Comparative Study of TCP Compatible Binomial Congestion Control Schemes

K. Chandrayana, B. Sikdar, S. Kalyanaraman,
Department of ECSE, Rensselaer Polytechnic Institute
Troy, NY 12180

Abstract— **Current TCP implementations employ Additive Increase Multiplicative Decrease (AIMD) as the congestion control mechanism. Recently, a new set of schemes called Binomial Congestion Control Schemes (BCCS) were proposed and a section of these schemes is TCP compliant. In this paper we evaluate the performance of these TCP compliant binomial schemes and show through simulations that AIMD performs better than the other BCCS policies in a wide range networking environments. Specifically, we study the performance of these schemes with respect to throughput, fairness, losses, timeouts and self-similarity. We show that the superior performance of AIMD can be attributed to its more conservative attitude in the presence of losses when it reduces its transmission rate much faster than the other schemes. This results in smaller congestion periods thereby reducing the losses and timeouts which in turn increases the throughput and decreases the degree of self-similarity of the traffic. We also evaluated the performance of TCP Compatible BCCS when they compete with TCP flows. It was found that with sufficiently large number of flows, BCCS competes fairly with TCP. However, with a smaller number of flows in the network TCP flows get smaller share of the bottleneck and disproportionately higher losses and timeouts.**

I. Introduction

The congestion control scheme of TCP has been the focus of numerous studies and has undergone a number of enhancements and the Additive Increase Multiplicative Decrease (AIMD) congestion control policy has become the de-facto standard in most TCP implementations. In [4] the authors successfully argued in favor of AIMD over other policies for achieving higher throughput and equi-fairness with bulk data transfers. They further showed that this scheme of things was in fact stable too. However, as the Internet continues to evolve, the demand for video and real time applications has grown and hence the focus shifted from AIMD to more slowly responsive congestion control schemes. This was indeed necessary, as the current implementations of TCP with AIMD control react very sharply to a loss indication resulting in large fluctuations in the data rate and thus affect the quality of service required for real time applications.

Recently, a number of alternatives have been suggested for real time applications which are also TCP compatible [8], [9], [5], [2]. TCP compatible schemes can be described as policies which interact well with TCP and maintain the stability of the network [3]. This can be further explained as: TCP compatible schemes provide fair bandwidth allocations when compared to TCP connections using AIMD. Specifically, we can associate the TCP compatibility definition to TCP's throughput formula as follows. The throughput of TCP's AIMD congestion control mechanism, λ, is defined as $\lambda \propto S/(R\sqrt{p})$, where *S* is the packet size, *R* is the round-trip time and *p* is the packet loss probability. For the same packet-size and round-trip time an algorithm is said to be TCP Compatible if its throughput, λ is, $\lambda \propto 1/\sqrt{p}$, where *p* is the packet drop probability for TCP's AIMD control with same settings.

Supported in part by ARO contract DAAD 19-00-1-0559, NSF contract ANI 9806660, DARPA contract F30602-00-2-0537 and Intel Corp.

In [2] the authors proposed a class of non-linear TCP compatible congestion control schemes called Binomial Congestion Control Schemes (BCCS) which are well suited for real time streaming applications. AIMD can be considered as one of congestion control schemes in the subset of TCP compatible BCCS. Formally, the Binomial Congestion Control scheme can be defined as:

$$\begin{array}{rl} W_{t+R} \leftarrow W_t + \alpha/W_t^k & \text{if no loss} \\ W_{t+\delta t} \leftarrow W_t - \beta W_t^l & \text{if loss} \end{array} \quad (1)$$

where *k* and *l* are window scaling factors for increase and decrease respectively and α and β are increase the decrease proportionality constants. For any given values of α and β TCP Compatible BCCS can be defined by $k + l = 1 : k \geq 0, l \geq 0$.

In this paper, we investigate the performance of the schemes defined under the TCP compatible BCCS. We investigate the extent to which these scheme can deliver on its goals of improving throughput, fairness, reducing timeouts etc. We find that AIMD control performs as well or better than the other TCP compatible BCCS, independent of the capacity and buffer size at the bottleneck and for both short and long flows. In addition to looking at the usual performance metrics of throughput, fairness etc., we also look at the self-similarity of various TCP compatible BCCS. We find that AIMD control is least self-similar, as compared to the other schemes in the TCP compatible BCCS in all the metrics. This effect can be attributed to the reduction in the timeouts with AIMD control. In [6], [10] the authors show that the timeouts are one of the main causes of the self-similar behavior of the TCP. AIMD control, by decreasing multiplicatively (drastically) avoids timeouts as against other schemes of the TCP compatible BCCS, which by reacting slowly to the losses are more prone to get into timeouts.

We also evaluated the performance of TCP Compatible BCCS when they compete with TCP flows. It was found

that with sufficiently large number of flows, BCCS competes fairly with TCP, i.e., the average throughput are same for both TCP and BCCS, and the losses and timeouts are distributed evenly. However, when the number of flows in the network is small, BCCS beats TCP flows, thus giving it a disproportionately higher losses and timeouts. This can be explained as, with smaller number of flows in the network, the average share of each flow is large; therefore the probability that TCP flows see a burst drop increases thus increasing the probability of more timeouts. The TCP flows thus spend comparatively more time in timeouts and hence have smaller average share of the bottleneck.

The rest of the paper is organized as follows. Section II discusses BCCS and previous work in this area. Section III describes the implementation details, performance metrics and the simulation setup. Comparison of the performance metrics of TCP compatible BCCS is presented in Section IV. Finally, we present the conclusions in Section V.

II. Background and Related Work

With the growth of the Internet, real time applications have gained in popularity and are claiming increasing portions of the Internet traffic. However, studies have shown that TCP is not suitable for carrying such traffic as it increases end-to-end delays and delay variations [2], [5]. Also, many applications fail to respond appropriately to large and sudden rate cuts in TCP, which owes its origin to the multiplicative decrease of AIMD control. As such, it is imperative to look into schemes which minimize delays and delay variations but at the same time are responsive schemes (by "responsive schemes", we refer to schemes which react by cutting down their rates when notified of congestion). Also, since TCP constitutes most of the Internet traffic, care should be taken in designing these scheme so that they conform to the definition of TCP Friendliness.

TCP compatible schemes can be described as schemes which interact well with TCP and maintain the stability of the network [3]. Specifically, we can associate the TCP compatible definition to the throughput formula for congestion control schemes, as follows. As mentioned earlier, a congestion control scheme said to be TCP Compatible if its throughput, λ is, $\lambda \propto 1/\sqrt{p}$, where p is the packet drop probability for TCP's AIMD control with same settings. Recently, a number of TCP compatible schemes have been proposed [8], [9], [5], [2]. In [5] the authors propose an equation based congestion control mechanism where the sender adjusts its sending rate as a function of the measured loss event rate. The performance of equation based congestion control is compared against AIMD control and proved to be better in [5]. RAP or Rate Adaptation Protocol proposed by [9] uses AIMD control and attempts to de-couple congestion and error control. The source's transmission rate is changed by reducing the gap between transmitted packets, which consequently increases the rate. The performance of RAP is compared to equation based control in [7] and equation based control is shown to be better than RAP in most of the cases.

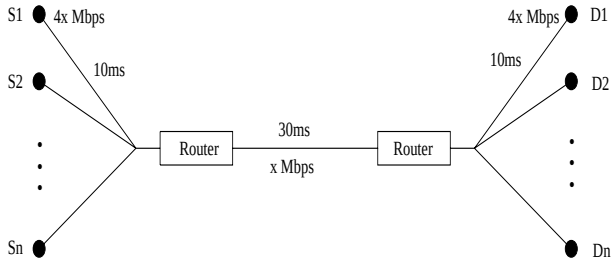


Fig. 1. Topology used in the simulation.

Binomial Congestion Control was proposed by [2] and can be considered as a non-linear generalization of AIMD type control. These algorithms can be characterized by four parameters k, l, α and β, (refer to equation 1) where α and β are the synonymous with the increase and decrease parameter of AIMD. Using simple techniques it can be shown that the throughput, λ of a BCCS can be stated as $\lambda \propto 1/p^{\frac{1}{k+l+1}}$. As such, all the schemes which have $k + l = 1 : k \geq 0, l \geq 0$ will have their throughput as $\lambda \propto 1/\sqrt{p}$, and therefore will be TCP compatible (AIMD can be identified with k=0 and l=1).

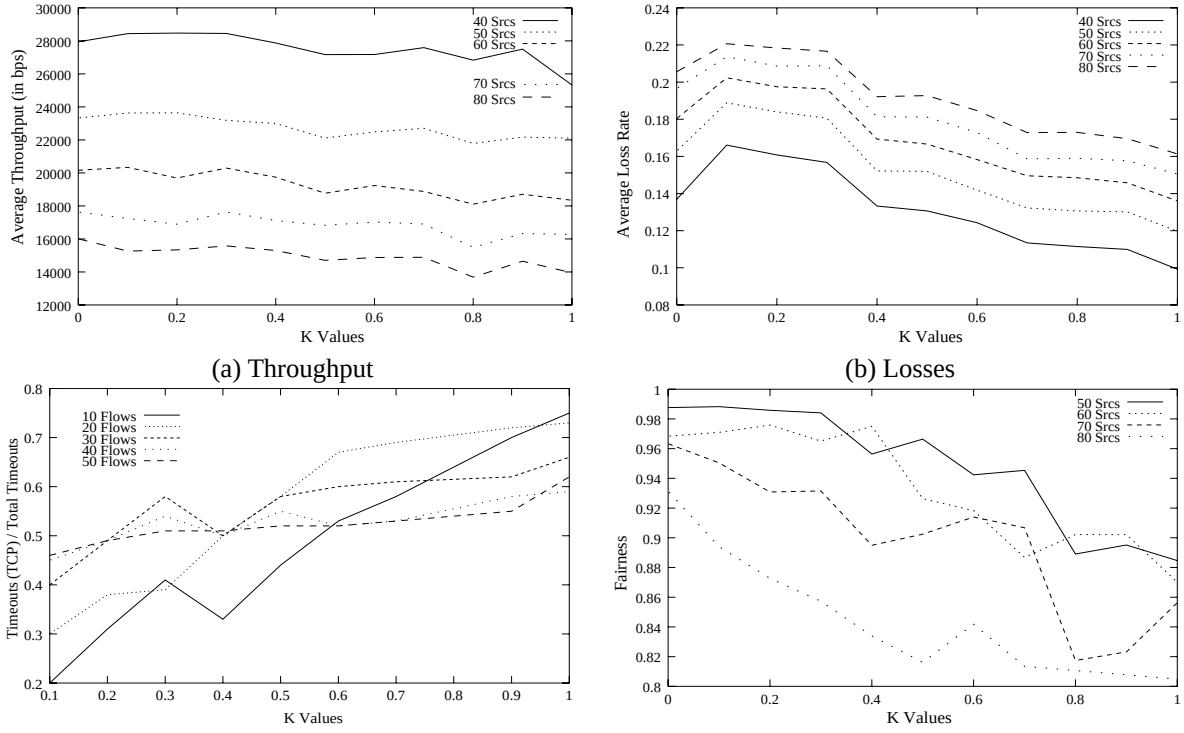
In [1] the authors test the dynamic behavior of TCP Compatible BCCS and compare it to the other proposed TCP friendly schemes. It is shown through simulations there that though these TCP friendly schemes are safe to deploy, their behavior changes under dynamic conditions. Specifically, they show that the TCP friendly schemes are indeed friendly under static conditions but, they do not compete fairly with dynamic scenario. However, BCCS do not suffer from this problem. They argue that this can be attributed to the self-clocking feature of BCCS. The reader is referred to [2], [1] for further details of BCCS and its dynamic behavior.

In this paper, we compare the performance of various schemes which fall in the BCCS, viz., AIMD, IIAD (Inverse Increase Additive Decrease, k=1, l=0), SQRT (Square Root, k=0.5, l=0.5). We also investigate the other schemes which lie between AIMD, IIAD and SQRT.

III. Simulation Setup

We have used the BCCS implementation in the network simulator *ns*. For our simulations, we used the congestion control and loss recovery mechanisms of TCP Reno and thus BCCS has the usual slow-start and fast recovery and retransmit mechanisms. For the simulations reported in this paper, we disabled the delayed acknowledgments option.

Figure 1 shows the topology used in the simulations. The access links were configured at a rate 4 times greater than that of the bottleneck link. All the links use taildrop (FIFO) queues unless otherwise specified. Default settings for the simulations are a round-trip time of 100ms, a bottleneck bandwidth of 1Mbps and with the bottleneck router having a buffer size of 100 packets. The maximum advertised window is set sufficiently high so that it does not constrain the



(a) Throughput (b) Losses

(c) Timeouts (d) Fairness

Fig. 2. Average Throughput, Average Loss Rate, Average Percentage Timeouts, Fairness Vs k Values, $k + l = 1, k, l \geq 0$

actual window. We use a segment size of 1000 bytes.

We evaluate the performance of BCCS for the following set of metrics: average throughput, fairness, drop rates, timeouts, latency and the degree of self-similarity of the traffic. We characterize fairness using the modified Jain's fairness index, [4]. Jain's fairness index is defined as

$$f = \frac{(\sum_{i=1}^{n} x_i.RTT_i)^2}{n(\sum_{i=1}^{n}(x_i.RTT_i)^2)} \quad (2)$$

where x_i is the throughput of the i^{th} flow, RTT_i is the round-trip time of flow i and n is the number of competing flows.

The results corresponding all the metrics other than the degree of self-similarity were calculated on a per flow basis and we present the average values computed over all the flows. For the results on the self-similarity, we calculated the self-similarity of the aggregate traffic at the bottleneck link. We report the degree of self-similarity, as represented by the Hurst parameter which was calculated for each simulation scenario using the following three widely used methods [11]: absolute value method, R/S statistics method and the periodogram method.

We present results for a class of BCCS schemes which encompasses the entire set of TCP friendly BCCS schemes. The results for each BCCS scheme was calculated over a wide range of networking scenarios which included passive (taildrop) or active queue management techniques, different buffer sizes at the bottleneck, various link speeds and propagation times and varying number of flows in the network. (For the reasons of space constraints we present the results with taildrop queues and varying number of flows in the network.) The range of the k and l values lie in the range $0 \leq k, l \leq 1$ with $k + l = 1$. As noted before, this covers schemes like AIMD, IIAD and SQRT.

IV. Results

In this section we present the simulation results. We investigated two scenarios, one where all the competing sources employ the same congestion control scheme (Section IV-A) and the second where the competing sources employ different congestion control schemes (Section IV-B).

A. Sources Employ Same Congestion Control Schemes

The results presented in this section correspond to a topology with a bottleneck bandwidth of 1 Mbps, access link speeds of 5 Mbps, round trip time of 100ms and taildrop queues of 100 packets at the bottleneck. These simulations were done for 40, 50, 60, 70 and 80 competing sources of same type to vary the load on the network. All the simulation results are for bulk-data transfers, where the sources have infinite data to send. Each simulation was run for 3600 simulated seconds.

We first observe the effect of changing the k, l values on the average throughput, fairness, loss rates and timeouts. Figures III, III, III and III plot the average throughput, fairness, loss rate and percentage timeouts respectively. The five curves in each figure represent various load conditions corresponding to 40, 50, 60, 70 and 80 competing sources of same type. As is evident from Figure III, in all the scenarios, the throughput is maximized for AIMD ($k = 0$, $l = 1$) and is least for IIAD ($k = 1$, $l = 0$). This can be explained from the loss rates and the percentage of losses

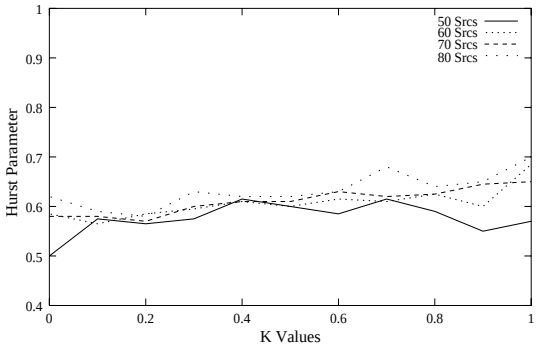


Fig. 3. Hurst Parameter Vs k Values, $k + l = 1, k, l \geq 0$

which resulted in timeouts as shown in Figures III and III respectively. Though the loss rate is least for IIAD, a larger fraction of these losses resulted in a timeout. This is because, IIAD decreases its window additively, i.e., by 1 on every loss, effectively resulting in more bursty losses since the flows do not reduce their rates fast enough. These bursty losses forces the IIAD into timeouts and hence decreasing the throughput. AIMD, in contrast by decreasing most aggressively (by cutting down its window by half on every loss), tends to be more conservative and avoids bursty losses. This results in most of the losses being recovered using fast retransmit and followed by fast recovery. Thus at the end of loss recovery, the flow's window is is only reduced by half instead of making it equal to one thereby obtaining higher transmission rates. The reduction in the number of timeouts and the associated exponential backoffs and slow start phases with AIMD ensures that the throughput stays higher than schemes which lead to higher degrees of timeouts. Also it is interesting to note that AIMD seems to be the most fair scheme of all the TCP compatible BCCS. Also, as the number of competing sources increase, IIAD and other slowly responsive schemes (in this case $k > 0.7$) deteriorate very fast.

For the self-similarity tests, we collected the packet arrival characteristics at the bottleneck link and measured the degree of self-similarity of the aggregate traffic arriving at the bottleneck. For estimating the Hurst parameter, H, we used three widely used methods which are the absolute value method, R/S statistics method and the periodogram method. The values of H obtained from all these methods are very close and lie within ± 0.03 of each other. Figure 3 shows the Hurst parameter when the number of competing flows are 50, 60, 70 and 80. As is evident from the graph, the lowest values of H occur when the value of k is less than 0.2 and keep increasing as k increases and we move towards IIAD policies. In [6], [10] the authors show that timeouts in TCP are one of the main contributors of the self-similar nature of network traffic. Thus, a reduction in the number of timeouts should result in lesser self-similar nature or lesser value of H. Again, reduction in the number of timeouts as shown for AIMD and for small values of k in Figure III can be cited as the reason for this decrease in values of the Hurst parameter. Note that as k increases, the number of timeouts also increases and we see a corresponding increase in the Hurst parameter.

B. Sources Employ Different Congestion Control Schemes

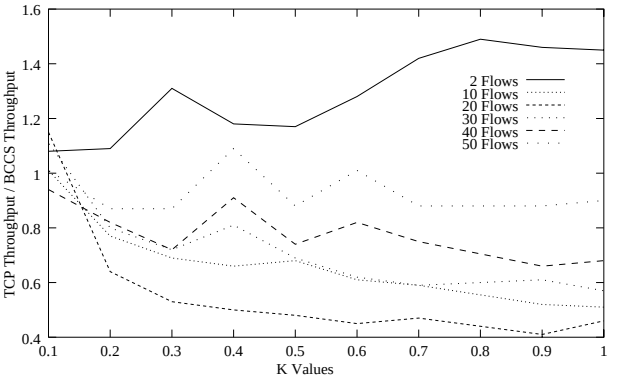
In this section, we present the results of how a TCP congestion control scheme competes with other TCP Compatible BCCS. The results presented in this section correspond to the topology as discussed in Section IV-A and were done for 2, 10, 20, 30 40 and 50 competing sources. In each simulation setup we had equal number of TCP and TCP Compatible BCCS sources, i.e. if the number of competing flows was 10 then we had 5 TCP sources and 5 TCP Compatible BCCS. All the simulation results are for infinite bulk-data transfers and were run for 500 seconds each.

Again, as in Section IV-A we vary the *k, l* values to generate different TCP Compatible BCCS. Specifically we vary *k* in the range [0.1, 1] and *l* in the range [0, 0.9], such that *k+l=1* is always maintained. We calculated the ratio of average throughputs of TCP and TCP Compatible BCCS as a measure of how these sources share resources amongst them. The results are plotted in IV-B. It can be inferred from the figure that as the number of flows increase in the network, the TCP and BCCS flows share bandwidth fairly. However, with a small number of flows in the network, it can be seen that TCP flows get beaten down (refer to simulation with 20 flows in the figure IV-B. We also plotted the percentage share of losses and timeouts for a TCP flow vis-a-vis total network losses and timeouts (figure IV-B and figure IV-B respectively). Again, it can be seen that with increasing flows in the network the losses and timeouts are disproportionately higher for TCP flows, however as the number of flows if further increased the losses and timeouts are more evenly distributed. This disproportionately larger number of timeouts and a smaller share of bottleneck with lesser flows in the network can be explained as: With smaller number of flows in the network, the average share of each flow is large; therefore the probability that TCP flows see a burst drop increases thus increasing the probability of more timeouts. The TCP thus spends more time in timeouts as compared to BCCS schemes and hence has smaller average share of the bottleneck.

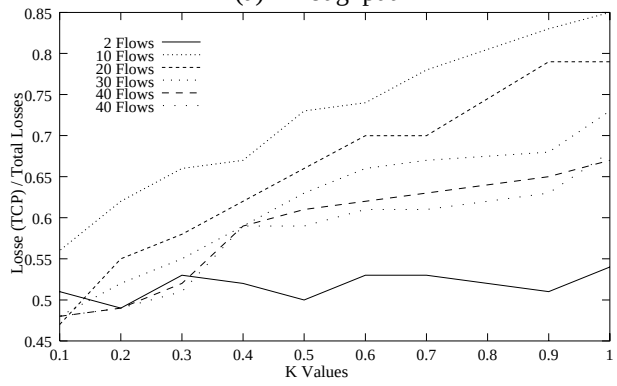
Thus there seems to be a saddle point with respect to number of flows, till where the TCP flows get beaten down by TCP Compatible BCCS but after that saddle point, TCP flows compete fairly with BCCS. Thus it can be concluded from the figures IV-B, IV-B and IV-B that with sufficiently large (in our simulation setup this equal to 50 flows) number of competing flows, TCP and TCP Compatible BCCS will compete fairly. Since, at any router in the Internet, the total number of flows will always be more than 50, hence TCP Compatible BCCS are safe to deploy wherein they don't suppress competing flows.

V. CONCLUSIONS

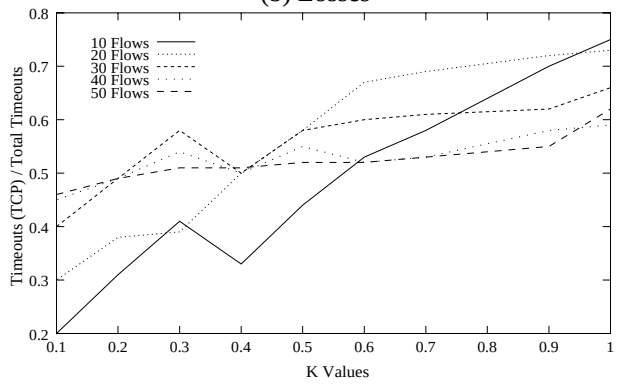
In this paper we investigated the performance of a class of TCP friendly binomial congestion control schemes.



(a) Throughput



(b) Losses



(c) Timeouts

Fig. 4. Ratio of Average Throughput, Average Loss Rate, Average Percentage Timeouts, of competing TCP and Binomial Flows Vs k Values, $k + l = 1, k, l \geq 0$

These schemes have the property of being able to vary their transmission rates smoothly in response to changing network conditions, making them suitable for real time and multimedia applications. However, before they are deployed, it is important to understand their behavior and operations under various networking environments. In this paper we have looked at the relative performance of BCCS schemes which satisfy the TCP friendly criterion using a number of metrics.

We compared the performance of the various congestion control schemes in terms of the throughput, loss rates, number of timeouts, fairness and the degree of self-similarity. For a wide range of simulation scenarios, we observed that AIMD outperforms the other schemes and there is a gradual deterioration in the performance as we move towards IIAD schemes (i.e. as k approaches 1). The reason behind such behavior can be attributed to the more conservative nature of AIMD in the presence of losses. When a loss is detected, AIMD reduces its window by half immediately thereby reducing the congestion and the occurrence of further losses which can lead to timeouts. On the other hand, as k increases and we approach IIAD schemes, the decrease of the window when a loss is detected is not very fast resulting in longer periods where the queues are full. This leads to higher loss rates and timeouts and consequently lower throughput which in turn increases the degree of self-similarity in the aggregate traffic.

We also evaluated the BCCS schemes under scenarios where they compete with TCP sources. As the number of flows increase we encounter a saddle point in terms of performance. With the flow multiplexing less than the saddle point, the TCP flows get beaten down by TCP Compatible BCCS but after that TCP flows compete fairly with BCCS. Thus it can be concluded that with sufficiently large (in our simulation setup this equal to 50 flows) number of competing flows, TCP and TCP Compatible BCCS will compete fairly. Since, at any router in the Internet, the total number of flows will always be more than 50, hence TCP Compatible BCCS are safe to deploy wherein they don't suppress competing flows.

REFERENCES

[1] D. Bansal, H. Balakrishnan, S. Floyd and S. Shenker , "Dynamic behavior of slowly-responsive congestion control schemes," *Proceedings of ACM SIGCOMM*, San Diego, CA, Aug 2001.

[2] D. Bansal and H. Balakrishnan, "Binomial congestion control algorithms," *Proc. IEEE INFOCOM*, Anchorage, AK, April 2001.

[3] B. Braden et. al , "Recommendations on queue management and congestion avoidance in the Internet," *RFC-2309*, April 1998.

[4] D-M. Chiu and R. Jain, "Analysis of increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1-14, June 1989.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," *Proc. of ACM SIGCOMM*, pp. 43-56, Stockholm, Sweden, August 2000.

[6] L. Guo, M. Crovella and I. Matta, "How does TCP generate pseudo-self-similarity?," *Proc. of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, Cincinnati, OH, August 2001.

[7] S. Hassan and M. Kara, "Simulation based performance comparison of TCP friendly congestion control protocols," *Proc. of the 16th Annual UK Performance Engineering Workshop*, Durham, UK, pp. 199-210, July 2000.

[8] S. Jacobs and A. Eleftheriadis, "Streaming video using TCP flow control and dynamic rate shaping," *Journal of Visual Communication and Image Representation*, vol.9, no.3, pp. 211-222, Sept '98.

[9] R. Rehaie, M. Handley and D. Estrin, "RAP: An end-to-end rate base congestion control mechanism for real-time streams in the Internet," *Proc. of IEEE INFOCOM*, New York, March 1999.

[10] B. Sikdar, K. Vastola, "The effect of TCP on self similarity of network traffic," *Proc. of* 35^{th} *CISS*, Baltimore, MD, March 2001

[11] M. S. Taqqu and S. Teverovsky, "On estimating long-range dependence in finite and infinite variance series," *A practical guide to Heavy Tails: Statistical Techniques and Estimation*, R. J. Alder, R. E. Feldman and M.S. Taqqu, Editors, pp 177-217 Birkhauser, Boston 1998.