Internet Engineering Task Force
INTERNET DRAFT
draft-bhandarkar-ltcp-01.txt
Expires : February 2005

Sumitha Bhandarkar
Saurabh Jain
A. L. Narasimha Reddy
Texas A&M University

August 2004

LTCP: A Layering Technique for Improving the Performance of TCP in Highspeed Networks.

Status of this Memo

Abstract:

This document proposes Layered TCP (LTCP for short), a simple layering technique for the congestion window response of TCP to make it more scalable in highspeed 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. This document provides the general intuition and framework for the LTCP protocol modifications. Using a simple design, the effectiveness of using layering for improving the efficiency without sacrificing the convergence properties of TCP, is illustrated. The chosen design is evaluated using mathematical analysis, ns-2 based simulations and emulation on a Linux testbed. The results 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 flexible framework opens a whole class of design options for improving the performance of TCP in

highspeed networks. This document is an effort to solicit more experimentation and feedback from the broader networking community.

1. Introduction

Over the past few decades the traffic on the Internet has increased by several orders of magnitude. However, the Internet remains a stable medium for communication. This stability has been attributed primarily to the wide-spread use of congestion control algorithms of TCP [F00] which use the additive increase / multiplicative decrease policy for moderating the congestion window. Specifically, the congestion window evolves as follows - when there are no losses in the network, the window is increased by one for each RTT and upon a loss of packet, the window is reduced by half. While such window behavior helps maintain proportional fairness (among flows with similar RTTs), recent studies have shown it to be highly inefficient in utilizing available bandwidth in high bandwidth networks with capacities close to or in excess of 1Gbps.

Several solutions have been proposed to remedy the problem. We bring our proposal of LTCP to the research community through this draft to be evaluated in comparison with the other proposed schemes. LTCP enhances the congestion control algorithms of TCP with a simple layering technique. The idea of layering to probe and utilize the available bandwidth has been studied previously in the context of video transmission on the Internet and in multicasting[MJV96,VRC98]. The contribution of this document is to extend this idea to the congestion control algorithms in TCP to provide a general framework where scalability can be achieved at the cost of minimal modifications to the existing implementations. We provide the intuition for the framework and a simple design for illustrating the effectiveness of layering. The design is evaluated through analysis, ns-2 simulations and emulations on a Linux testbed. We welcome comments and research regarding other possible design options.

2. Problem Description

The throughput of a TCP connection is given by T = (1.2 * S) / (R * sqrt(p)), where 'S' is the packet size, 'R' is the round trip time for the connection and 'p' is the packet loss rate [PFTK98]. This means that for a standard TCP connection using a packet size of 1500 bytes over a connection with round trip delay of 200ms and packet loss rate of 10^(-5), the maximum throughput that can be achieved is 23.2Mbps. If the packet loss rate were reduced to 10^(-7) the maximum throughput could be increased to 232.4Mbps. Conversely, to achieve a throughput of 1Gbps, the packet loss rate required is 5.4 X 10^(-9) or lower and for 10Gbps it should be 5.4 X 10^(-11). These loss rates are unreasonable - for the 10Gbps, the loss rate translates to a loss of at most one packet in 1.85 X 10^(10) packets or at most one loss for every six hours ! Clearly, the congestion response function of the standard TCP connections does not scale in high capacity networks, and new solutions are required for mitigating this problem.

3. Design Guidelines

The proposal for LTCP in this document is motivated by the following requirements -

* Better efficiency in bandwidth utilization : the new scheme should be capable of making good use of available bandwidth in high capacity networks with realistic packet loss rates. For any given loss rate, the performance should be atleast as good as standard TCP implementation.

* Fairness to each other : when there are several flows of the new scheme in the network, they should be capable of achieving similar throughputs (provided they have similar RTTs).

* Fairness to TCP : the flows using the new scheme should be fair to TCP flows in networks with low available bandwidth where the congestion window remains below a predefined window threshold W_T. In networks that allow the congestion window to grow larger than W_T, the flows using the new scheme should allow the TCP flows to operate without starvation.

* Incremental Deployment : The new scheme should require minimal modifications to the existing TCP implementations and none to the network infrastructure. In other words, no additional feedback or modifications should be required at the network routers or receivers.

* Flexible Design : The new scheme should be capable of using options such as SACK, ECN etc. It should also be capable of operating with new proposals such as limited slow start [F04] for modified slowstart etc.

* RTT Unfairness Issues : The new scheme should be no worse than the existing TCP implementation regarding RTT unfairness.

4. LTCP Protocol

4.1. Framework

The layered TCP scheme is a sender-side modification to the congestion response function of TCP for making it more scalable in high-speed networks. The congestion window response of the LTCP protocol is defined in two dimensions - (a) At the macroscopic level,

LTCP uses the concept of layering to quickly and efficiently probe for available bandwidth (b) At the microscopic level, it extends the existing AIMD algorithms of TCP to determine the per-ack behavior. This section presents the intuition for the layering framework.

We start by defining the parameter LTCP window threshold (W_T) as the window size below which LTCP is fair to standard TCP implementations. The optimal value of W_T is debatable and likely a topic of research by itself. In this document, we choose a heuristic value of 50 packets for W_T. This value is motivated by the fact that when the window scale option [JBB92] is not turned on, the maximum window size allowed is 64Kb which is about 44 packets (of size 1500 bytes). The window scale option is used in highspeed networks, to allow the receiver to advertise large window size. In order to ensure that the new scheme with aggressive bandwidth probing is fair to TCP is slow networks, we choose a window threshold value W_{T} that is slightly greater than 44 packets. This imposes a constraint on LTCP to maintain proportional fairness to TCP flows in slow networks. Other arguments for keeping new protocols fair to TCP below a window threshold have been put forth in [F03,K03].

In order to ensure that below the threshold W_T, LTCP is fair to TCP, all new LTCP connections start with only one layer and behave in all respects the same as TCP. If the congestion window increases beyond the threshold W_T, the congestion window response is modified. 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. At the microscopic level, when operating at some 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 [CO98].

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 delta_K. When the current congestion window exceeds the window corresponding to the last addition of a layer (W_K) by the step-size delta_K, a new layer is added. Thus,

W_1 = 0, W_2 = W_1 + delta_1, ... W_K = W_(K-1) + delta_(K-1)

and the number of layers is 'K', when W_K <= W < W_(K + 1). Fig. 1 shows this graphically.


```
            Layer                                Minimum Window
            Number                          Corresponding to the Layer
               |                                       |
```

```
              |                                    |
              V                                    V
            K+1 ------------------------------ W_(K+1)
                                 ^
                                 |
                                 | delta_K
                                 |
                                 V
             K  ------------------------------   W_K
                                 ^
                                 |
                                 | delta_(K-1)
                                 |
                                 V
            K-1 ------------------------------ W_(K-1)
```

Fig 1: Graphical Perspective of Layers in LTCP

The step size delta_K associated with the layer K should be chosen such that convergence is possible when several flows share the bandwidth. Consider the simple case when the link is to be shared by two LTCP flows. Say, the flow that started earlier operates at a higher layer K1 (with a larger window) compared to the later-starting flow operating at a smaller layer K2 (with the smaller window). In the absence of network congestion, the first flow increases the congestion window by K1 packets per RTT, whereas the second flow increases by K2 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 delta_K1 is the stepsize associated with layer K1 and delta_K2 is the stepsize associated with layer K2, then

    delta_K1 / K_1 > delta_K2 / K_2

when K1 > K2, for all values of K1, K2 >= 2.

The design of the decrease behavior is guided by similar reasoning - in order for two flows 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 time it takes the smaller flow to regain the bandwidth it gave up. Suppose the two flows are operating at layers K1 and K2 (K1 > K2), and WR_K1 and WR_K2 is the window reduction of each flow upon a packet loss. After

the window reduction, suppose the layers corresponding to the two flows is K1' and K2'. Then, the flows take WR_K1/K1' and Wr_K2/K2' RTTs respectively to regain the lost bandwidth. From the above reasoning, this gives us -

    WR_K1/K1' > WR_K2/K2'

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. 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 architecture is to determine an appropriate relationship for the step size (delta) and window reduction that satisfy the conditions -

```
      delta_K1 / K_1 > delta_K2 / K_2 ---> [ the LTCP Constraint 1 ]
      WR_K1 / K1' > WR_K2 / K2'       ---> [ the LTCP Constraint 2 ]
```

when K1 > K2, for all values of K1, K2 >= 2.

Several different choices are possible and we encourage discussion and feedback from the research community for the choice of the optimal design. To evaluate the effectiveness of the architecture we use a simple design, the details of which are presented in the next section. This design is by no means the only or the best design solution for the problem. We choose it for its simplicity and the ease of deployment that it provides.

4.2. Design Choice

In order to evaluate the effectiveness of the LTCP protocol, we chose a simple design choice where we retain the multiplicative window reduction behavior. Upon a packet loss the window reduction is chosen to be

```
        WR = beta * W
```

where beta is a constant < 1. 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, if K1' and K2' are the layers at which the larger and the smaller flow operate after a packet loss, there are four possible cases
(a) K1' = K1, K2' = K2
(b) K1' = K1, K2' = (K2-1)
(c) K1' = (K1-1), K2' = K2
(d) K1' = (K1-1), K2' = (K2-1).
It is most difficult to maintain the convergence properties, when the larger flow does not reduce a layer but the smaller flow does, ie, K1'=K1, K2'=(K2-1).

With this worst case situation, the LTCP Constraint 2 can be written as -

$$WR\_K1/K1 > WR\_K2/(K2-1)$$

If this inequality is maintained for adjacent layers, then by simple extension, it will be maintained across all layers. So consider the case where K1 = K and K2 = (K-1). Also, suppose the window of the larger flow is W' and that of the smaller flow is W''. Then, the above inequality may be written as

$$W'/K > W''/(K-2)$$

The scenario that could result in the case considered above (K1' = K, K2' = (K-2)) will be when the window W' is close to transitioning into the layer (K+1) when the packet drop occurs, whereas the smaller flow has just transisioned into the layer (K-1). Suppose W_K is used to denote the window size when the flow transitions into layer K, then in the worst case, the following inequality should be satisfied -

$$W\_(K+1) > [\ K/(K-2)\ ] * W\_(K-1)$$

Based on this we conservatively choose,

$$W\_K = [\ (K+1)/(K-2)\ ] * W\_(K-1)$$

which defines the increase behavior.

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

Since layering starts at W_2 = W_T we have,

$$W\_K = [\ K\ (K+1)\ (K-1)\ /\ 6\ ] * W\_T$$

By definition, delta_K = W_(K+1) - W_K. By simple substitution, it

can be shown that this design satisfies the LTCP Constraint 1. Also, since the scheme was designed with the worst case for the inequality in the LTCP constraint 2, it satisfies the constraints when the flows are in adjacent layers which by simple extention holds for non-adjacents layers as well. It can also be shown that when two flows operate at the same layer, but with different window sizes, the inequality is still maintained.

4.2.1. Choice of beta

The above presented 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 beta carefully. The worst case for this situation occurs when the flow has just added the layer K and the window W = W_K + 'X', 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

$$\beta * W_K < \delta_{(K-1)}$$

(Ignoring the reduction due to 'X' since we are computing the worst case behavior.) On simple substitution, this yields,

$$\beta < 3/(K+1)$$

We show in later sections that a choice of 0.15 for the value of beta will allow the number of layers K to be sufficiently large enough to efficiently utilize the available link bandwidth in highspeed networks while maintaining the above inequality.

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 beta.

4.2.2. Time to claim bandwidth and packet recovery time

Suppose the maximum window size corresponding to the available throughput is W_K. Then, if we assume that slowstart terminates when layering starts, we can show that the time to increase the window to W_K is -

$$T' + [(K-2)(K+3) / 4] * W_T$$

where T' is the time spent in slow start. This document DOES NOT recommend terminating slowstart when the layering starts. The analysis here uses this assumption to explain the characteristics of the lTCP protocol behavior.

Table 1. below 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 is therefore K=15.

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.

Also, an LTCP flow with window size W will reduce the congestion window by beta * W. 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 packet loss recovery time then, for LTCP is (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 1 shows the speed up in packet recovery time for LTCP with beta = 0.15 compared to TCP. Based on the conservative assumption that the layer number is (K-1) 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).

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

Table 1: Comparison of LTCP (with W_T = 50 and beta = 0.15) to TCP

4.2.3. Convergence

The inequality in the LTCP constraint 2 of the LTCP framework ensures that flows will converge asymptotically to a fair share, since a larger flow takes longer time to recover lost throughput than a smaller flow. With the current design, we can show that this inequality is held after a congestion event irrespective of the layer transitions and hence asymptotic convergence is ensured.

4.2.4. Throughput Analysis

The bandwidth BW of an LTCP flow operating at layer K in steady state, in a network with uniform loss probability p and round trip time RTT can be shown to be [BJR04] -

$$BW = \frac{\mathrm{Sqrt}(CK')}{(RTT * \mathrm{sqrt}(p))}$$

where C is a constant = (1/beta - 0.5)

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 Table 1, 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 throughput compared to TCP.

4.2.5. RTT Fairness

In [XHR04] the authors have shown that some of the recent proposals for TCP in highspeed networks that change the congestion window behavior of TCP to scale to high bandwidth might aggravate the RTT unfairness problem of TCP. Analysis conducted along the same lines as that in [XHR04] shows that the above design for LTCP could potentially alleviate the RTT unfairness problem. For two LTCP flows with RTTS RTT1 and RTT2 operating at window layer K1' and K2' after a packet drop, the throughput ratio can be shown to be

```
        K2'/K1' * Square(RTT2/RTT1)
```

The relationship between K and RTT for an LTCP flow is -

```
        K ~ O( (1/RTT) ^ 0.3333)
```

Thus LTCP can potentially alleviate the RTT unfairness problem of TCP. This has been verified through simulations on the ns-2 simulator.

4.2.6. Alternate Designs

This document presents one possible design for LTCP and provides the relevant analysis to understand the protocol behavior with this design. This design has several desirable properties such as efficient scaling of the window in high speed networks, quick convergence to fairness in the presence of multiple flows and alleviated RTT fairness problem. However, we would like to stress at this point that 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. The design was derived based on the basic premise of choosing a multiplicative decrease. Several alternate design options are possible. For instance by adding layers using the simple rule W_K = alpha * W_(K-1) an LTCP protocol can be designed that has very similar dynamics as TCP regarding window reduction and convergence but faster increase behavior. We solicit feedback and suggestions from the research community in our quest for the design choice that optimises the tradeoff between improving performance and implementation overhead.

4.3. Implementation Details

The LTCP protocol requires simple sender-side changes to the congestion window response function of TCP. It uses two additional parameters - W_T, and beta. Default recommendation for, W_T and beta are 50 and 0.15 respectively. Additionally, variables need to be used for saving the number of layers (K) and the window corresponding to layer K (W_K).

When a new connection is established, the protocol is started with K = 1, and the slowstart algorithm of standard TCP. When slowstart is exited, the number of layers K is obtained based on the current cwnd. If K = 1, LTCP behaves in all respects similar to TCP. Otherwise (congestion window exceeds W_T), the following changes are made to the TCP congestion response function

```
if (newack)
{
        cwnd = K/cwnd
        if (window() > W_(K+1))
                K++
}
if (packet loss)
{
        cwnd = cwnd (1 - beta)
        if (window() < W_K)
                K--
}
```


The rest of the algorithms used in the traditional implementation of TCP - for instance the algorithms for RTT calculations, SACK processing, timer management etc, remain unchanged. The LTCP modifications work with most flavors of the TCP protocol. However, this document advocates the use of LTCP with TCP-SACK to ensure that the performance can be maintained high even under the conditions of multiple losses per round trip time. If the receiver is not SACK-capable, however, then the sender will have to use NewReno. The LTCP changes affect the behavior of TCP only in the congestion avoidance phase. The slowstart algorithm is not modified and hence LTCP maybe used with newer slowstart proposals such as limited slowstart [F04].

5. Performance Evaluation

The performance of LTCP for the above mentioned design choice was evaluated through both ns-2 [NS-2] simulations and experiments on the real network using a modified Linux kernel. These results may be found in [BJR04]. This section of the document provides a brief summary of the same. Also, the Linux implementation is currently being tested against other proposals for highspeed networks at the Stanford Linear Accelerator Center at Stanford University, and the results will be made available in the future.

In both simulations and emulations, LTCP exhibited substantially improved link utilization compared to TCP. The window size required to fill the link bandwidth was reached several orders of magnitude faster than TCP for links with 1Gbps capacity. In steady state, the fluctuations about the optimal value was much smaller than that of TCP. When several LTCP flows used the same bottleneck link, the available bandwidth was shared in a fair manner with the Jain fairness index being very close to 1, as the number of flows was varied from 2 to 10. When the dynamic link conditions were varied by adding and removing LTCP flows at different times, LTCP exhibited good convergence properties. Established LTCP flows gave up a portion of the link bandwidth when standard TCP flows were introduced, to let them run free of starvation. RTT unfairness among LTCP flows was lesser than the RTT unfairness among TCP flows under similar network conditions. Several tests were conducted in the presence of non-responsive traffic and the quick response to varying traffic was verified.

6. Incremental Deployment

The LTCP modifications proposed in this document lend themselves to incremental deployment. Only the TCP stack on the sender side needs to be modified. No changes are required at the receivers or the routers and no additional feed back is expected from either. The use of LTCP does not require the sender and receiver to negotiate any

conditions during connection setup. Neither the receivers nor the routers need to be aware that the sender is using the LTCP congestion response function. The sender-side LTCP modifications themselves are simple and can be distributed easily as kernel patches.

7. Relationship to other work

Solutions for improving the performance of TCP for high-speed networks can be classified into four main categories - a) Tuning the network stack (web100[MHR03], net100 [DMT02], Dynamic Right Sizing [WF01], Enable Tuning [TGLSE01] etc) b) Opening parallel TCP connections between the end hosts (XFTP [OAK96], GridFTP [LGTABBT01], storage resource broker [BMRW98], Parallel Sockets Library [SBG00], MulTCP [CO98] etc) c) Modifications to the TCP congestion control (HSTCP [F03], FAST [JWL04], Scalable TCP [K03], Bic-TCP [XHR04], H-TCP [SLFK03] etc) d) Modifications to the network infrastructure or use of non-TCP transport protocol (XCP [KHR02], Tsunami [Tsunami], RBUDP [HLYD02], SABUL [SGMPZ] etc).

The LTCP solution falls in the third category and deals with modifications to the TCP congestion control. It tries to emulate parallel TCP connections like MulTCP (mentioned in the second category above), with the key difference that the number of virtual flows is not fixed. Instead, layering concept similar to that in [MJV96,VRC98] is used for increasing/decreasing the number of layers dynamically to find the optimal number of virtual flows required to keep the bottleneck link full, while at the same time maintaining a notion of fairness.

8. Security Considerations

This proposal makes no changes to the underlying security of TCP.

9. Conclusions

In this document we have proposed LTCP, a layering technique for the congestion control mechanism of TCP to make it more scalable in highspeed networks. We have presented the general framework for LTCP and explored the proposal though one possible design choice, its analysis and experimental results. We believe that LTCP provides a simple solution for improving the the performance of TCP in highspeed networks without modifying the TCP semantics significantly, and with minimal implementation overhead. We welcome additional analysis, simulations, experimentation or feedback regarding regarding this proposal.

We are bringing this proposal to the IETF to be considered as an Experimental RFC.

11. References

[BJR04] Sumitha Bhandarkar, Saurabh Jain and A. L. Narasimha Reddy, ``LTCP: A layering technique for improving the performance of TCP in high speed networks'', Technical Report. http://ee.tamu.edu/~reddy/papers/jogc2003.pdf

[BMRW98] C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC storage resource broker", In Proc. CASCON'98 Conference, Dec 1998.

[CO98] 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.

[DMT02] Tom Dunigan, Matt Mathis and Brian Tierney, "A TCP Tuning Daemon", SuperComputing (SC) November, 2002.

[F00]Sally Floyd, "Congestion Control Principles", RFC 2914, September 2000

[F03] Sally Floyd, "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

[F04] Sally Floyd, "Limited Slow-Start for TCP with Large Congestion Windows", RFC 3742, January 2004.

[HLYD02] 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.

[JBB92] V. Jacobson, R. Braden and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[JWL04] Cheng Jin, David X. Wei and Steven H. Low, "FAST TCP: motivation, architecture, algorithms, performance", IEEE Infocom, March 2004.

[K03] Tom Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks", ACM Computer Communications Review, April 2003.

[KHR02] Dina Katabi, Mark Handley, and Chalrie Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", Proceedings of ACM SIGCOMM 2002, August 2002.

[LGTABBT01] 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.

[MHR03] M. Mathis, J Heffner and R Reddy, "Web100: Extended TCP Instrumentation for Research, Education and Diagnosis", ACM Computer Communications Review, Vol 33, Num 3, July 2003.

[MJV96] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast", Proceedings of ACM SIGCOMM '96, August 1996.

[NS-2] ns-2 Network Simulator. http://www.isi.edu/nsnam/

[OAK96] Shawn Ostermann, MArk Allman, and Hans Kruse, "An Application-Level solution to TCP's Satellite Inefficiencies", Workshop on Satellite-based Information Services (WOSBIS), November, 1996.

[PFTK98] J.Padhye, V.Firoiu, D.Towsley, and J.Kurose, "Modeling TCP throughput: A simple Model and its empirical validation", ACM SIGCOMM '98, Oct. 1998.

[SBG00] 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.

[SGMPZ] H. Sivakumar, R. Grossman, M. Mazzucco, Y. Pan, and Q. Zhang, ``Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks'', submitted for publication.

[SLFK03] 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.

[TGLSE01] 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.

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

[VRC98] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer", Proceedings of IEEE Infocom ’98, March 1998.

[WF01] 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.

[XHR04] 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.

13. Author’s Addresses

Sumitha Bhandarkar And Saurabh Jain
Dept. of Elec. Engg.
214 ZACH
College Station, TX 77843-3128
Phone: (512) 468-8078 / (979) 260-2811
Email: {sumitha,saurabhj}@tamu.edu
URL : http://students.cs.tamu.edu/sumitha/
       http://ee.tamu.edu/~saurabhj

A. L. Narasimha Reddy
Associate Professor
Dept. of Elec. Engg.
214 ZACH, Mailstop - 3128
College Station, TX 77843-3128
Phone : (979) 845-7598
Email : reddy@ee.tamu.edu
URL   : http://ee.tamu.edu/~reddy/