Analysis of TCP Congestion Control using a Fluid Model *

Rajarshi Roy, Raghuraman C. Mudumbai and Shivendra S. Panwar †
Center for Advanced Technology in Telecommunications
Polytechnic University
6 Metrotech Center
Brooklyn, NY 11201

Abstract

In this paper we develop an analytical fluid model for TCP Reno based on delay differential equations. Using this model, we have developed a prototype for a fluid based simulation tool for TCP. We empirically validate this model by matching the window, throughput and queue evolution curves obtained from the model to that offered by a simulation output. We claim that this analytical model provides the basis for an analyzer that can complement simulators that perform a packet by packet simulation.

Keywords: TCP algorithms, Delay Differential Equations, Network Management.

1 Introduction

In this paper our aim is to describe a fluid based abstract model for TCP [17, 10, 15] and use it to develop a predictive tool for TCP-like congestion control protocols. We consider the behavior of a small number of TCP sessions, each supplied by a greedy source. The different connections can interact in the network in different ways, depending on the service and resource allocation policies. Such questions have been considered analytically, for traffic with stochastic arrivals. The main theme of our work is to develop a general framework to answer the above questions for deterministic TCP traffic.

We recognize that in addition to the transport layer algorithm, consideration of the statistical characteristics of the connection arrivals is also important for network performance. However, in this work, we have considered only the former. The advantage of our approach is that we are able to examine under fairly realistic conditions, the effect of the transport protocol and the way it interacts with the resource management policies within the network. We have validated our model by comparing it with results from simulation by ns [7] of some very simple network scenarios whose parameters correspond directly to those of our abstract model.

The paper is organized as follows: Section 2 introduces our model and presents the derivation of a set of equations that is central to this work. Section 2 also describes the prototype simulation tool we have developed based on our model. Section 3 describes some simple numerical results obtained from the tool. Section 4 concludes the paper and suggests possibilities for future work.

2 A Fluid Model for TCP

In this section, we present the derivation of a fluid based model for TCP. The concept of fluid in data networks is based on the assumption that in high capacity networks, most important dynamics depend not on how individual packets are processed, but rather on how aggregates of packets are processed. A good criterion to test the applicability of these ideas is that packet size should be a small fraction of typical buffer capacities in the network. Fluid models should be increasingly useful as we move towards higher capacity networks in the future.

The motivation for building a model for TCP is the following: Recent studies of internetwork traffic have shown that a high percentage of traffic uses TCP's congestion control. Thus, the effect of TCP's parameters on network performance is a very important issue. Our approach will enable us to obtain useful information and insight about key network performance measures, in a computationally inexpensive and flexible way.

2.1 Model description

We used some abstractions in constructing our fluid model. Referring to the block diagram shown in Fig-

*This research was supported by the New York State Center for Advanced Technology in Telecommunications, Polytechnic University and by the National Science Foundation under grant ANIR 0081527. The authors also wish to acknowledge the help of Jaewoo Park.

†Tel: 718 260 3420, Fax: 718 260 3074, e-mail: rroy@photon.poly.edu, mraghur@photon.poly.edu, panwar@catt.poly.edu

0-7803-7097-1/01/$10.00 ©2001 IEEE

ure 1, we have shown the control flow in a general TCP connection. If the connection has multiple hops, we focus on the bottleneck node defined as the node where packet loss occurs. It was discussed in [1] that TCP connections experience multiple bottlenecks in the *slow start* [10] phase. It is straightforward to extend the analysis and its numerical implementation to include the *slow start* phase or *time-out*. However in the rest of the paper we do not consider these issues further.

Using the fluid abstraction, we can describe the behavior in time of a TCP Reno source as follows: Each source has a state-variable associated with it, the window size $W(t)$, which represents the maximum amount of unacknowledged fluid that can be transmitted in time t. Whenever the window permits, the source transmits fluid into the network at the rate $i(t)$.

Further, let
$a(t)$ = ACK Rate; $n(t)$ = NAK Rate,
$c = \mu$ = Service rate; B = Buffer size at bottleneck node,
$q(t)$ = Queue length at the bottleneck node,
P = Packet size for the connection,
d = Return path propagation delay,
d_0 = Forward path propagation delay up to the bottleneck node,
c_0 = Maximum input transmission rate at bottleneck node,
$T(t)$ = Total amount of unacknowledged fluid.

TCP Reno's source dynamics can be described as follows: The window W increases by $1/W$ for every packet successfully acknowledged. Therefore, for the *congestion avoidance* phase [10] when $W(t) = T(t)$ under the fluid approximation we can state that,

$$\frac{dW(t)}{dt} = \frac{P}{W(t)} \cdot a(t-d). \tag{1}$$

When a packet loss is detected, the window halves. In the fluid case, when negative acknowledgments (NAKs) equivalent to one packet (representing P units of fluid) has accumulated, the window is halved. Mathematically, we can state that,

$$\int_{t_{n-1}}^{t_n} n(t-d)dt = P \Longrightarrow W(t_n) \leftarrow \frac{W(t_n^-)}{2}, \tag{2}$$

where t_{n-1} is the time when the previous packet loss occurred. This completes the specification of the TCP Reno source.

In the equivalent MATLAB Simulink model that we use, the block representing the bottleneck network node in Figure 2 has the functionality of a router and TCP sink incorporated into it, for simplicity. It is possible to do this, because the TCP sink maintains no state relating to congestion control. The router has a maximum buffer size B and a service rate μ. In general both quantities will vary in time as the network resources are shared among multiple connections according to some kind of sharing policy, e.g. Generalized Processor Sharing (GPS) for the service and Virtual Partitioning (VP) [4, 16] for the buffer.

The queue occupancy $q(t)$, evolves as follows:

$$\frac{dq(t)}{dt} = i(t-d_0) - \mu, \text{ if } q(t) < B. \tag{3}$$

The specification for the TCP sink/router is very simple:

$$a(t) = \mu \text{ if queue is non-empty,} \tag{4}$$

$$a(t) = min(\mu, i(t-d_0)) \text{ otherwise.} \tag{5}$$

$$n(t) = i(t-d_0) - \mu \text{ if } q(t) > B, \tag{6}$$

$$n(t) = 0 \text{ otherwise.} \tag{7}$$

Incidentally, the TCP sink must satisfy the condition that fluid is conserved, i.e., all transmitted fluid must be accounted for with either ACKs or NAKs. Formally, this is expressed as:

$$lim_{T\to\infty} \frac{1}{T} \int_0^T [i(t) - a(t-d) - n(t-d)]dt = 0. \tag{8}$$

2.2 Prototype for a TCP analysis tool

As mentioned in Section 1, we have developed our model into a prototype for a software tool for studying TCP's congestion control algorithm. The platform we chose for this purpose is MATLAB's SIMULINK utility. One reason for choosing this tool is that it has features appropriate for modeling feedback control type of scenarios. It provides excellent GUI features, and also makes it possible to extend this framework to model other types of congestion control, e.g. TCP Vegas [3].

We believe that this could serve as a prototype for a general analysis tool for TCP. Such a tool could be expected to be more efficient than packet by packet simulation schemes that are common today. One reason to expect such efficiency is the fact that we concentrate on only the transport layer protocol, so we can simplify the lower layers of the network architectures.

0-7803-7097-1/01/$10.00 ©2001 IEEE

However, there is also another interesting source of efficiency: the fluid model itself. We already mentioned that for high-speed networks, we could usually work on a granularity level larger than a single packet. This is not an accident but the result of the fact that most network protocols have been designed with good scaling properties in mind. The exception is in the TCP multiplicative decrease algorithm, where the TCP source dynamics is extremely sensitive to a single packet loss. If we are numerically integrating the equations to solve for the network dynamics, we could use very coarse time-steps for most of the simulation and only reduce the granularity when the queue is close to overflow for instance. However, it is not clear how this would scale for a large number of connections. This issue has also been addressed in [6].

3 Numerical Results

3.1 A Single Node Model

In this section, we model a one hop network where a source is connected to the destination by a router and the link rate between the source and the router is more than the link rate between router and the destination, i.e. the router is the bottleneck. This model is shown in Figure 1.

For Figure 5 we have the following parameter values:$c = 5$, $d = 1$, $B = 3$, $c_0 = 7.5$. We combined the backward and forward delay parameters into a single round trip time (RTT) parameter, d. In other words, we set $d_0 = 0$ for the rest of the paper. For a packet size of 100 bytes this is equivalent to a single connection single hop network with a bottleneck router with capacity 100 kbps, buffer size 75 packets and a path propagation delay of 1 second if we assume that one fluid unit is equivalent to 25 packets and 1 fluid time unit is equivalent to 1 second. The corresponding network is also simulated using the ns simulator [7] and its window and buffer evolution curves are plotted in Figure 6. We can see from the figures that the window evolution curves and the queue evolution curves are close. The throughput we get from both the ns simulation and numerical calculations is 96 kbps.

We used our model to run some simple tests to see how the connection throughput degrades with decreasing buffer capacity. We repeated this test for several different values of the connection's RTT (Figure 7) and service rate (Figure 8). It confirms what earlier theoretical studies predicted [13, 2], that as long as storage capacity is more than a certain minimum of about half the bit-pipe capacity, the throughput is close to saturation value, i.e. bottleneck router service rate. However, we noted that even when the router buffer capacity is about 0.3 times bit-pipe capacity, the throughput is close to 80 %. The normalized buffer is defined as the ratio of bottleneck buffer size to the bit-pipe capacity for the connection; it is a function of buffer size, connection RTT and the router's service rate.

3.1.1 Complete Buffer Sharing (CS) and Generalized Processor Sharing (GPS)

Here we extend the model in the previous subsection to complete buffer sharing (CS) type of buffer management between two connections. The sessions get GPS (Generalized Processor Sharing) [14] service at the router. Each session shares the total amount of buffering available at the router.

We present results (Figure 3) of two such sessions with parameters $c_1 = c_2 = 3, B = 30, c_0 = 10, d_1 = 4, d_2 = 8$. We again note that the large difference in the window size of the two sessions. Note that the TCP window evolution is synchronized. The window synchronization phenomena came out as a natural consequence of our analytical model. Thus the session with longer round trip time whose window grows slowly fails to achieve a throughput as high as the session with shorter round trip time.

3.1.2 Complete Buffer Partitioning (CP) / Virtual Buffer Partitioning (VP) and Generalized Processor Sharing (GPS)

Next we extend the previous model with the sessions getting GPS (Generalized Processor Sharing) [14] service and separate logical output queues [5] at the router. Each session has its own dedicated buffer space in the router, corresponding to the complete buffer partitioning (CP) case.

We present results (Figure 9) of two such sessions with parameters $c_1 = c_2 = 3.5, B_1 = B_2 = 20, c_0 = 7.5, d_1 = 5, d_2 = 0.5$. For a packet size of 500 bytes, this is equivalent to a two connection single hop network with a bottleneck router with capacity 140 kbps, total buffer size 40 packets and a path propagation delay of 1 second and 100 ms, respectively, if we assume that one fluid unit is equivalent to 1 packet and 1 fluid time unit is equivalent to 0.2 second. The corresponding network is simulated using the ns simulator and its window evolution curves are plotted in Figure 10. The throughput we get from ns simulation is 56 kbps and 83 kbps for the connection with large and small round trip times respectively and from the numerical calculations we get 55.9 kbps and 83.1 kbps respectively. We believe the difference between the peak values of the window evolution curves

0-7803-7097-1/01/$10.00 ©2001 IEEE

between the numerical model and ns is dependent on the initial values of queue lengths, windows and fluid in flight for the numerical calculations, which we are manually setting. In ns the corresponding quantities for the *congestion avoidance* phase of TCP Reno are determined in the initial *slow start* phase.

Note that complete buffer partitioning helps to achieve window desynchronization and improved fairness of throughput between two connections. This phenomenon is captured in our mathematical model.

Under the virtual partitioning (VP) policy [4, 16] we let the queue lengths make excursions which are above their own thresholds as long as the total queue occupancy is below the total buffer size. Once the total queue occupancy exceeds the total buffer size the fluid of the queue which exceeded its own threshold starts getting marked [8] (from the head of the queue [12]). We also plot the window and queue evolutions of a specific case where we have $c_1 = c_2 = 3.5, B_1 = B_2 = 20, c_0 = 7.5, d_1 = 5, d_2 = 0.5$ in Figures 11 and 12.

Buffer management does not help much with connections whose round trip times are not so different. We get this result for both complete partitioning (CP) and virtual partitioning (VP) case. For example, for $c_1 = c_2 = 5, B_1 = B_2 = 20, c_0 = 7.5, d_1 = 6, d_2 = 0.6$ we obtained throughput for connection 1 as 4 fluid units per fluid time unit and throughput for connection 2 as 5.9 fluid units per fluid time unit. For the same network, under the complete sharing scenario, with a total buffer size of 40, the throughput for connection 1 was 3.3 fluid units per fluid time units and throughput for connection 2 was 6.5 fluid units per fluid time units. So we obtained a significant throughput improvement. But with delays of $d_1 = 6, d_2 = 5$ we have a throughput of 4.2 fluid units per fluid time unit for connection 1 and 4.8 fluid units per fluid time unit for connection 2 in the complete partitioning case and 4.15 and 4.85 fluid units per fluid time unit for the complete sharing case, respectively. Therefore, our conclusion is that it pays to insulate connections with very different RTTs from each other while the connections of comparable RTTs can share resources with each other.

A general way to do this would be to use a Class Based Queuing strategy, where the traffic is aggregated into two classes depending on whether their RTT is greater or less than a threshold value. One very good justification for the above policy is the following: it has been observed in practice [9] that network traffic falls broadly into two categories: one type of traffic is largely localized within a geographical or administrative domain and the other type consists of general wide area traffic, e.g. WWW based traffic. Interestingly, there is evidence to suggest that the wide-area part of the traffic has an RTT distribution with a long tail.

Thus, we would have a natural threshold, based on observed usage patterns for identifying long RTT connections. However, in practice, it may not be easy for a router to estimate RTT values of different flows, in which case we could make use of simpler metrics. For example, the TTL field in the IP header would provide a straightforward indication of the path length. In addition, the router could use its own routing database to identify some destinations as long-RTT. We believe that while imperfect, simple estimates like the above, could provide a reasonable basis for classifying flows and help improve TCP Reno's performance with a minimum amount of complexity in the network.

In other words, it would represent a reasonable compromise between best-effort strategies with associated fairness and synchronization problems and per-flow allocation strategies which have scalability problems.

One important advantage of Virtual Partitioning (VP) is that it offers us multiplexing gain in terms of the minimum normalized buffer that is required for supporting reasonable throughput for a specific connection. To study this, we have two connections with virtual partition buffer allocation of $B_1 = B_2 = 20$, $c_1 = c_2 = 5$, $d_2 = 5$. We vary d_1 (Figure 15) so that normalized buffer allocation of connection 1, i.e., $B_1/(B_1 + c_1 \cdot d_1)$ varies from 0.2 to 0.8 and we calculated the normalized throughput which is throughput divided by c_1. We also plotted this curve for $c_1 = 3$. In all these experiments we find that a normalized buffer size of 0.2 is good enough to support a normalized throughput of 0.8. In another set of curves we vary the value of d_1 from 3 to 12 (Figure 16) and for most of the cases we find that normalized buffer required to support a normalized throughput of 0.8, plotted on the vertical axis, is close to 0.2. Recall that in a single connection case this required normalized buffer was at least 0.3.

3.2 Multiple Router Case

We now consider a multiple router case (Figure 4). We have implemented in SIMULINK a case where multiple connections are passing through multiple routers. We have implemented GPS with CP at each router. At each router buffer allocation per connection is 10 units of fluid and the capacity allocation per connection is 5 units of fluid per fluid time unit. The connection that is passing through both the routers (connection 1) has a total backward path delay of 4.1 fluid time units and the connection that is passing only through first router (connection 2) has a backward path propagation delay of 2.5

0-7803-7097-1/01/$10.00 ©2001 IEEE

fluid time units and the one that is passing through only the second router (connection 3) has a backward path propagation delay of 1.5 fluid units per fluid time units. In Figure 13 and 14 we plotted the window and queue length evolution of the three TCP connections. In the plot we only show the queue length evolution of connection 1 in the first router and the queue length evolution of other two connections in their respective routers. The throughputs we get are 4, 4.5 and 5.3 fluid units per fluid time units for connections 1, 2 and 3 respectively.

3.3 Multiplexing gains with 4 connections

As an example of the multiplexing gains that could be achieved by our class based policy based on propagation delays, we present a simple 4 connection case. Here, there are two connections with long RTTs (3.0 and 3.5 fluid time units) and two connections with short RTTs (1.1 and 1.5 fluid time units) sharing a bottleneck. We assign GPS rates of 5.0 fluid units/fluid time units for each connection and then compare the minimum amount of buffering needed with the constraint that all connections receive a fair share of service (which we fix as 4.7 fluid units/fluid time units). We consider two cases. In one case, all connections share the buffer space and in the other case, the long RTT connections and short RTT connections have separate buffers. Our simulation shows that in the first case, B_{min} is of the order of 55 packets compared to 30+10=40 packets in the second case. We note that the separate buffer case gives results similar to the 2-connection case examined earlier. For the shared buffer case, some throughput results are plotted in Figure 17.

4 Conclusions and Future Work

In this paper we presented a mathematical model for TCP Reno using systems of delay differential equations. We have shown for several simple cases that our model matches well with the ns simulator output. We intend to further refine our model for quantitative accuracy and for speed of simulation. We have also proposed a new class based queuing (CBQ) approach to address the throughput unfairness problem. We intend to implement and study this approach.

We have also shown that by using appropriate buffer and bandwidth allocation policies we can mitigate the throughput unfairness problem among TCP connections with different round trip times, and also reduce buffering requirements in the network.

We believe our approach will help engineer a fluid simulator for TCP based on analytical models. This will make it possible to optimize TCP throughput by using class based buffer management policies in the network [16, 11].

References

[1] C. Barakat and E. Altman, "Analysis of TCP with Several Bottleneck Nodes," INRIA Report Number 3620, February 1999 and also in *GLOBECOM 1999.*

[2] T. Bonald, "Comparison of TCP Reno and Vegas via fluid Approximation," INRIA Report No. 3563, November, 1998.

[3] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE J. on Selected Areas in Communications*, Vol. 13, pp.1465-1480, 1995.

[4] I. Cidon, L. Georgiadis, R. Guerin and A. Khamisy, "Optimal Buffer Sharing", *IEEE JSAC*, vol. 13, No. 7, pp. 1229-1241, September 1995.

[5] G. J. Foschini and B. Gopinath, "Sharing Memory Optimally", *IEEE Transactions on Communications*, vol. COM-31, no. 3, pp. 352-360, March, 1983.

[6] Y. Gao, W. B. Gong and Don Towsley, "Time-stepped Hybrid Simulation (TSHS) for Large Scale Networks," *Proceedings of INFOCOM 2000*, vol. 2, pp. 441-450.

[7] http://www-mash.cs.berkeley.edu/ns/

[8] http://www.aciri.org/floyd/ecn.html

[9] http://www.caida.org/tools/measurement/skitter/RSSAC/

[10] V. Jacobson, "Congestion Avoidance and Control", *Computer Communication Review*, 18(4):314-329, August 1988.

[11] V. P. Kumar, T. V. Lakshman, D. Stiliadis, "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet", *IEEE Communications Magazine, May 1998.*

[12] T. V. Lakshman, A. Neidhardt, and T. J. Ott, "The Drop from Front Strategy in TCP and in TCP over ATM," *in Proceedings of INFOCOM, 1996*, pp. 1242-1250.

0-7803-7097-1/01/$10.00 ©2001 IEEE

[13] T. V. Lakshman and Upmanyu Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Trans. Networking, June, 1997.*

[14] K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", *IEEE/ACM Transactions on Networking*, vol. 1, No. 3, pp. 344-357, June 1993.

[15] J. Postel, "Transmission Control Protocol", RFC 793, September 1981.

[16] B. Suter, T.V. Lakshman, D. Stiliadis, and A. K. Choudhury, "Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-flow Queueing", *IEEE Journal on Selected Areas in Communications: Special issue on Next Generation IP Switches and Routers*, vol. 17, No. 6, pp. 1159-1169, June 1999.

[17] W. Richard Stevens, "TCP/IP Illustrated, Volume 1: The protocols", Addison Wesley, January 1994.

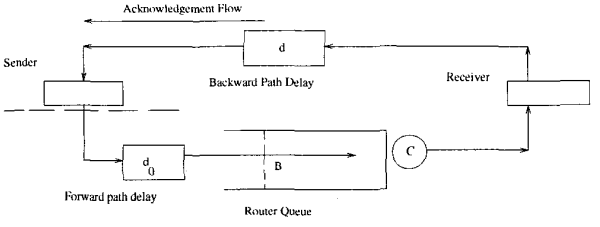

Figure 1: The queuing Model for a Single Hop One Session Network

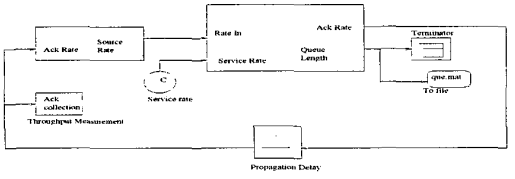

Figure 2: The Simulink Model

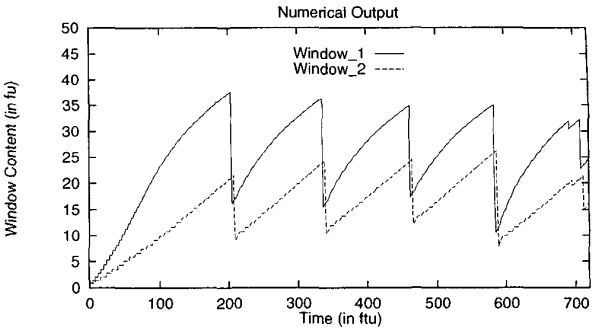

Figure 3: Numerical Output: Window Evolution

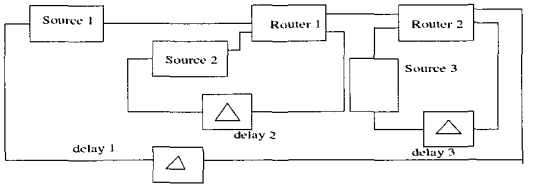

Figure 4: Multiple Connections through multiple routers.

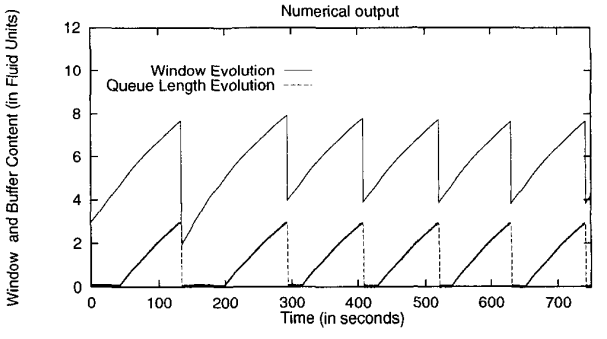

Figure 5: Numerical Output: Window Evolution

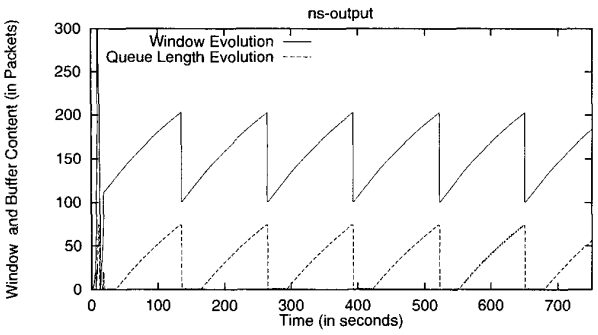

Figure 6: ns Output: Window Evolution

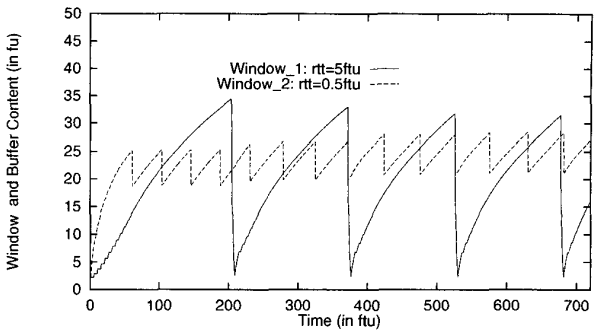

Figure 9: Numerical Output: Window Evolution

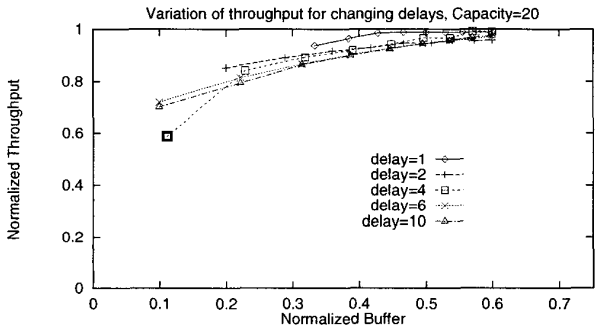

Figure 7: Normalized throughput vs. normalized buffer

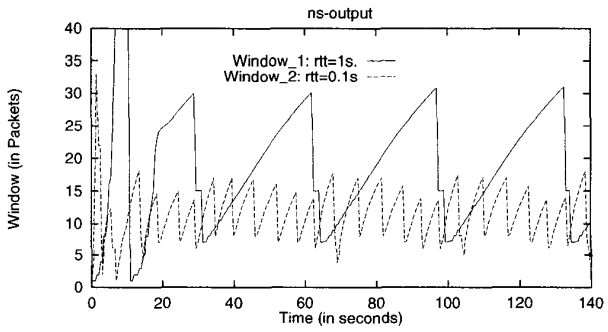

Figure 10: ns Output: Window Evolution

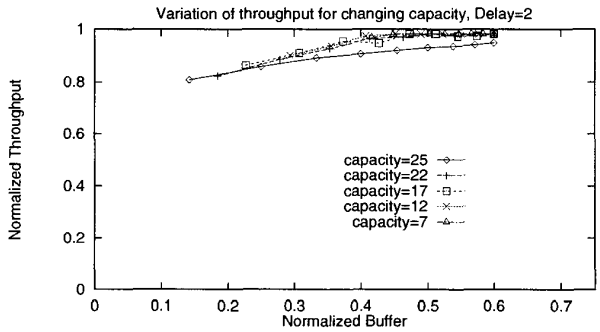

Figure 8: Normalized throughput vs. normalized buffer

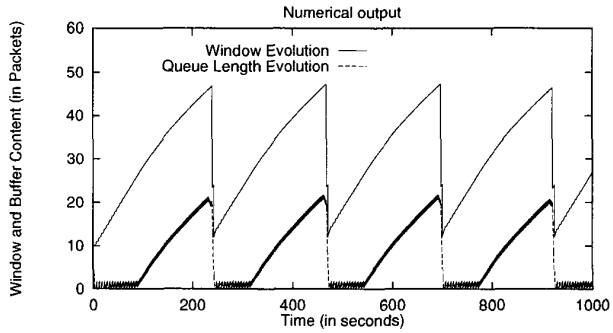

Figure 11: Long RTT connection (VP+GPS)

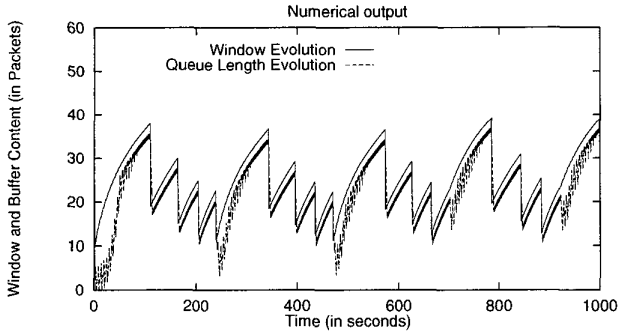

Figure 12: Short RTT connection (VP+GPS)

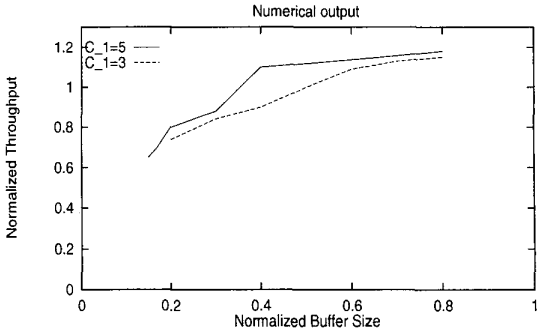

Figure 15: Normalized buffer vs. normalized throughput

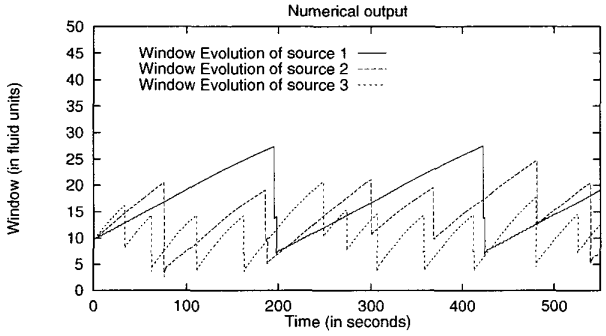

Figure 13: Window Evolution of TCP connections in Multiple Router

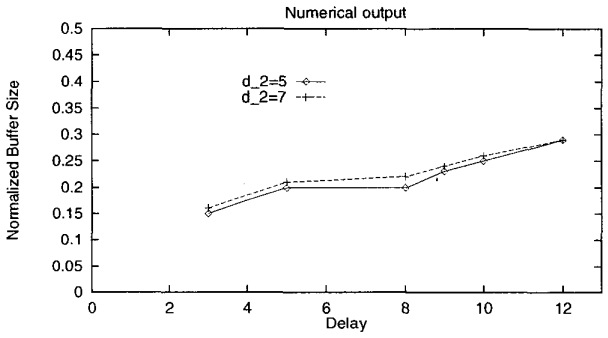

Figure 16: Normalized buffer vs. Delay

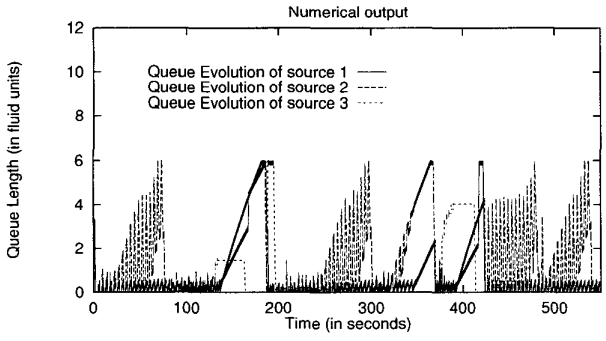

Figure 14: Queue Evolution of TCP connections in Multiple Router

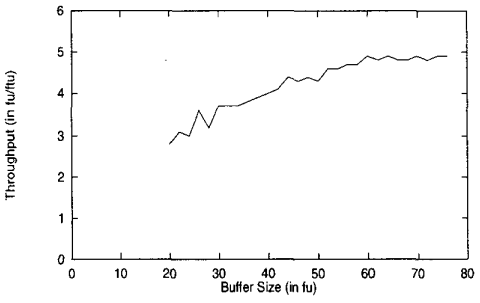

Figure 17: Throughput variation for shared buffer with 4 connections

0-7803-7097-1/01/$10.00 ©2001 IEEE