Congestion Control in TCP: Performance of Binary Congestion Notification Enhanced TCP Compared to Reno and Tahoe TCP*

Dorgham Sisalem
GMD-Fokus, Berlin
dor@fokus.gmd.de

Henning Schulzrinne
Columbia University, New York
schulzrinne@cs.columbia.edu†

Abstract

In previous work, we proposed to enhance TCP's congestion control mechanisms using binary congestion notification (BCN). With this combined scheme called BCN-TCP, congestion is indicated both by packet losses and by having switches inform sources about their congestion state by setting a congestion bit. Unlike other studies, our approach maintains the basic elements of TCP's congestion control mechanisms and models the response of TCP to the congestion bit to that of the ATM available bit rate service (ABR). We compared the performance of BCN-TCP to the versions currently implemented in hosts (Tahoe and Reno TCP). However, BCN-TCP only offered significant performance improvements if part of the path contained ATM ABR services.

In this paper, we present further results on the performance of BCN-TCP when used in heterogeneous environments, i.e., Internet paths where only some end systems support BCN-TCP. We also consider the fairness between long and short-haul connections and discover that BCN-TCP treats long-haul connections worse than Tahoe and Reno TCP. We investigate the cause of this unfairness and suggest possible remedies.

1 Introduction

One of the main problems with transport protocols, such as TCP, that use packet losses as a congestion indication is that congestion is only handled with when it is already too late. That is, sources react only when the switch buffers reach their limits and the switches start dropping packets.

Upon detecting the loss of a packet, 4.3BSD Reno TCP retransmits the lost packet and reduces the transmission window to half of its current size. Older versions, e.g., 4.3BSD Tahoe, reduce the transmission window to only one packet. The retransmission and severe window reduction actions taken in TCP after detecting a loss lead to the waste of bandwidth and underutilisation. To overcome this drawback we introduced in previous work [1] an enhanced version of TCP that uses binary congestion notification (BCN) for congestion indication. In contrast to other approaches, our scheme maintains the basic TCP behaviour and adapts TCP's response to BCN based upon the source behaviour proposed for connections using ATM's available bit rate (ABR) service.

In the next section, some related work to this subject is introduced and some previous work describing the implementation of BCN-TCP is summarised. The effects of using BCN-TCP in a heterogeneous environment is then described in Sec. 3. Finally, Sec. 4 handles the fairness issue and describes bandwidth distribution results achieved with Tahoe-, Reno- and BCN-TCP when long and short-haul connections share congested links. The unfairness of BCN-TCP towards long haul traffic is investigated and two approaches for improving its fairness are described briefly.

2 Related Work and Previous Proposals

In this section we describe some of the work upon which we built our proposal for using binary congestion notification methods in TCP and a short introduction to our scheme.

2.1 Congestion Control in TCP

First, we briefly describe the congestion control mechanisms used currently in TCP, i.e., slow start, congestion avoidance, fast retransmission and fast recovery.

There are two phases to the window adjustment algorithm in TCP:

- **Slow start**: The slow start algorithm adds another window to the TCP source, a congestion window (cwnd) that is measured in packets, i.e., in units of the

*This work was funded in part by the Commission of European Communities (CEC) under project R2116 TOMQAT.

†This work was done while the author was still with the GMD Fokus.

size of the maximum packet size possible for this connection. When a new connection is established or after a packet loss, this window is set to the size of one packet. Each time an acknowledgement is received, the cwnd is increased by the size of the acknowledged packets. The source can now transmit up to the minimum of the cwnd and the window size advertised in the acknowledgement packets.

- **Congestion avoidance**: Through the exponential increase of the sending rate caused by the slow start algorithm, the capacity of the network will be reached at some time and an intermediate router will start discarding packets. In order to avoid this, the cwnd is increased using the slow start algorithm until it reaches the slow start threshold (ssthresh), the setting of which is described below. Beyond that, TCP goes into the congestion avoidance phase and cwnd is increased by 1/cwnd for each acknowledgement received. This way the congestion window increases by at most one packet each round trip instead of doubling it as was the case for slow start.

In the early versions of TCP, packet losses were indicated through timeout mechanisms. That is, whenever a packet was sent a timer was set. If the timer expired before receiving an acknowledgement for the sent packet, the packet was considered to be lost and had to be retransmitted. With the *fast retransmission* scheme, a packet X_n is additionally considered to be lost if a source receives a number of duplicate acknowledgements, usually three, for packet X_{n-1}. This significantly reduces the time required for detecting the loss.

In the Tahoe version, TCP reacts to a packet loss - detected by either the fast retransmission scheme or after a timeout- by setting a variable called the slow start threshold (ssthresh) to half of the current congestion window (cwnd) and decreasing cwnd to 1. After receiving the acknowledgement for the retransmitted packet the source enters the slow start phase and the transmission window can be increased exponentially with the slow start scheme while (cwnd $\leq$ ssthresh). Afterwards, the sender enters the congestion avoidance phase and the window is increased by 1/cwnd for each received acknowledgement.

In the Reno version, the *fast recovery* scheme is additionally used. With this scheme, the source retransmits the lost packet after receiving three duplicate acknowledgement packets and sets ssthresh to half of the current congestion window (cwnd). Cwnd is then set to ssthresh plus 3 times the packet size. Each time another duplicate acknowledgement is received, cwnd is incremented by the packet size and a packet is transmitted if allowed by the new value of cwnd. When the next acknowledgement arrives that acknowledges new data cwnd is set to ssthresh, instead of 1 as was the case with Tahoe-TCP and the congestion avoidance mode is entered. If loss was detected with the timeout scheme then the same procedures used in the Tahoe version are used here as well. For a more detailed description of the different congestion control mechanisms of TCP, see [2].

2.2 Congestion Control in ATM

With the introduction of ATM technology, the issue of integrating TCP and ATM in a heterogeneous environment will play an important role as many of the current applications are based on the currently widespread TCP/IP infrastructure. Different studies [3, 4, 5, 6] have shown that TCP's performance over ATM is severely degraded due to the segmentation of the TCP packets into ATM cells. Dropping only one cell in the ATM level results in the loss of a whole TCP packet. Thereby, even a small cell loss ratio results in a large data loss ratio in the TCP level [7].

In order to allow for a smooth integration of TCP and ATM, it might be profitable to benefit from the congestion control schemes proposed for ATM and to adapt the TCP congestion control mechanisms.

The ATM Forum suggested the use of a rate control based mechanism as the standard mechanism for congestion control in ATM. The mechanism works as follows: After every N[1] sent data cells, the source sends a special cell, the so-called resource management (RM) cell. The source writes its current rate and the rate it would like to use in those cells. At the intermediate switches the fair rate share for each connection is determined and appropriate congestion information is written into the RM cells. The destination end system returns then the RM cells back to the source. Using the congestion information noted in the backward RM cells the user network interface can use a traffic policer or shaper to control the sending rate of the source.

Depending on the switch used, two kinds of congestion information can be conveyed back to the source:

- **Explicit rate indication**: The switches write the fair bandwidth share of the connection directly into the RM cells. At the destination, the RM cells are turned around and sent back to the source that should change its rate according to the rate noted in the RM cells.

- **Intelligent marking**: Connections that advertise in their forward RM cells a current rate that is higher than the calculated fair rate share at the switch are signalled to reduce their rates. This is done by setting a congestion indication (CI) bit in the backward RM cells that are sent back to the source by the destination. Upon receiving a RM cell with CI set to congested, the source end system has to multiplicatively reduce its rate. Otherwise, it is entitled to increase its rate additively.

[1] N is usually set to 32 [8]

2.3 Binary Congestion Control in TCP

In TCP/IP based networks congestion is indicated by dropping packets at congested routers. Packets are dropped when the queue of the router reaches its limit (drop tail scheme).

To handle congestion situations before packets actually get dropped several proposals for using binary congestion control have been made. With such an approach, routers that have the capability of detecting incipient congestion can just mark the arriving packets as congested instead of discarding them. The destination copies the value of the congestion bit of the received packets into the acknowledgement packets sent back to the source. The source then changes its transmission window in accordance with the value of the congestion bit.

2.3.1 Previous Proposals for Binary Congestion Control in TCP

Probably the most important work done in the area of binary congestion notification in TCP was done by Sally Floyd [9]. In her paper [9], she proposes guidelines for TCP's response to binary congestion notification (BCN) that can be summarised as follows:

1. TCP's response to BCN should be similar over longer time scales to its response to dropped packets or timeouts.

2. Over smaller time scales TCP's response to BCN can be less conservative than its response to lost packets.

3. TCP should react to BCN at most once per round trip time.

4. The response to BCN does not trigger the sending or retransmitting of any data packets.

For the simulations done using this approach TCP was modified as follows [9]: Upon receiving a BCN message and if no response to congestion had been made in the last round trip time the usual congestion control mechanisms are applied, however, without retransmitting any data. That is, the transmission window is decreased to 1 in the case of Tahoe-TCP and to half of its current value for the Reno implementation. During the next round trip time all BCN messages are then ignored. If three duplicate acks are received after responding to a BCN message the lost packet is retransmitted and no further actions are taken.

If three duplicate acks are received and no BCN messages have been received during the last round trip time, the TCP source follows the usual procedure of fast retransmission and fast recovery.

In another work by Calhoun [10], the importance of using binary congestion notification is stressed as well. The author proposes a response scheme similar to older bit marking schemes that were proposed for ATM [11]. In this scheme, the source decreases its window by a factor of 0.625 if within a round trip time a packet with a set congestion bit is received. If within two round trip times no congestion indications were received the window can be increased by one.

This proposal still needs further development and testing. The author does not give any explanation for choosing the decrease factor of 0.625 or the way this multiplicative decrease can be integrated with TCP's congestion control mechanisms.

2.3.2 Our Proposal for Binary Congestion Control in TCP

In a previous paper [1], we described the usage of binary congestion notification in TCP. With this enhanced model, IP packets could have a congestion bit either as an additional option or directly in the header. Packets are sent with the congestion bit set to 0, i.e, not congested. When passing a congested switch the congestion bit is set to 1. The value of the congestion bit of the arriving packets at the receiver is then included in the acknowledgement packets sent back to the sender. If an acknowledgement was received with the congestion bit set and no packet losses were seen during the last round trip time, the source enters the binary congestion notification mode. During this mode the source reduces its congestion window multiplicatively by a multiplicative decrease factor (MDF) for each received acknowledgement with the congestion bit set. The source stays in this mode until the acknowledgement for the packet sent just before receiving the first marked acknowledgement arrives. If three duplicate acks were received and the source was not in the binary congestion notification mode normal TCP action are taken, i.e, retransmission and window reduction. For detailed description of the implementation of BCN-TCP, see [1].

In this, the scheme resembles the congestion control schemes using bit marking proposed for ATM data services [12]. So, for the case of TCP running over an ABR connection this resemblance will allow better integration of ATM and TCP as both schemes would react in a similar way to the same congestion information. Also, handling congestion in advance reduces the amount of buffer required in the intermediate switches and the delay caused by retransmissions.

3 Introducing BCN-TCP in a Heterogeneous Environment

Using different simulation models we investigated in [1] the performance of BCN-TCP compared to Tahoe-TCP and Reno-TCP. Our results revealed that, just as was the case for the model proposed by Sally Floyd in [9], the increase in throughput performance was marginal when compared with Tahoe-TCP or Reno-TCP. However, a noticeable gain in performance was seen when integrating BCN-TCP with ABR. Using TCP over ABR leads in general to multiple packet losses. This results especially for Reno-TCP in severe performance degradation. Using BCN-TCP these multiple losses are avoided by adjusting TCP's transmission window in accordance with the available rate reported by the network.

In this section, we investigate the effects of using BCN-TCP in a heterogeneous environment with a part of the connections using BCN-TCP and the other part using some other version, here, Reno-TCP. As a metric, we use the effective link utilisation, i.e., the utilisation resulting from the effective TCP throughput.

3.1 The Simulation Environment

We have chosen a very simple test topology consisting of nine connections sharing a single link. All the hosts have the same parameters and share the same buffer of the congested link. Therefore, we can expect all connections to have the same bandwidth share. The shared link has a propagation delay of τ seconds. To simulate the behaviour of TCP over a LAN as well as a WAN distance we set the value of τ to 5 μsec and 5 msec respectively. The links rates were set to 10 Mb/s.

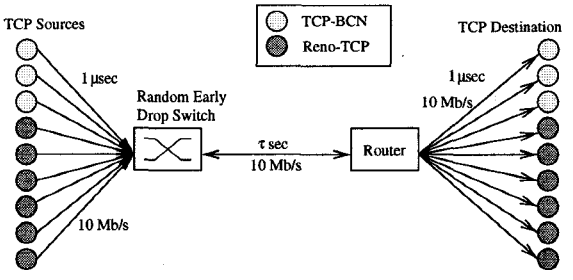


Figure 1. Simulation test topology

For simulating the behaviour of TCP we used the PTOLEMY [13] simulation tool to write a TCP model [14]. We implemented the Tahoe-TCP and Reno-TCP versions described in Sec. 2.1. Both versions were enhanced to support windows larger than the 64 Kbytes currently allowed [15]. As the round trip times (RTT) values in our simulations are in the order of just a few milliseconds, the coarse-grain timer used in Unix TCP (typically, a granularity of 500 msec is used) would result in imprecise timeout values. Therefore, we added an extra parameter with which the user can determine the appropriate clock granularity. Here, we used a granularity of 200 msec which considering the propagation delays used in our simulations, the buffer size and link bandwidth is still large enough for avoiding early timeouts yet small enough for ensuring fast reaction.

The sources are persistent ones, i.e., they always have some data to send whenever the transmission window allows them to do so. The packet size was set here to 500 bytes. The multiplicative decrease factor was set to 0.975. This value yielded the best results in the simulations we made in [1]. Values smaller than 0.975 resulted in large window reductions and low link utilisation. Using larger values resulted, on the other hand, in slow reactions to congestion and packet losses.

The switch is a random early drop (RED) gateway as proposed by Floyd and Jacobson [16]. A RED gateway detects incipient congestion by computing the average queue size[2]. When the average queue size exceeds a preset minimum threshold (Min_{th}) the switch either marks or drops each incoming packet with some probability, depending on the used TCP version. Exceeding a second maximum threshold (Max_{th}) leads to the marking of all arriving packets when using BCN-TCP. For the case of TCP without the binary congestion notification the packets are not marked but dropped.

In our simulations, the minimum threshold is set to 1/20 of the buffer size and the maximum threshold is set to six times the minimum threshold. The buffer was set to 512 Kbytes. After running different test simulations, we found that these values were most suited for our simulation configuration in terms of buffer requirements and link utilisation.

Compared to the drop tail scheme higher utilisation can be reached with this scheme, fairer bandwidth distribution is achieved and global synchronisation can be avoided [16]. Note that for the binary congestion notification scheme, any other switch algorithm could be used that can detect incipient congestion [17].

3.2 Simulation Results

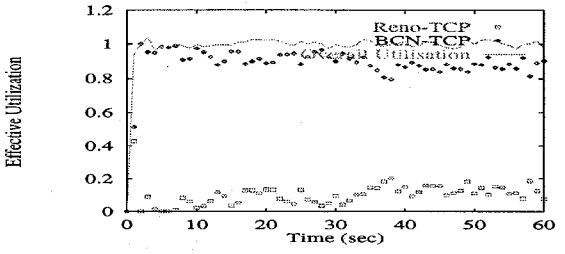
In this section, we investigate the effects of mixing connections using binary congestion notification for congestion control and Reno-TCP connections.

In Fig. 1 only a third of the connections is using binary congestion notification. While the link utilisation remains

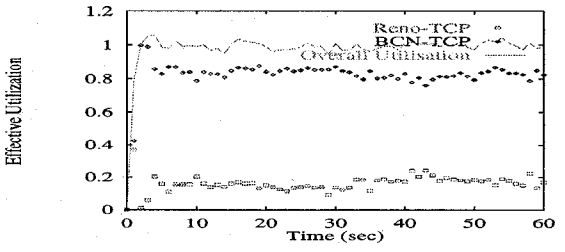
[2]The RED algorithm uses a low pass filter to calculate the average queue size. The low pass filter is of the exponential weighted moving average (EWMA) type:

$$\text{average} = (1 - w_q)\text{average} + w_q q$$

with w_q as a constant weight and q as the actual queue length.



(a) Propagation Delay = 5 μsec



(b) Propagation Delay = 5 msec

Figure 2. Utilisation achieved with only a third of the connections using BCN-TCP

as high as 90%, the bandwidth distribution is completely unfair. Even though only a third of the connections uses BCN-TCP, these connections have an effective throughput four times as high as that of Reno-TCP, see Fig. 2. However, this is not very surprising. In our configuration, all connections pass through the same random early drop switch and the drop or marking probability is calculated for the entire traffic with the same parameters. This means, that the probability of dropping a packet is the same as that for marking one. As dropping a packet results in a large window reduction and marking a packet results in only a relatively small multiplicative decrease, it is obvious that the Reno-TCP connections will suffer much more during congestion phases.

To overcome this problem, we propose to modify the switch algorithm in such a manner that the probability of marking a packet is higher than that of dropping one. This would be the appropriate approach to use with drop tail switches as well. In those switches, arriving packets are dropped whenever the queue length exceeds a certain preset threshold. When using the binary congestion notification, these switches would use a marking threshold that is lower than that used for dropping the packets.

The decision to drop or to mark a packet using the RED switch is done as follows: Whenever the average queue length exceeds a minimum threshold (Min_{th}) and is lower than a maximum threshold (Max_{th}) the number of the packet to be dropped or marked (N) is determined as follows:

$$N = [R/p_b]$$

with R is from the uniform distribution on [0,1] and p_b is calculated using the average queue length and the threshold values as follows:

$$p_b = C_1 * \text{average} - C_2$$

with C_1 and C_2 constants calculated from Max_{th} and Min_{th}. To increase the marking probability compared to the probability of dropping a packet N, must be determined in the following way:

$$N = [R/\alpha * p_b]$$

with $\alpha < 1$ for Reno-TCP connections

Fig. 3 depicts the results achieved using $\alpha = 0.25$. The bandwidth is now fairly distributed, with the 6 Reno-TCP connections receiving $\approx$ 60% of the available Bandwidth. The validity of the value of α is, however, limited to this configuration and was only reached after running several simulations with different values of α. Actually using a different MDF value would already lead to a different value of α. Therefore, further work must be done on investigating the way switches should deal with mixed traffic and setting the appropriate marking probability in accordance with the passing traffic.

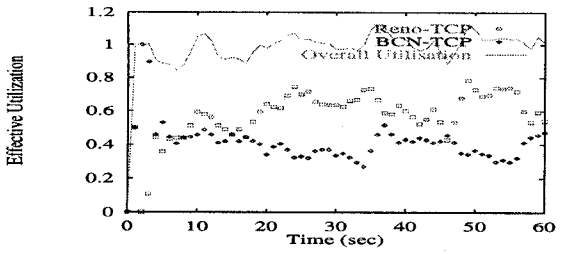
4 Fairness and Binary Congestion Notification

For testing the fairness performance of TCP with and without binary congestion notification we use a chain topology with five switches and 21 connections originating from seven sources, where each source is the origin of three similar connections. The distance between two neighbouring switches is 1000 km and the distance from a source to a switch is 0.01 km. Depending on the number of switches they have to pass, there are three kinds of traffic:

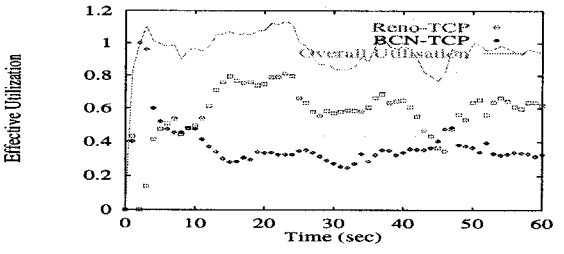
- Long distance connections: Connections starting from source 0 traverse all four links;
- Medium distance connections: Connections starting from sources 1 and 4 traverse two links;
- Short distance connections: Connections starting from sources 2, 3, 5 and 6 traverse only one link.

The switches are RED switches with a buffer size of 500 kbytes and Max_{th} set to 50 kbytes and Max_{th} set to 200 kbytes.

Again, persistent sources are used with a packet size of 500 bytes and a multiplicative decrease factor of 0.975. As



(a) Propagation Delay = 5 μsec



(b) Propagation Delay = 5 msec

Figure 3. Achieved utilisation for Reno-TCP combined with BCN-TCP traffic with the marking probability higher than the drop probability

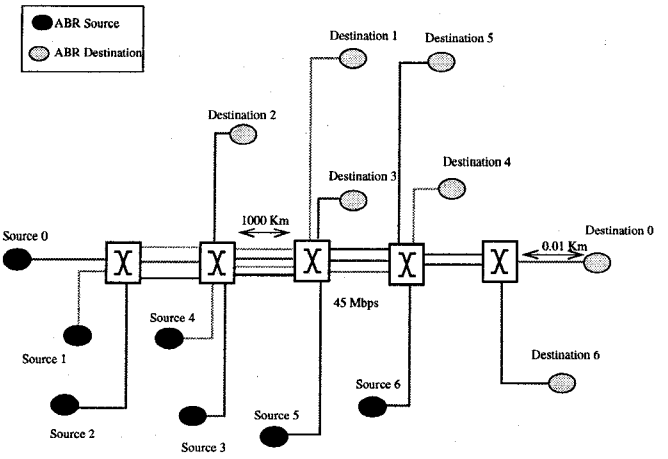


Figure 4. Test topology

Version	connection						
	0	1	2	3	4	5	6
BCN	0.2	0.63	2.5	1.8	0.64	2.5	3.1
Tahoe	0.32	0.68	2.3	1.6	0.69	2.3	3
Reno	0.36	0.7	2.3	1.6	0.68	2.2	2.9
Fair	0.83	0.83	1.66	0.83	0.83	1.66	2.5

Table 1. Average throughput in Mb/s for each connection compared to the calculated fair share with the max-min criterion

a fairness criterion we use the max-min fairness definition as it was described in the ATM Forum traffic management specification 4.0 [12]. A connection's fair bandwidth share is at the minimum

$$\frac{\text{Bandwidth of link}}{\text{Number of connections}}$$

and at most

$$\frac{B - \sum B \text{ of bottlenecked connections}}{N - \sum N \text{ bottlenecked}}$$

with B as the bandwidth and N as the number of connections. A bottlenecked connection indicates that the fair share calculated for this connection at some other link is smaller than its fair share at this link.

For a quantitative description of fairness Jain [18] suggests using the so-called fairness index.

$$F = \frac{(\sum_i x_i)^2}{n \sum_i x_i^2} \quad \text{with } x_i = \text{ratio of the actual throughput to the fair throughput}$$
$$\text{and } n = \text{the number of connections}$$

Tab. 1 contains the average effective throughput for the three schemes examined and the fair bandwidth share for each connection according to the max-min fairness criterion. The values of the corresponding fairness indexes are depicted in Tab. 2. While Tahoe-TCP and Reno-TCP achieve nearly the same fairness index of 0.86, BCN-TCP achieves only an index of 0.65. For the fairness discussion two factors have to be considered:

1. The probability of packets getting dropped or marked at congested switches increases proportionally to the number of switches a connection traverses. This factor is common among all the tested schemes here.

2. The unfairness towards long distance connections is also influenced by the way the short distance connections react to the congestion notifications. Consider a simple case of a switch with a short distance connection with a round trip time of t_{short} and a long distance connection with a round trip delay of t_{long}. During congestion periods packets of the Reno-TCP connections will be dropped and the packets of the BCN-TCP connections will be marked with the same probability. Upon receiving a congestion notification from the network, the sources reduce their transmission windows. So, after a time period of t_{short} the effects of the

version	BCN-TCP	Tahoe-TCP	Reno-TCP
Fairness index	0.65	0.87	0.86

Table 2. Fairness index for each TCP version

window reduction can be noticed at the switch. With less traffic arriving at the switch the drop or marking probability would be decreased. So, even though the traffic of the long distance connection arriving at the switch decreases only after a period of t_{long}, the drop or marking probability of this traffic will be decreased after a period of only t_{short}. Upon detecting the loss of a packet a Reno-TCP source reduces its sending window to half of its current size. A Tahoe-TCP source would even reduce its window to only 1 packet. Thereby, considerably less traffic would arrive at the switch after a period of t_{short} and the drop probability would therefore be substantially reduced. BCN-TCP connections, however, reduce their windows only by the multiplicative reduction factor after receiving a marked packet. Thereby, after a period of t_{short} the marking probability would still be high and the long distance traffic would suffer congestion for a longer time period. To tackle this problem we tried two approaches:

- First, we decreased the value of the multiplicative decrease factor from 0.975 to 0.875. This resulted in a better fairness performance and a fairness index of 0.89. However, using smaller reduction factors leads to a more oscillating traffic and possibly lower throughput, see [14].
- In our second approach each connection uses an MDF value in accordance with the number of switches it passes through [19]. That is, to compensate for the higher marking probability when passing through more switches long distance connections should reduce their transmission windows slower than connections passing through only one switch. We ran the same simulation again using different MDF values for the different connections and found that for the case of an MDF value of 0.975 for connection 0 that traverses all four switches, an MDF of 0.875 for connections 1 and 4 that traverse two switches and an MDF of 0.7 for the other connections a fairness index of 0.98 was reached.

  The simplest method to implement a hop dependent MDF would be to write the value of the time to live (TTL) field in the incoming packets into an IP option in the acknowledgement packets created at the receiver. The TTL field sets an upper limit on the number of routers through which a packet can pass. It is initialised by the sender and decremented by one by every router that handles the packet. When the field reaches 0 the packet gets discarded. By adding the TTL value of the incoming packets to the acknowledgement packets the source can calculate the number of switches passed and set its MDF value in accordance to this value. However, the relation between the number of passed hops and the value of multiplicative reduction factor needs further investigations.

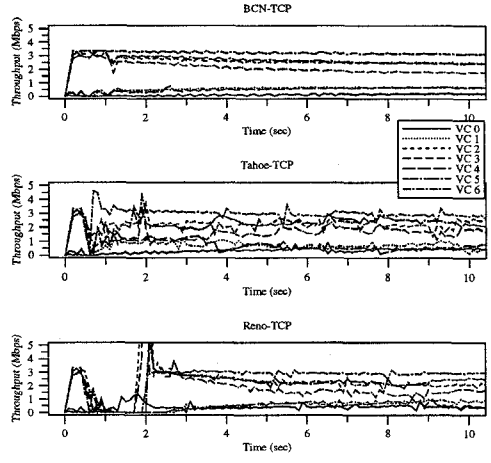


Figure 5. Throughput achieved with Tahoe, Reno and BCN-TCP

5 Summary and Open Issues

Building on previous work [1] we investigated the effects of using binary congestion control in TCP. Our results suggest that such a scheme, while being a good integration basis for TCP and ABR, suffers from some fairness problems. We have demonstrated some approaches for overcoming this problem but further investigations are still necessary.

There are still a number of open issues that must be addressed in future work:

- The efficiency of BCN schemes depends to a great extent on the efficiency of the switch algorithms. The decisions of when to mark a packet and which packets to mark play an important role in determining the fairness of the scheme and the possible link utilisation reached with it. Other than the random early drop

scheme used during this study, there have been several proposals for switch algorithms for the ABR services, see [17, 8]. However, there is still a lot of work to be done in this area.

- Further work must be done in investigating the effects of integrating connections using binary congestion notification in heterogeneous environments.We hope to extend this work with further simulations using topologies with more than one switch and to test the case when a connection passes different switches that are not all capable of bit marking.

- Different studies [19] have shown that using the explicit rate indication mode in ATM results in much better utilisation and fairness results compared to binary congestion notification schemes. It might be interesting to investigate the applicability of such algorithms for TCP/IP and the possibility of integrating the explicit rate indication scheme in TCP/IP.

References

[1] D. Sisalem and H. Schulzrinne, "Binary congestion notification in TCP," in *Conference Record of the International Conference on Communications (ICC)*, (Dallas, Texas), IEEE, June 1996.

[2] W. R. Stevens, *TCP/IP illustrated: the protocols*, vol. 1. Reading, Massachusetts: Addison-Wesley, 1994.

[3] P. P. Mishra, "Throughput degradation for TCP over ATM in the presence of traffic policing," in *Proc. of Gigabit Networking Workshop*, (Boston, Massachusetts), Apr. 1995.

[4] C. Fang, H. Chen, and J. Hutchins, "Simulation analysis of TCP performance in congested ATM LAN using DEC's flow control scheme and two selective cell-drop schemes," Tech. Rep. 94-0119, ATM Forum, Jan. 1994.

[5] A. Romanow and S. Floyd, "The dynamics of TCP traffic over ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 633–641, May 1995.

[6] R. Morris, "Detailed behavior of TCP over ATM," tech. rep., Harvard University, 1995.

[7] T. Chen, L. Jones, S. Liu, and V. K. Samalam, "Effect of ATM cell loss rate on TCP packet loss," Tech. Rep. 94-0914, ATM Forum, Sept. 1994.

[8] R. Jain, "Congestion control and traffic management in ATM networks: Recent advances and a survey." A PostScript version is available at http://www.cis.ohio-state.edu/ jain/papers.html, Jan. 1995.

[9] S. Floyd, "TCP and explicit congestion notification." A PostScript version of this document is available from ftp://ftp.ee.lbl.gov/papers/tcp_ecn.4.ps.Z, 1994.

[10] P. Calhoun, "Congestion control in IPv6 Internetworks," internet draft (work in progress), Internet Engineering Task Force, May 1995.

[11] B. Makrucki, T. Tofigh, A. Barnhart, B. Holden, J. Diagle, M. Hulchyj, H. Suzuki, G. Ramamurthy, P. Newman, N. Giroux, R. Kositpaiboon, S. Sathe, G. Garg, and N. Yin, "Closed-loop rate-based traffic management," Tech. Rep. 94-0438, ATM Forum, May 1994.

[12] S. S. Shirish, "ATM Forum traffic management specification version 4.0," Tech. Rep. 94-0013R6, ATM Forum, June 1995.

[13] S. Bhattacharyya, J. T. Buck, *et al.*, *The Almagest: A manual for Ptolemy*. Berkeley, California, 1994.

[14] D. Sisalem, "Rate based congestion control and its effects on TCP over ATM," Diplomarbeit, School of Engineering, TU Berlin, Berlin, June 1995.

[15] D. Borman, R. Braden, and V. Jacobson, "TCP extensions for high performance," Request for Comments (Proposed Standard) RFC 1323, Internet Engineering Task Force, May 1992. (Obsoletes RFC1185).

[16] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[17] D. Sisalem and H. Schulzrinne, "Switch mechanisms for the ABR service: A comparison study," in *Telecommunication Distribution Parallelism (TDP'96)*, (La Londes Les Maures, France), Institut National des Telecommunications, Evry, June 1996.

[18] R. Jain, "Fairness: How to measure quantitatively?," Tech. Rep. 94-0881, ATM Forum, Sept. 1994.

[19] D. Sisalem and H. Schulzrinne, "End-to-end rate control in ABR," in *First Workshop on ATM Traffic Management (WATM'95)*, (Paris, France), pp. 281–287, IFIP WG 6.2, Dec. 1995.