flow. Thus, if δ_{K_1} is the stepsize associated with layer K_1 and δ_{K_2} is the stepsize associated with layer K_2, then

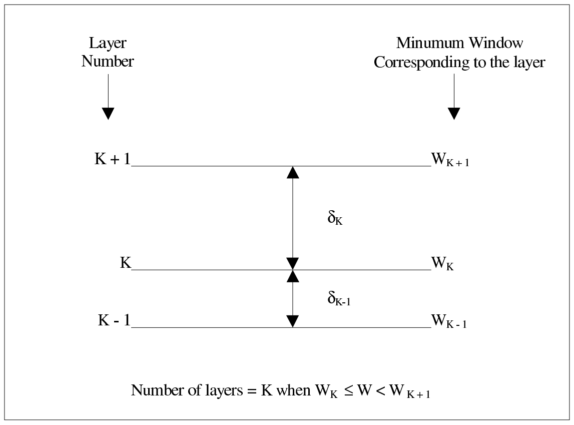
Layer Number
Minumum Window Corresponding to the layer
K + 1
W_{K+1}
δ_K
K
W_K
δ_{K-1}
K - 1
W_{K-1}
Number of layers = K when W_K ≤ W < W_{K+1}

Fig. 1. Graphical Perspective of Layers in LTCP

$$\frac{\delta_{K_1}}{K_1} > \frac{\delta_{K_2}}{K_2} \quad (2)$$

when $K_1 > K_2$, for all values of $K_1, K_2 \geq 2$.

The design of the decrease behavior is guided by the following reasoning - in order for two flows with same RTT, starting at different times to converge, the time taken by the larger flow to regain the bandwidth it gave up after a congestion event should be larger than the recovery time of the smaller flow. Suppose the two flows are operating at layers K_1 and K_2 ($K_1 > K_2$), and WR_1 and WR_2 is the window reduction of each flow upon a packet loss. After the packet drop, suppose the flows operate at layers $K_1^{'}$ and $K_2^{'}$ respectively. Then, the flows take $\frac{WR_1}{K_1^{'}}$ and $\frac{WR_2}{K_2^{'}}$ RTTs respectively to regain the lost bandwidth. From the above reasoning, this gives us -

$$\frac{WR_1}{K_1^{'}} > \frac{WR_2}{K_2^{'}} \quad (3)$$

The window reduction can be chosen proportional to the current window size or be based on the layer at which the flow operates. If the latter is chosen, then care must be taken to ensure convergence when two flows operate at the same layer but at different window sizes.

This framework provides a simple, yet scalable design for the congestion response function of TCP for the congestion avoidance phase in highspeed networks. The congestion window response in slow start is not modified, allowing the architecture to evolve with experimental slowstart algorithms such as [27]. At the end of slowstart the number of layers to operate at can easily be determined based on the window size. The key factor for the framework is to determine an appropriate relationships for the step size δ_k (or equivalently, the window size W_k at which the layer transitions occur) and the window reduction that satisfy the conditions in Equations 2 and 3.

III. A Design Choice

Several different design options are possible for choosing the appropriate relationship for the stepsize δ and the window reduction WR. In this section, we present one possible design. We support this design with extensive analysis to understand the protocol behavior.

For our design, we first choose the decrease behavior. Since we want to keep the design as close to that of TCP as possible, we choose a multiplicative decrease. However, the window reduction is based on a factor of β such that -

$$WR = \beta * W \tag{4}$$

Based on this choice for the decrease behavior we determine the appropriate increase behavior such that the conditions in Equation 2 and Equation 3 are satisfied. To provide an intuition for choice of the increase behavior, we start by considering Eq. 3. Also, in order to allow smooth layer transitions, we stipulate that after a window reduction due to a packet loss, atmost one layer can be dropped i.e., a flow operating at layer K before the packet loss should operate at layer K or $(K-1)$ after the window reduction. Based on this stipulation, there are four possible cases - (1) $K_1^{'} = K_1, K_2^{'} = K_2$, (2) $K_1^{'} = (K_1 - 1), K_2^{'} = K_2$, (3) $K_1^{'} = K_1, K_2^{'} = (K_2 - 1)$, and (4) $K_1^{'} = (K_1 - 1), K_2^{'} = (K_2 - 1)$. It is most difficult to maintain the convergence properties, when the larger flow does not reduce a layer but the smaller flow does, ie, $K_1^{'} = K_1, K_2^{'} = (K_2 - 1)$. With this worst case situation, Eq. 3 can be written as -

$$\frac{WR_1}{K_1} > \frac{WR_2}{(K_2 - 1)} \tag{5}$$

If this inequality is maintained for adjacent layers, we can show by simple extension, that it can be maintained for all other layers. So consider $K_1 = K, K_2 = (K-1)$. Then, the above inequality is

$$\frac{WR_1}{K} > \frac{WR_2}{K-2} \tag{6}$$

Suppose, the window for flow 1 is $W^{'}$ when the packet loss occurs and the window of flow 2 is $W^{''}$ then, substituting Eq. 4 in the above equation, we have,

$$\frac{W^{'}}{K} > \frac{W^{''}}{K-2} \Rightarrow W^{'} > \frac{K}{K-2} W^{''}$$

In order for the worst case behavior ($K_1^{'} = K, K_2^{'} = (K-2)$) to occur, the window $W^{'}$ could be close to the transition to the layer $(K+1)$ and the window $W^{''}$ could have recently transitioned into layer $(K-1)$. In order to get the estimate of the worst case we substitute these values in the above equation to get -

$$W_{K+1} > \frac{K}{K-2} W_{K-1} \tag{7}$$

Based on this, we conservatively choose the increase behavior to be

$$W_K = \frac{K+1}{K-2} W_{K-1} \tag{8}$$

Note that alternate choices are possible. This is essentially a tradeoff between efficiently utilizing the bandwidth and ensuring convergence between multiple flows with similar RTT sharing the same link. While it is essential to choose the relationship between W_K and W_{K-1} such that the condition in equation 7 is satisfied to ensure convergence, a very conservative choice would make the protocol slow in increasing the layers and hence less efficient in utilizing the bandwidth.

With this choice, since layering starts at $W_2 = W_T$, we have -

$$W_K = \frac{K(K+1)(K-1)}{6} W_T \tag{9}$$

By definition, $\delta_K = W_{K+1} - W_K$ and hence we have,

$$\delta_K = \frac{K(K+1)}{2} W_T \tag{10}$$

By simple substitution, we can show that the inequality in Eq. 2 is satisfied. Also, since this scheme was designed with the worst case for the inequality in Eq. 3, that condition is satisfied as well, when two competing flows are at adjacent layers. The result for adjacent layers can then be easily extrapolated for non-adjacent layers. It can also be shown that when two flows operate at the same layer, the inequality in Eq. 3 is satisfied.

A. Choice of β

The above analysis is hinged on the stipulation that after a window reduction due to packet drop, at most one layer is dropped. In order to ensure this, we have to choose the parameter β appropriately. The worst case for this situation occurs when the flow has just added the layer K and the window $W = W_K + \Delta$, when the packet drop occurs. In order to ensure that the flow does not go from layer K to $(K-2)$ after the packet drop, we need to ensure that

$$\beta W_K < \delta_{K-1} \tag{11}$$

On simple substitution, this yields,

$$\beta < \frac{3}{K+1} \tag{12}$$

Thus, β should be chosen such that the above equation is satisfied. We use a value of $\beta = 0.15$ in this paper. The reason for this choice of β is explained in the next section.

With this design choice, LTCP retains AIMD behavior. At each layer K, LTCP increases the window additively by K, and when a packet drop occurs, the congestion window is reduced multiplicatively by a factor of β.

B. Analysis

As explained earlier, the primary goal for designing the LTCP protocol is to be able to utilize available link bandwidth aggressively in highspeed networks. In this section we provide some quantitative analysis for the chosen design.

1) Time to claim bandwidth and Packet Recovery time: Suppose the maximum window size corresponding to the available throughput is W_K. Then, time to increase the window to W_K, in the absence of congestion, can be obtained as the sum of the time to transition from layer 1 to 2, 2 to 3 and so on until layer K. In other words, the time to increase the window to W_K is -

$$T(\delta_1) + T(\delta_2) + ... + T(\delta_{K-2}) + T(\delta_{K-1})$$

where $T(\delta_K)$ is the time (in RTTs) for increasing the window from layer K to $(K+1)$. When the flow operates at layer K, to reach to the next layer, it has to increase the window by δ_K and the rate of increase is K per RTT. Thus $T(\delta_K)$ is given by $\frac{\delta_K}{K}$ RTTs. Substituting this in the above equation and doing the summation we find that the time to reach a window size of W_K is

$$T(\delta_1) + \frac{(K-2)(K+3)}{4} W_T \tag{13}$$

Note that the above analysis assumes that slowstart is terminated before layering starts.

Table in Fig. 2 shows the number of layers corresponding to the windowsize at layer transitions (W_K) with $W_T = 50$. For a 2.4Gbps link with an RTT of 150ms and packet size of 1500 bytes, the window size can grow to 30,000. The number of layers required to maintain full link utilization in this case is atleast $K = 15$. Based on this we choose $\beta = 0.15$ (corresponding to $K = 19$).

The table also shows the speedup in claiming bandwidth compared to TCP, for an LTCP flow with $W_T = 50$, with the assumption that slowstart is terminated when window = W_T. This column gives an idea of the number of virtual TCP flows emulated by an LTCP flow. For instance, a flow that evolves to layer 15, behaves similar to establishing 10 parallel flows at the beginning of the connection.

K	W_K	Speedup in Claiming Bandwidth	Speedup in Packet Loss Recovery Time
1	0	-	-
2	50	1.00	1.00
3	200	2.00	6.67
4	500	2.57	10.00
5	1000	3.17	13.33
6	1750	3.78	16.67
7	2800	4.40	20.00
8	4200	5.03	23.33
9	6000	5.67	26.67
10	8250	6.31	30.00
11	11000	6.95	33.33
12	14300	7.60	36.67
13	18200	8.25	40.00
14	22750	8.90	43.33
15	28000	9.56	46.67
16	34000	10.21	50.00
17	40800	10.87	53.33
18	48450	11.52	56.67
19	57000	12.18	60.00
20	66500	12.84	63.33

Fig. 2. Comparison of LTCP (with W_T = 50 and β = 0.15) to TCP

An LTCP flow with window size W will reduce the congestion window by βW. It then starts to increase the congestion window at the rate of atleast $(K-1)$ packets per RTT (since we stipulate that a packet drop results in the reduction of atmost one layer). The upper bound on the packet loss recovery time for LTCP then, is $\frac{\beta W}{(K-1)}$. In case of TCP, upon a packet drop, the window is reduced by half, and after the drop the rate of increase is 1 per RTT. Thus, the packet recovery time is $W/2$. The last column of Table in Fig.2 shows the speed up in packet recovery time for LTCP with $\beta = 0.15$ compared to TCP. Based on the conservative estimate that a layer reduction occurs after a packet drop, the speed up in the packet recovery time of LTCP compared to TCP is a factor of $3.33 * (K-1)$.

2) Throughput Analysis: In order to understand the relationship between throughput of an LTCP flow and the drop probability p of the link, we present the following analysis. Fig. 3 shows the steady state behavior of the congestion window of an LTCP flow with a uniform loss probability model. Suppose the number of layers at steady state is K and the link drop probability is p. Let W'' and W' represent the congestion window just before and just after a packet drop respectively. On a packet loss the congestion window is reduced by $\beta W''$. Suppose the flow operates at layer K' after the packet drop. Then, for each RTT after the loss, the congestion window is increased at the rate of K' until the window reaches the value W'', when the next packet drop occurs. The window behavior of the LTCP flow, in general, will look like Fig. 3 at steady state.

With this model, the time between two successive losses, say T_D, will be $\frac{\beta W''}{K'}$ RTTs or $\frac{\beta W''}{K'} * RTT$ seconds. The number of packets sent between two successive losses, say N_D, is given by the area of the shaded region in Fig. 3. This can be shown to be -

$$N_D \simeq \frac{\beta (W'')^2}{K'}(1 - \frac{\beta}{2}) \tag{14}$$

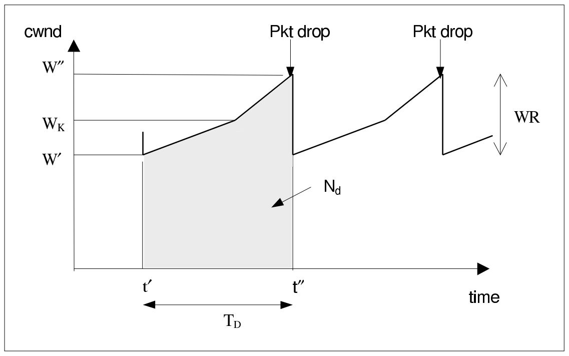

Fig. 3. Analysis of Steady State Behavior

The throughput of such an LTCP flow can be computed as $\frac{N_D}{T_D}$. That is,

$$BW = \frac{W''}{RTT}(1 - \frac{\beta}{2}) \tag{15}$$

The number of packets sent between two losses N_D is nothing but $\frac{1}{p}$. By substituting this in Eq. 14, and solving for W'' we have,

$$W'' = \sqrt{\frac{K'}{\beta(1 - \frac{\beta}{2})p}} \tag{16}$$

Substituting in Eq. 15 we have

$$BW = \frac{\sqrt{\frac{K'}{\beta}(1 - \frac{\beta}{2})}}{RTT\sqrt{p}} \tag{17}$$

Again if we consider the example above of the 2.4Gbps link with an RTT of 150ms and packet size of 1500 bytes, the window size can grow to 30,000. From, Fig. 2 we see that this window size corresponds to a layersize of $K = 15$. Substituting this value in the above equation, we notice that for the 2.4Gbps link mentioned above with $\beta = 0.15$, LTCP offers an improvement of a factor of about 8 for the achievable single-flow throughput compared to TCP.

3) RTT Unfairness: In this section we assess the RTT unfairness of LTCP under the assumptions of random loss model as well as synchronized loss models. In [29], for a random loss model the probability of the packet loss λ is given by -

$$\lambda \propto \frac{A(w, RTT)}{A(w, RTT) + B(w, RTT)} \tag{18}$$

where $A(w, RTT)$ and $B(w, RTT)$ are the window increase and decrease functions respectively.

For LTCP $A(w, RTT) = K/w$ and $B(w, RTT) = \beta W$. Substituting these values in the above equation and approximating $K \propto W^{1/3}$, we can calculate the loss rate λ as -

$$\lambda \propto \frac{1}{(1 + \beta W_s^{5/3})} \tag{19}$$

where W_s is the statistical equilibrium window.

It is clear from the above equation that the two LTCP flows experiencing the same loss probability, will have the same equilibrium window size, regardless of the round trip time. However, throughput at the equilibrium point becomes inversely proportional to its round trip time since the average transmission rate r_s and is given by W_s/RTT.

The loss probability for TCP with similar assumptions is given by:

$$\lambda \propto \frac{1}{(1 + 0.5W_s)} \tag{20}$$

The equilibrium window size of the TCP flow does not depend on the RTT either. Therefore, for random losses the RTT dependence of window of an LTCP flow is same as TCP. Thus LTCP has window-oriented fairness similar to TCP and will not perform worse than TCP in case of random losses.

Because of the nature of current deployment of high bandwidth networks, it is likely that the degree of multiplexing will be small and as such, an assumption of synchronized loss model may be more appropriate. So we present here the details of the analysis with the synchronised loss model as well.

Following a similar analysis in [17], for synchronized losses, suppose the time between two drops is t. For a flow i with round trip time RTT_i and probability of loss p_i, the average window size is

$$W_i = \frac{\frac{1}{p_i}}{\frac{t}{RTT_i}} = \frac{RTT_i}{tp_i} \tag{21}$$

since the flow will send $\frac{1}{p_i}$ packets between two consecutive drop events and the number of RTTs between the two consecutive loss events $\frac{t}{RTT_i}$.

From Eq. 17, we have the bandwidth of an LTCP flow to be

$$BW = \frac{W_i}{RTT_i} \quad = \frac{\sqrt{\frac{K_i'}{\beta}(1 - \frac{\beta}{2})}}{RTT_i\sqrt{p_i}}$$

$$\Rightarrow p_i = \frac{K_i' C}{W_i^2} \quad where \quad C = \frac{1}{\beta}(1 - \frac{\beta}{2}) \tag{22}$$

By substituting the above in Eq. 21 and simplifying we get,

$$W_i = \frac{tK_i' C}{RTT_i} \tag{23}$$

From Eq. 9 we see that the number of layers K is related to the window size W based on the relationship $K \propto W^{\frac{1}{3}}$. By substituting this relationship in the above equation we have,

$$W_i \propto (\frac{tC}{RTT_i})^{\frac{3}{2}} \tag{24}$$

When the RTT unfairness is defined as the throughput ratio of two flows in terms of their RTT ratios, the RTT unfairness for LTCP is -

$$\frac{(\frac{W_1}{RTT_1})(1 - p_1)}{(\frac{W_2}{RTT_2})(1 - p_2)} \simeq \frac{\frac{W_1}{RTT_1}}{\frac{W_2}{RTT_2}} \propto (\frac{RTT_2}{RTT_1})^{\frac{5}{2}} \tag{25}$$

(since $p << 1$).

This is slightly worse than the RTT unfairness of unmodified TCP shown to be proportional to $(\frac{RTT_2}{RTT_1})^2$ in [17].

In case of LTCP, a flow with larger window size operates at a higher layer and increases the window at the rate corresponding to that layer. Therefore, it is relatively more aggressive than a flow with a smaller window size which operates at a lower layer and hence increases the window at a lower rate. This, coupled with the difference in RTTs could make the RTT unfairness of LTCP worse than that of TCP. When the losses are random, the larger flow experiences more packet drops than the smaller flow. The resulting difference in the number of loss events reduces the effect of RTT unfairness of LTCP. However, with synchronous loss model, when equal number of loss events may be observed by different flows, the RTT unfairness could be worse than that of TCP.

In order to compensate for this dependence of aggressiveness on RTT, we introduce the RTT compensation factor K_R and modify the per-ack behavior such that, an LTCP flow at layer K will increase the congestion window at the rate of $K_R * K$ for the successful receipt of one window of acknowledgements. The RTT compensation factor is set to 1 for the base RTT of 10ms. For larger RTTs, K_R is made proportional to the ratio $RTT/(base\_RTT)$. This uniformly scales up the rate at which the higher RTT flow increase its window. Since the RTT compensation factor K_R is constant for a given RTT and multiplicative in nature, it does not alter the analyses in the earlier discussion for flows with same RTT. However, the throughput equation for LTCP and the RTT unfairness can now be re-written as -

$$BW = \frac{\sqrt{K_i' * K_{R_i} * C}}{RTT_i\sqrt{p_i}}$$

$$\frac{\frac{W_1}{RTT_1}}{\frac{W_2}{RTT_2}} \propto (\frac{RTT_2}{RTT_1})^{\frac{5}{2}}(\frac{K_{R_1}}{K_{R_2}})^{\frac{3}{2}} \tag{26}$$

The above equation shows that by choosing K_R appropriately, the RTT unfairness of LTCP flows can be controlled. For instance, choosing $K_R \propto RTT^{\frac{1}{3}}$, the RTT unfairness of the LTCP protocol will be similar to the AIMD scheme used in TCP. By choosing $K_R \propto RTT$, the effect of RTT on the scheme can be entirely eliminated and the LTCP protocol behaves like a rate controlled scheme independent of the RTT. By choosing an intermediate value such as, $K_R \propto RTT^{\frac{1}{2}}$, we can reduce the RTT unfairness of LTCP in comparison to TCP.

In general, suppose we choose, K_R proportional to RTT^{α}, where α is a constant. After a window reduction WR, suppose a flow operates at layer $K^{'}$. When RTT compensation is used it takes $\frac{WR}{K_R * K^{'}}$ RTTs or $\frac{WR * RTT}{K_R * K^{'}}$secs to regain the lost bandwidth. Suppose two flows with different RTTs are competing for the available bandwidth, Equations 3 can be re-written as

$$\frac{WR_1 * RTT_1}{K_{R1} * K_1^{'}} > \frac{WR_2 * RTT_2}{K_{R2} * K_2^{'}}$$

Substituting the value of WR and further solving it, we have,

$$\Rightarrow (\frac{RTT_1}{RTT_2})^{\alpha - 1} < \frac{(K^{'} + 1)}{K^{'}}$$
$$\Rightarrow (\alpha - 1) < log(1 + \frac{1}{K^{'}}) \quad \Rightarrow \alpha <= 1 \tag{27}$$

The above equation has been derived by assuming a worst case RTT ratio of 10 while taking the logarithm and is specific to the design choice mentioned at the beginning of this section. An alternate design choice can give a different dependence of K_R on RTT. In order to ensure that the flows do not become aggressive as queues build up, we make K_R is dependent only on the propagation delay of the link. In our experiments we use the lowest measured RTT sample for choosing the value of K_R.

C. Alternate Designs:

In this section we presented one possible design for LTCP and provided the relevant analysis to understand the protocol behavior with this design. This is by no means the only possible or the best possible design choice. The aim of this design was to illustrate the effectiveness of using a simple concept like layering in the context of TCP congestion control to improve efficiency without sacrificing convergence properties. We are currently in the process of evaluating other designs. Details will be made available in a technical report.

IV. RESULTS

To evaluate the LTCP protocol, we conducted experiments on the ns-2 simulator. In this section we present some of the results. Fig. 4 shows the network topology used. The topology is a simple dumbbell network. The bottleneck link bandwidth is set to 1 Gbps unless otherwise specified. The links that connect the senders and the receivers to the router have a bandwidth of 2.4Gbps. The end-to-end RTT is set to 70ms, unless mentioned otherwise. The routers have the default queuesize of 6000 packets unless specified otherwise. DropTail queue management is used at the routers. The LTCP protocol is implemented by modifying the TCP/Sack1 agent. The unmodified TCP/Sack1 agent is used for TCP. The receiver advertised window is set to a large value to ensure that it does not interfere with the simulations. For the LTCP flows, the parameter W_T is set to 50 packets and the parameter β was set to 0.15. The traffic constitute of FTP transfer between the senders and receivers.

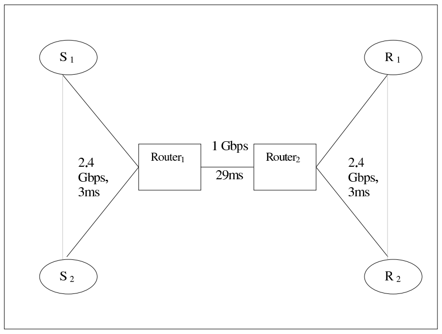

Fig. 4. Simulation Topology

1) Comparison of LTCP with TCP: Since LTCP uses adaptive layering, it is capable of increasing its window size to the optimal value much faster than TCP. Also, when a packet loss occurs, the window reduction of LTCP is not as drastic as TCP. As a result the window adaptation of LTCP is much more efficient in utilizing the link bandwidth in highspeed networks. Fig. 5 shows congestion window of LTCP in comparison with that of TCP, when the network consists of only one flow. As seen from the figure, the congestion window of the TCP flow takes over 600 seconds more than that of an LTCP flow to recover from a packet loss due to its drastic window reduction (factor of 1/2) followed by a conservative window increase (one per RTT). The table in Fig. 6 shows the comparison

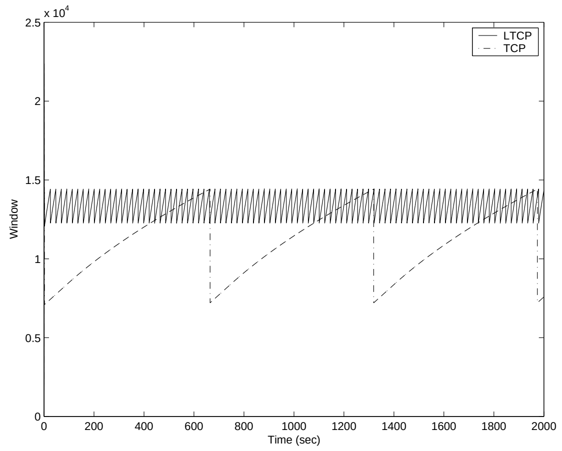

Fig. 5. Comparison Between LTCP and TCP Windows

between the long-term average steady state link utilization and average packet loss rates between a TCP flow and an LTCP flow for different link bandwidths. The throughput is calculated over a period of 2000 seconds after the flow reaches steady state. The buffersize at the router is set to the delay-bandwidth product. At large values of link bandwidth, the link utilization by the TCP flow is close to 75% , whereas the LTCP link utilization remians close to 96%. Since LTCP can operate close to the optimal value most of the time and keep the link utilization high, the congestion loss rate of the LTCP flow is larger than that of TCP, which owing to under-utilization of the link sees lower congestion losses.

Link Bandwidth	TCP		LTCP	
	Avg. Steady State Link Utilization (Mbps)	Packet Loss Rate (%)	Avg. Steady State Link Utilization (Mbps)	Packet Loss Rate (%)
10Mb	9.34	1.41E-02	9.62	1.33E-01
100Mb	96.15	2.08E-05	96.15	1.12E-03
1Gb	866.75	2.33E-06	961.54	4.49E-04
2.4Gb	1864.80	*	2307.69	2.01E-04

(* very low, could not be measured)

Fig. 6. Link Utilization and Packet Loss Rate for LTCP and TCP

2) Dynamic Link Sharing: In this experiment, we evaluate how LTCP flows respond to dynamically changing traffic conditions created by multiplexing of several flows starting and stopping at different times. One LTCP flow is started at time 0, and allowed to reach steady state. A new LTCP flow is then added at 150 seconds and lasts for 1250 seconds. A third LTCP flow is added at 350 seconds and lasts for 850 seconds. Finally, the fourth flow starts the transfer at 550 seconds and sends data for 450 seconds. All these flows share the same bottleneck link. Fig. 7 shows the throughput of each flow. From the graph we see that, when a new flow is started and the available link bandwidth on the bottleneck link decreases, the existing LTCP flows quickly give up bandwidth until all flows reach the fair utilization level. The per-flow link utilizations remain stable at this value until some of the flows stop sending traffic. When that happens, the remaining LTCP flows quickly ramp up the congestion window to reach the new fair sharing level, such that the link is fully utilized.

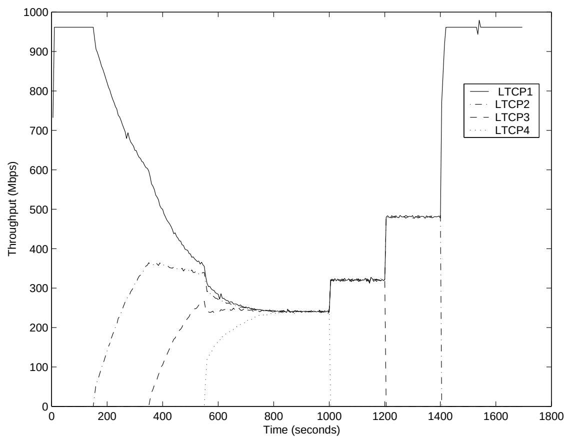

Fig. 7. Dynamic Link Sharing

3) Fairness Among multiple LTCP Flows: In this experiment, we evaluate the fairness of LTCP flows to each other. Different number of LTCP flows are started at the same time and the average per-flow bandwidth of each flow is noted. The table in Fig 8 shows that when the number of flows is varied, the maximum and the minimum per-flow throughputs remain close to the average, indicating that the per-flow throughput of each flow is close to the fair proportional share. This is verified by calculating the Fairness Index proposed by Jain et. al., in [28]. The Fairness Index being close to 1 shows that the LTCP flows share the available network bandwidth equitably.

No. of Flows	Avg. per-flow Throughput(Mbps)	Min. per-flow Throughput(Mbps)	Max. per-flow Throughput(Mbps)	Jain's Fairness Index
2	480.77	480.72	480.82	1.000
4	240.38	240.33	240.42	1.000
6	160.26	160.16	160.36	1.000
8	120.19	115.72	138.86	0.996
10	96.15	96.14	96.18	1.000

Fig. 8. Fairness Among LTCP Flows

4) Interaction with TCP: In this section, we study the effect of LTCP on regular TCP flows. It must be noted that the window response function of LTCP is *designed* to be more aggressive than TCP in high speed networks. So a single flow of TCP cannot compete with a single flow of LTCP and thus, for this simulation we compare the aggregate throughput of ten TCP flows to that of one LTCP flow at different link bandwidths. Also, to verify that an *established* LTCP flow gives up a share of its bandwidth to TCP flows, we first start the LTCP transfer and let it run for 300 seconds, so that it utilizes the link fully. At this point the TCP flows are started. The throughput is calculated for the flows after another 300 seconds. Fig.9 shows the results. As the link bandwidth increases, TCP flows become more inefficient in utilizing the available bandwidth, and as such, the percentage of the bandwidth used by TCP decreases. But, the aggregate throughput of TCP flows increases with higher link capacity, showing that inspite of the aggressive nature of LTCP congestion control, it still gives up a share of the bandwidth to competing TCP flows. For instance, on the 500Mbps link, the aggregate throughput of the TCP flows is 101.25Mbps but on the 1Gbps link it is 185.64 Mbps.

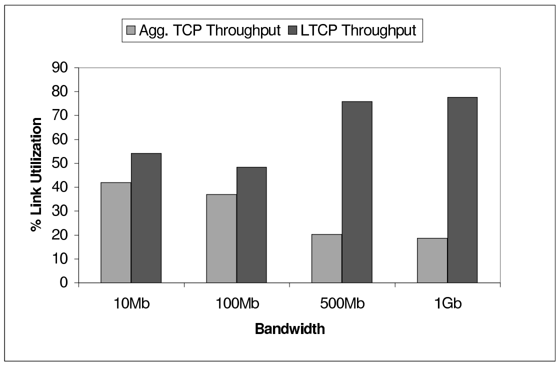

Fig. 9. Interaction of LTCP with TCP

5) RTT Unfairness: Table in Fig.10 shows the ratio of throughput of two flows with different RTTs sharing the same bottleneck link. The ratio of RTT of the two flows is set to 1, 2 and 3 for the three runs of the experiment, with the RTT of the shorter-delay flow being 40ms. For the computation of the RTT compensation factor K_R, a base RTT of 10ms is used.

It can be seen from the table that, the use of RTT compensation factor, reduces the RTT unfairness of LTCP. With an RTT compensation factor of $K_R \propto RTT^{\frac{1}{3}}$ LTCP shows RTT unfairness comparable to TCP while with RTT compensation

RTT Ratio	TCP	Unmodified LTCP	LTCP with $K_R \propto RTT^{1/3}$	LTCP with $K_R \propto RTT^{1/2}$
1	1.00	1.02	1.00	1.01
2	3.80	5.23	3.61	2.94
3	8.63	12.22	8.71	6.04

Fig. 10. RTT Unfairness for Different Values of K_R

factor of $K_R \propto RTT^{\frac{1}{2}}$ the RTT unfairness of LTCP in comparison to that of TCP is reduced.

Several other experiments were conducted verifying the effect of the RTT compensation factor on the link utilization, flow throughput, fairness and loss rates. However, due to lack of space, those results have not been included here. They will be made available in a technical report.

V. Conclusions and Future Work

In this paper we have proposed LTCP, a layered approach for modifying TCP for high-speed high-delay links. LTCP employs two dimensional control for adapting to available bandwidth. At the macroscopic level, LTCP uses the concept of layering to increase the congestion window when congestion is not observed over an extended period of time. Within a layer K, LTCP uses modified additive increase (by K per RTT) and remains ack-clocked. The layered architecture provides flexibility in choosing the sizes of the layers for achieving different goals. This paper explored one design option. For this design, the window reduction on a packet loss is chosen to be multiplicative.

We have shown through analysis and simulations that a single LTCP flow can adapt to nearly fully utilizing the link bandwidth. Other significant features of the chosen design are (a) it provides a significant speedup in claiming bandwidth and in packet loss recovery times compared to unmodified TCP (b) multiple flows share the available link capacity equitably (c) an RTT compensation can be used to ensure that the RTT unfairness is no worse than unmodified TCP (d) requires simple modifications to TCP's congestion response mechanisms and is controlled only by simple parameters - W_T, β and K_R.

We have also implemented LTCP in the Linux kernel and are currently comparing its performance with other schemes. We plan to characterize the traffic in high speed links to understand the level of multiplexing and the nature of the losses. Comparative third party evaluation of LTCP against other proposals is currently underway at SLAC, Stanford.

Our design is hinged on an early decision to use multiplicative decrease, and on the stipulation that atmost one layer is dropped after a congestion event. A number of other possibilities exist for alternate designs of the general LTCP approach. We plan to pursue these options in future.

References

[1] M. Mathis, J. Semske, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithms," *ACM Computer Communication Review, vol. 27(3),* July 1997.

[2] Jeffrey Semke, Jamshid Mahdavi, and Matthew Mathis, "Automatic TCP Buffer Tuning", *Proceedings of ACM SIGCOMM,* October 1998.

[3] Eric Weigle and Wu-chun Feng, "Dynamic Right-Sizing: a Simulation Study", *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN),* October 2001.

[4] Brian L. Tierney, Dan Gunter, Jason Lee, Martin Stoufer and Joseph B. Evans, "Enabling Network-Aware Applications", *10th IEEE International Symposium on High Performance Distributed Computing (HPDC),* August 2001.

[5] Tom Dunigan, Matt Mathis and Brian Tierney, "A TCP Tuning Daemon", *SuperComputing (SC)* November, 2002.

[6] Ostermann, S., Allman, M., and H. Kruse, "An Application-Level solution to TCP's Satellite Inefficiencies", *Workshop on Satellite-based Information Services (WOSBIS),* November, 1996.

[7] J. Lee, D. Gunter, B. Tierney, B, Allcock, J. Bester, J. Bresnahan and S. Tuecke, "Applied Techniques for High Bandwidth Data Transfers Across Wide Area Networks", *Proceedings of International Conference on Computing in High Energy and Nuclear Physics,* September 2001.

[8] C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC storage resource broker", *In Proc. CASCON'98 Conference,* Dec 1998.

[9] R. Long, L. E. Berman, L. Neve, G. Roy, and G. R. Thoma, "An application-level technique for faster transmission of large images on the Internet", *Proceedings of the SPIE: Multimedia Computing and Networking 1995,* Feb 1995.

[10] H. Sivakumar, S. Bailey and R. Grossman, "PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks", *Proceedings of Super Computing,* November 2000.

[11] Jon Crowcroft and Philippe Oechslin, "Differentiated End-to-End Internet Services using a Weighted Proportional Fair Sharing TCP", ACM CCR, vol. 28, no. 3, July 1998.

[12] Thomas Hacker, Brian Noble and Brian Athey, "Improving Throughput and Maintaining Fairness using Parallel TCP", *Proceedings of IEEE Infocom 2004,* March 2004.

[13] Hung-Yun Hsieh and Raghupathy Sivakumar, "pTCP: An End-to-End Transport Layer Protocol for Striped Connections", *Proceedings of 10th IEEE International Conference on Network Protocols,* November 2002.

[14] Sally Floyd, "HighSpeed TCP for Large Congestion Windows", *RFC 3649* December 2003.

[15] Tom Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks", *ACM Computer Communications Review,* April 2003.

[16] Cheng Jin, David X. Wei and Steven H. Low, "FAST TCP: motivation, architecture, algorithms, performance", *IEEE Infocom,* March 2004.

[17] Lisong Xu, Khaled Harfoush, and Injong Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks", To appear in *Proceedings of IEEE Infocom 2004,* March 2004.

[18] R. N. Shorten, D. J. Leith, J. Foy, and R. Kilduff, "Analysis and design of congestion control in synchronized communication networks", June 2003, submitted for publication.

[19] Dina Katabi, Mark Handley, and Chalrie Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", *Proceedings of ACM SIGCOMM 2002,* August 2002.

[20] README file of tsunami-2002-12-02 release. http://www.indiana.edu/ anml/anmlresearch.html

[21] Eric He, Jason Leigh, Oliver Yu and Thomas A. DeFanti, "Reliable Blast UDP : Predictable High Performance Bulk Data Transfer", *Proceedings of IEEE Cluster Computing,* September, 2002.

[22] H. Sivakumar, R. Grossman, M. Mazzucco, Y. Pan, and Q. Zhang, "Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks", to appear in *Journal of Supercomputing,* 2004.

[23] R.X. Wu and A.A. Chien, "GTP: Group Transport Protocol for Lambda-Grids", *4th IEEE/ACM International Symposium on Cluster Computing and the Grid,* April 2004.

[24] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast", *Proceedings of ACM SIGCOMM '96,* August 1996.

[25] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer" *Proceedings of IEEE Infocom '98,* March 1998.

[26] V. Jacobson, R. Braden and D. Borman, "TCP Extensions for High Performance", *RFC 1323,* May 1992.

[27] S. Floyd, "Limited Slow-Start for TCP with Large Congestion Windows", *RFC 3742,* March 2004.

[28] Dah-Ming Chiu and Raj Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Computer Networks and ISDN Systems, 17,* June 1989

[29] S. J. Golestani and K. K. Sabnani, "Fundamental observations on multicast congestion control in the Internet", In proceedings of *IEEE INFOCOM'99,* New York, NY, March 1999.