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

Dorgham Sisalem
Henning Schulzrinne
GMD-Fokus, Berlin
{dor, hgs}@fokus.gmd.de

October 31, 1995

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

Abstract

We propose to enhance TCP's congestion control mechanisms using binary congestion notification (BCN). With this scheme congestion is not simply indicated by packet losses as is currently the case. Instead, switches inform the sources about their congestion state by setting a congestion bit in the data packets.

Unlike other studies, our approach maintains the basic elements of TCP's congestion control mechanisms and at the same time tries to adapt TCP's response to BCN based on the source behavior proposed for sources using ATM's available bit service (ABR).

Through different simulation models we compare the throughput achieved with TCP using BCN (BCN-TCP) and 4.3BSD Tahoe-TCP and Reno-TCP. Also, the effects of integrating TCP with BCN-TCP are investigated.

The effects of running TCP over ABR are studied with the help of a simple simulation model. Here, the benefits of using BCN-TCP in avoiding multiple packet losses are demonstrated and explained.

Finally, fairness issues when mixing long and short distance traffic are considered. Our simulations suggest that BCN-TCP has a worse fairness performance towards long distance traffic than Tahoe- and Reno-TCP. We investigate the reason for this and suggest different approaches for overcoming this problem.

1 Introduction

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

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 using fast retransmission and fast recovery. Older versions, e.g., 4.3BSD Tahoe, use only the fast retransmission scheme where the transmission window is reduced to one packet. The retransmission and severe window reduction actions taken in TCP after detecting a loss lead to bandwidth wastage and underutilization. In this paper, an enhancement to TCP's congestion control mechanisms is introduced that tries to overcome this drawback by using binary congestion notification (BCN).

In this 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 acknowledgment packets sent back to the source. The source then reduces the transmission window by a multiplicative decrease factor (MDF). In this, the scheme resembles the congestion control schemes using bit marking proposed for ATM data services [1]. 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 buffers and the delay caused by retransmissions and the longer queues can be decreased this way.

In the next section, some related work to this subject is introduced. In section 3 our approach is described and the differences to other proposals are highlighted. The simulation environment and simulation results over LAN as well as WAN distances are then presented in sections 4 and 5. The effects of integrating TCP and ABR are described in section 6. Here, the benefits of introducing binary congestion notification in avoiding multiple packet losses are shown and the performance improvements, especially for Reno-TCP, are demonstrated. Finally, section 7 handles the fairness issue and describes bandwidth distribution results achieved with Tahoe-, Reno- and BCN-TCP for the case of a mixed long and short distance traffic. The unfairness of BCN-TCP towards long distance 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 proposals for using binary congestion notification methods in TCP and ATM. Our scheme is mainly based on these proposals and will be described in the next section.

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 size of the maximum packet size possible for this connection. When a new connection is established or after the expiration of a timer, this window is set to the size of one packet. Each time an acknowledgment 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 acknowledgment 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 slow start algorithm was supplemented through the congestion avoidance algorithm. With this algorithm the congestion window is increased for each acknowledgment as follows:

$$\text{cwnd} \mathrel{+}= \frac{1}{\text{cwnd}}$$

  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.

The loss of a packet can be detected either through the expiration of a timer or with the fast retransmission scheme. With fast retransmission, a packet X_n is considered to be lost if a source receives a number of duplicate acknowledgments, usually three, for packet X_{n-1}. Upon that, the lost packet is retransmitted.

In the Tahoe version, TCP reacts to a packet loss 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 acknowledgment for the retransmitted packet the source enters the slow start phase and the transmission window can be increased exponentially while (cwnd <= ssthresh). In the Reno version, the fast recovery scheme is used. With this scheme the source retransmits the lost packet after receiving three duplicate acknowledgment 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 acknowledgment 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 acknowledgment arrives that acknowledges new data cwnd is set to ssthresh and the congestion avoidance mode is entered. For a more detailed description of the different congestion control mechanisms of TCP, see [2].

In TCP/IP based networks congestion is indicated by dropping packets at congested routers. Packets are usually dropped when the queue of the router reaches its limit (drop tail scheme). However, to efficiently use the binary congestion notification scheme the intermediate routers must be capable of detecting incipient congestion and mark packets before their buffers are filled. In this study we use another approach for dropping packets that is more appropriate for the binary congestion notification scheme. With the random early drop (RED) scheme described in [3] the packets are dropped or marked when the average queue length exceeds a certain threshold; a more detailed description will be provided in sec. 4. Compared to the drop tail scheme higher utilization can be reached with this scheme, fairer bandwidth distribution is achieved and global synchronization can be avoided [3]. Note that for the binary congestion notification scheme, any other router algorithm could be used that can detect incipient congestion.

Probably the most important work done in the area of binary congestion notification in TCP was done by Sally Floyd [4]. In her paper [4], she proposes guidelines for TCP's response to binary congestion notification (BCN) that can be summarized 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 a BCN does not trigger the sending or retransmitting of any data packets.

For the simulations done using this approach TCP was modified as follows [4]: 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 [5] the importance of using binary congestion notification is stressed as well. The author proposes a response scheme similar to some extent to the older bit marking schemes that were proposed for ATM [6]. 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 mechanisms.

With the introduction of ATM technology the issue of integrating TCP and ATM in a heterogeneous environment will play an important role. Many current applications are based on the currently widespread TCP/IP infrastructure. Different studies [7, 8, 9] 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 connection 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 [10].

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

In its September 1994 meeting the traffic management group (TMG) of the ATM Forum* voted for a rate control based mechanism as the standard mechanism for congestion control in ATM. The mechanism works as follows: After every $N^{\dagger}$ 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 destinations end systems return 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.

*The ATM Forum is an international consortium whose goal is to accelerate the use of ATM products and services through the development of interoperability specifications and the promotion of industry cooperations.

$^{\dagger}N$ is usually set to 32 [11]

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 signaled 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.

This brief introduction of the rate control schemes proposed for ATM suggests additionally the importance of introducing the binary notification capability in TCP. Otherwise, some degradation in TCP's performance running over rate controlled ATM connections must be expected [12, 13].

3 Implementing Binary Congestion Notification in TCP

Our aim in this study is to investigate the effects of using binary congestion notification in TCP. To actually be able to use this scheme, the IP packets have to include a congestion bit and a congestion flag with which a sender can inform the intermediate switches of its willingness to use binary congestion notification. There are a few possibilities of adding the congestion bits to the IP packets:

- The simplest solution would be to use one of the IP options.
- The flow label field proposed for the IPv6 header can be used for requesting the usage of binary congestion notification [14].
- Instead of marking packets the congested switch could just send an ICMP Source Quench message.

Adding the congestion information to the IP level allows a wider usage of this information. So, it would be possible to add simple congestion control mechanisms to currently unreliable services such as UDP.

For the time being, we will just assume that the IP packets have a congestion bit as well as a congestion flag.

With our approach for using binary congestion notification in TCP we try to integrate the guidelines for using congestion notification as was proposed in [4] and the ATM source response to bit marking control schemes [1].

To provide for the appropriate response to congestion notification we had to modify our TCP simulator in the following way:

3.1 General Behavior

The general scenario for using binary congestion notification in a TCP/IP environment looks as follows:

- Data packets are sent with a congestion bit set to 0 and the congestion flag set to 1.
- When encountering a congested link, the congestion bit is set to 1.
- At the receiver the value of the congestion bit in the received packets is added to the acknowledgment packets.
- According to the received congestion information in the acknowledgment packets, the source must then change its transmission window appropriately.

3.2 Source Behavior

Depending on the received congestion information, the source works in one of the following two modes:

- **Binary Notification Mode:** The source goes into this mode upon receiving an acknowledgment packet with the congestion bit active, and if there were no packets waiting to be acknowledged after a retransmission or a timeout. This mode lasts then until the packet sent just before receiving the acknowledgment with the set congestion bit is acknowledged as well. That is, the binary notification mode lasts for a complete round trip time.

  After entering this mode, the slow start threshold (ssthresh) is set to half the current transmission window and the transmission window is reduced multiplicatively by the multiplicative decrease factor (MDF). During this mode, the following cases can be distinguished:

  1. An acknowledgment packet is received with the congestion bit set. This causes the transmission window to be further reduced by the multiplicative decrease factor (MDF).
  2. Three duplicate acks are received. This case is handled exactly the same way as the previous case. That is, the transmission window is decreased by MDF. In addition to that, the lost packet is retransmitted. Receiving duplicate acks during this mode can be interpreted as follows: Either the switch that was marking the packets as congested is now severely congested and starts dropping packets or the connection passes through a congested older switch with no marking capability. As the window size has already been decreased in response to the marked congestion bits in the previously received acknowledgments an additional decrease by MDF should be enough to solve the congestion situation.
  3. An acknowledgment packet is received with the congestion bit unset. The window is simply advanced by the acknowledged amount, and the window is increased by 1/cwnd.
  4. The acknowledgment for the packet sent just before entering the binary notification mode is received. That is, a complete round trip time since the start of the binary notification mode is over and the source can leave the binary notification mode and increase its transmission window again. Depending on the value of the transmission window reached at the end of the binary notification mode the source can increase its transmission window as follows:

- If the transmission window is below the ssthresh value set at the beginning of the binary notification mode the source can exponentially increase its rate using the slow start scheme until the ssthresh value is reached. Afterwards, the window can only be increased additively with the congestion avoidance scheme.
- Is the transmission window larger than the value of ssthresh then the congestion avoidance scheme must be used to increase the window size.

5. In the case of a timeout event the binary notification mode is left and the source goes into the slow start mode.

- **Normal TCP Mode**: If three duplicate acknowledgments are received or a retransmission timer expired and the source was not in the binary notification mode then the usual retransmission and window reduction actions are taken. That is, for the Tahoe version the window is set to 1 and for Reno the transmission window is set to half of its current value. See Section 2 for a more detailed description of the taken responses. The large window reduction done in this case is advisable as the packet loss can be interpreted as follows: The connection passes either through a severely congested bit marking switch or an older congested switch that uses packet drops as a congestion indication. In both cases, a large window reduction is needed to solve the congestion situation.

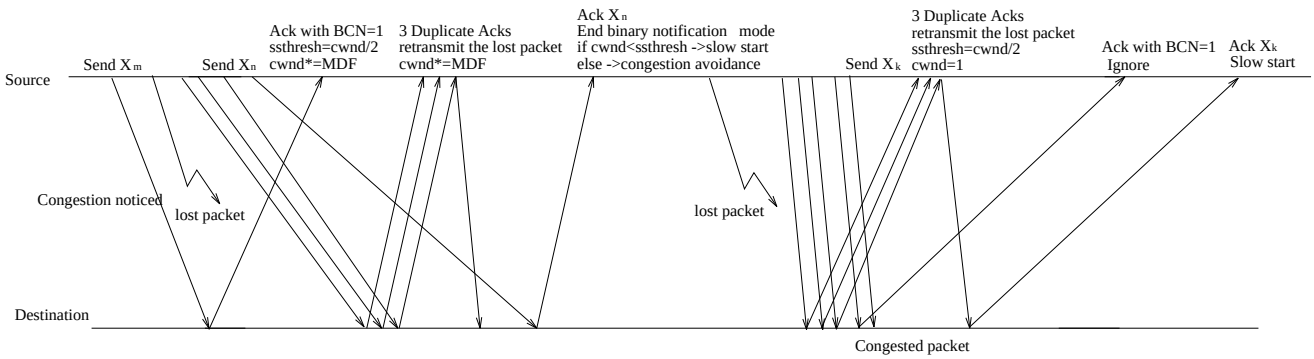

Figure 1: An example of the packet exchange with BCN-TCP

As an example for the source behavior consider the following scenario, see also Fig. 1: The source has just sent packet X_n before receiving the acknowledgment for packet X_m with the congestion bit set. Upon this, the source enters the binary notification mode until the acknowledgment for packet X_n is received. If the value of the transmission window at this time was cwnd $= W$ then ssthresh is set to $W/2$ and the transmission window is decreased by MDF (cwnd *=MDF). During this mode the source decreases its transmission window by MDF each time an acknowledgment with an active congestion bit is received or after receiving three duplicate acknowledgments. Upon receiving the acknowledgment for packet X_n, the source can leave the binary notification mode and start increasing its transmission window. If cwnd was now lower than ssthresh, then the slow start procedure is used until cwnd >= ssthresh, afterwards cwnd is increased using the congestion avoidance scheme, i.e., in steps of 1/cwnd.

If three duplicate acks were received for packet X_l and the source was not in the binary notification mode then the normal TCP procedures are taken, i.e., slow start, congestion avoidance and

fast recovery (if implemented). If packet X_k was the last sent packet before receiving the duplicate acknowledgments then all binary congestion notifications that might arrive at the source before acknowledging X_k can be ignored.

At this point we have to restate the main goals we aimed at achieving with the here presented implementation:

- First, enhancing TCP with binary congestion notification capabilities should not affect the source behavior and performance in environments that support packet discard as the only way to indicate congestion. That is, the usual congestion control mechanisms of fast retransmission, fast recovery, slow start and congestion avoidance must be applied directly if the congestion bit is not used.
- Secondly, to ease the future integration of TCP and ATM, TCP implementations that can handle binary congestion notification should try to adopt the source behavior specified for ATM sources using the available bit rate service [1]. An ABR source that receives a binary congestion notification from the network reduces its rate by a multiplicative reduction factor. If a TCP connection was using this ABR link and reduced its rate after a congestion indication by more than this multiplicative factor then it would only be wasting bandwidth and underutilizing the bandwidth it could have actually been using.

4 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 behavior of TCP over a LAN as well as a WAN distance we set the value of τ to 10 msec and 100 msec respectively. The links rates were set to 10 Mbps.

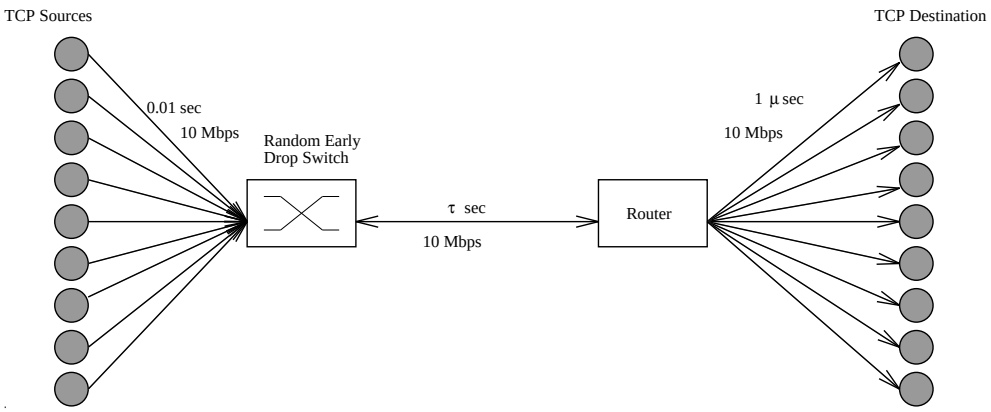

Figure 2: Simulation test topology

For simulating the behavior of TCP we used the PTOLEMY [15] simulation tool to write a TCP model [12]. We implemented two versions of the 4.3BSD TCP: Tahoe-TCP with fast retransmission, delayed acknowledgments, round trip calculation and the Reno-TCP version with the fast recovery scheme as well. Both versions were enhanced through large windows, i.e., the maximum window

size can be larger than the 64 kbytes allowed with the current implementations of TCP [16]. As the 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 very 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 maximum packet size was set here to 500 bytes and a multiplicative decrease factor of 0.975 was used.

The switch is a random early drop (RED) gateway as was proposed by Floyd and Jacobson [3]. A RED gateway detects incipient congestion by computing the average queue size[‡]. When the average queue size exceeds a preset minimum threshold (Min_{th}) the switch marks each incoming packet with some probability. Exceeding a second maximum threshold (Max_{th}) leads to the marking of all arriving packets. For the case of TCP without the binary congestion notification the packets are not marked but dropped. This approach not only keeps the average queue length low but ensures fairness and avoids synchronization effects.

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

5 Simulation Results

We tested and verified our scheme in two steps. In the first step the performance of TCP using binary congestion notification is compared to the performance achieved without the BCN enhancement. We ran simulations using round trip delays of 40 msecs and 220 msecs and with different buffer values.

In the second set of experiments the effects of integrating BCN-TCP traffic with normal TCP connections are tested.

5.1 A Comparison of TCP and BCN-TCP

The results depicted in Figures 3 and 4 show the link utilization achieved with Tahoe-TCP, Reno-TCP and BCN-TCP over a time period of one minute. As expected, using Reno-TCP results in general in higher throughput than that achieved with Tahoe-TCP. The results achieved with BCN-TCP resemble to a great extent those achieved for Reno-TCP. The throughput improvements are especially evident for the WAN cases. Due to the long round trip delay the recovery from packet losses takes more time increasing thereby the idle time of the sources and leading to the oscillating behavior that can be seen in Fig. 4.

[‡]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.

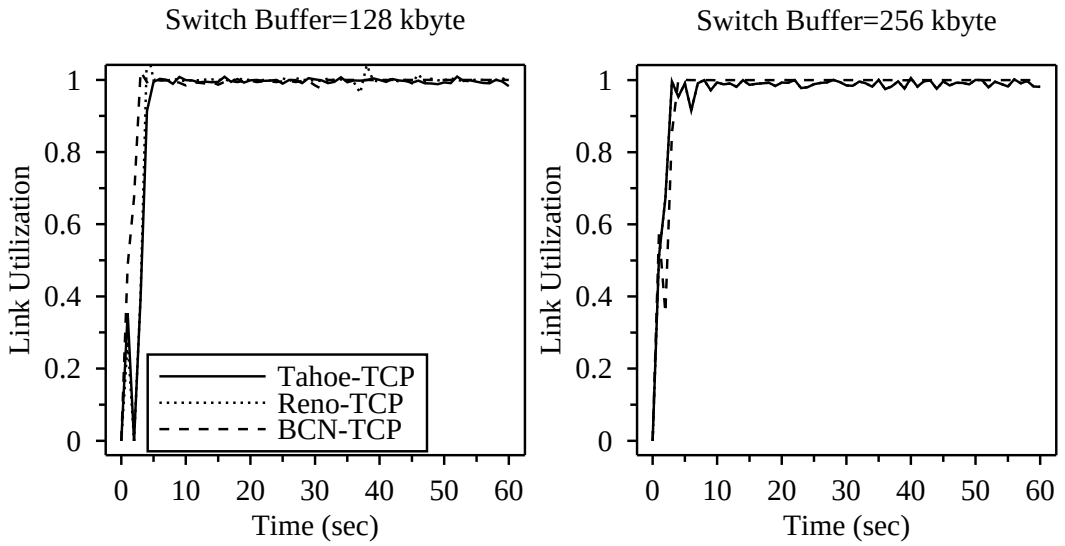

Figure 3: Link utilization for Tahoe-, Reno- and BCN-TCP with a propagation delay of 40 msec

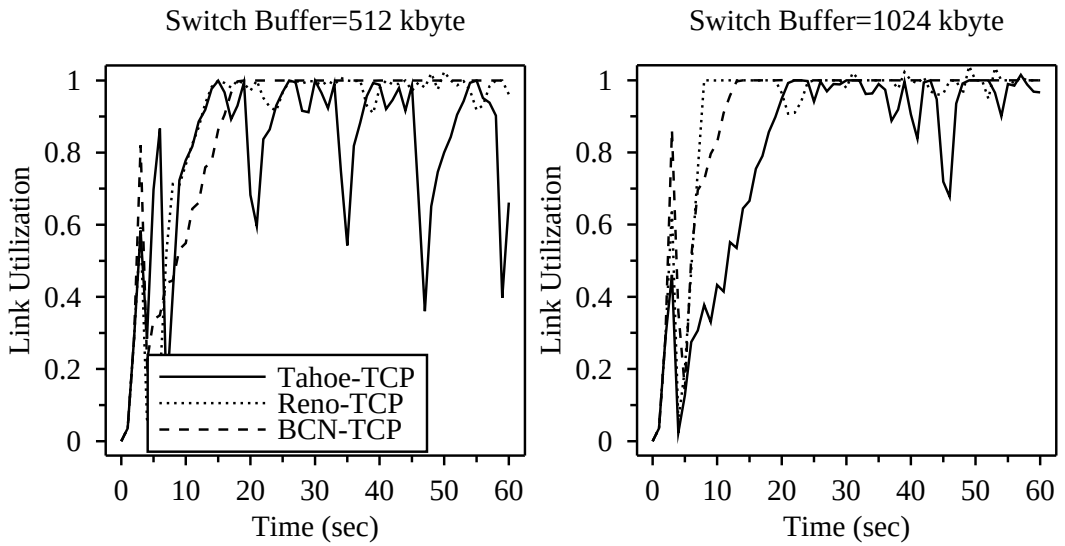

Figure 4: Link utilization for Tahoe-, Reno- and BCN-TCP with a propagation delay of 220 msec

Propagation Delay	Buffer Length	Tahoe-TCP	Reno-TCP	BCN-TCP
40 msec	128 kbytes	95%	96%	98%
	256 kbytes	97%	97%	98%
220 msec	512 kbytes	80%	87%	88%
	1024 kbytes	81%	92%	92%

Table 1: Link utilization reached using Tahoe-, Reno- and BCN-TCP with different switch buffer sizes and different round trip delays

5.2 Introducing BCN-TCP in a Heterogeneous Environment

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

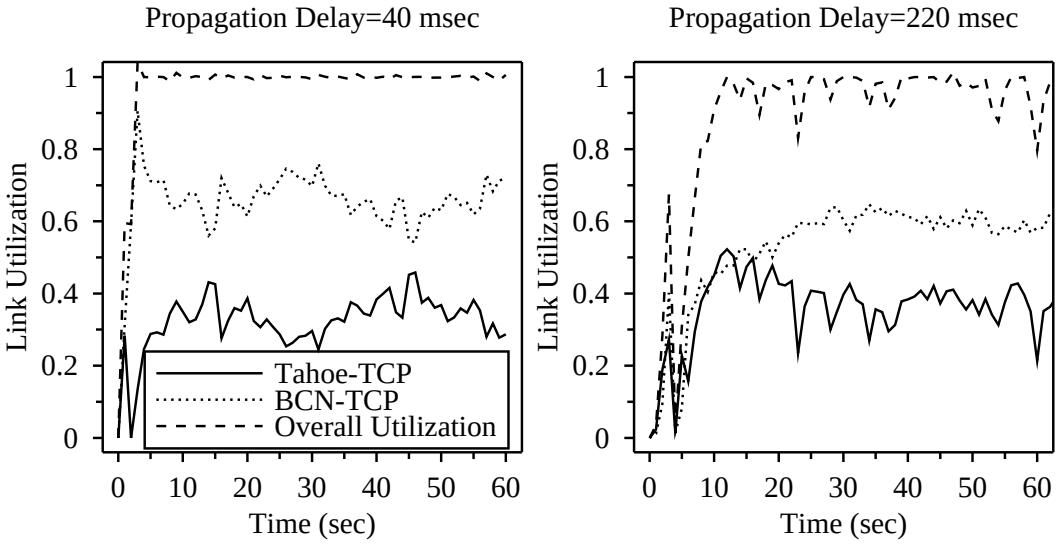

Figure 5: Utilization achieved with only a third of the connections using BCN-TCP

In Fig. 5 only a third of the connections is using binary congestion notification. While the link utilization remains as high as 90%, the bandwidth distribution is completely unfair. Even though that only a third of the connections uses BCN-TCP, these connections receive about two thirds of the available bandwidth. 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 Tahoe-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 slightly higher than that of dropping one. This would also be the appropriate approach to use with drop tail switches. In those switches arriving packets are dropped whenever the queue length exceeds a certain preset threshold. When using the binary

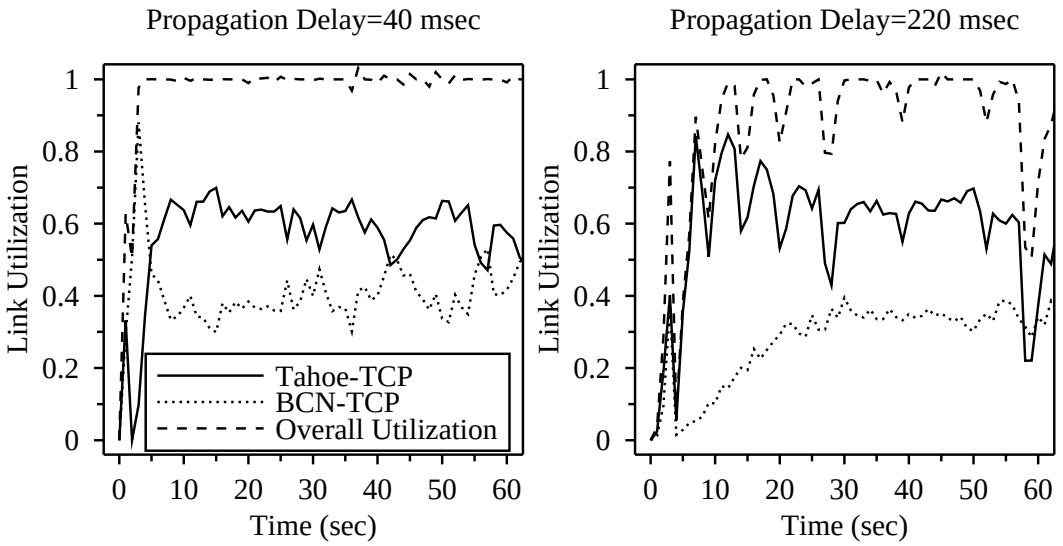

Figure 6: Achieved utilization for Tahoe-TCP combined with BCN-TCP traffic with the marking probability higher than the drop probability

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 $R = \text{Random}[0,1]$ is from the uniform distribution on [0,1] and p_b is calculated using the average queue length and the threshold values. 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 Tahoe-TCP connections

Fig. 6 depicts the results achieved using $\alpha = 0.85$. The bandwidth is now fairly distributed. The validity of the value of α is, however, limited to this configuration. Therefore, further work must be done on investigating the way switches should deal with mixed traffic.

6 Integration of TCP and ABR

In this section we briefly investigate the effects of integrating TCP with the available bit rate service provided by ATM.

We chose a very simple testing configuration, see Fig. 7, consisting of three TCP/IP sources that are connected to an ATM switch through separate user-network interfaces (UNI). The ATM link has a bandwidth of 45 Mb and the links from the TCP source to the switch have a bandwidth

of 40 Mb. The connections have an end-to-end propagation delay of 0.010004 sec and a TCP packet size of 0.5 kbytes. The simulations were run for an UNI buffer size of 25 kbytes and 5 kbytes with a multiplicative decrease factor of 0.975.

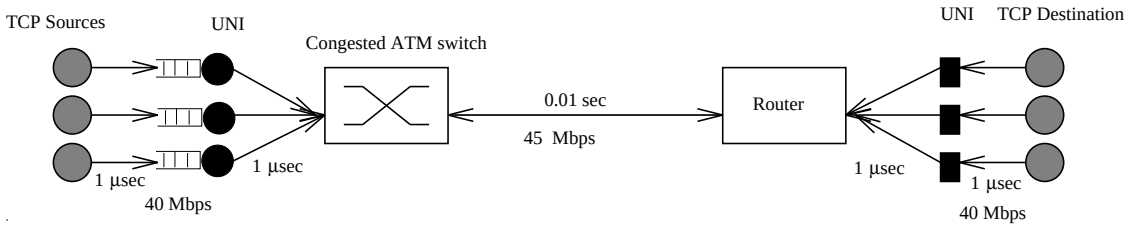

Figure 7: Simulation test topology for the integration of TCP and ABR

The switch determines which packets to mark using the enhanced proportional rate control algorithm (EPRCA) [11]. EPRCA computes a mean allowed cell rate (MACR) using exponential weighted averaging.

$$\text{MACR} = (1 - \alpha)\text{MACR} + \alpha\text{CCR}$$

Here, α is the exponential averaging factor and is usually set to 1/6 [17]. CCR is the current cell rate of the connection.

Connections that advertise a current rate that is higher than the calculated MACR are asked to reduce their rates by setting the congestion indicator (CI) bit in their RM cells.

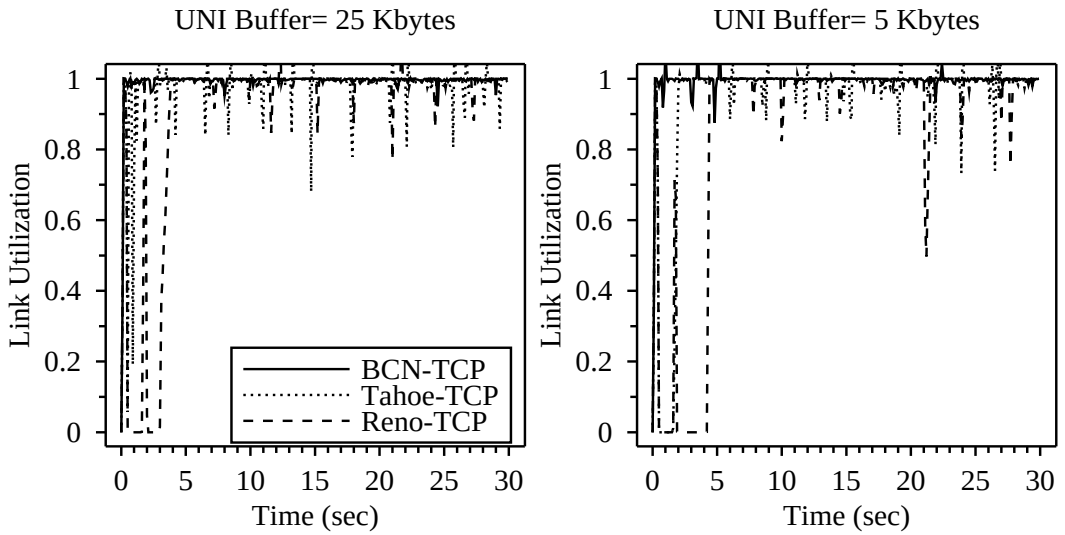

Figure 8: Achieved link utilization with Tahoe-, Reno- and BCN-TCP over ABR

Fig. 8 and Tab. 2, depicting the utilization results achieved with Tahoe-TCP, Reno-TCP and BCN-TCP reveal two important observations. First, while BCN-TCP's throughput remains the same for the different used buffer values, the throughput achieved with the Tahoe and Reno versions depends to a great extent on the available buffer at the UNI. This can be explained as

Buffer Length	Tahoe-TCP	Reno-TCP	BCN-TCP
25 kbytes	98%	90%	99%
5 kbytes	94.6%	86%	99%

Table 2: Utilization achieved with Tahoe-, Reno- and BCN-TCP over ABR

follows: While BCN-TCP connections reduce their transmission windows as soon as congestion is signaled, normal TCP sources keep on increasing their transmission windows. This is done until the UNI buffer reaches its capacity and packets get discarded. That is, the larger the UNI buffer the higher the transmission window values that can be reached before detecting the packet loss. This yields then a higher slow start threshold (ssthresh) and a smaller recovery period after the retransmission. Thus, higher throughput can be achieved with the provision of larger UNI buffers.

Secondly, on the contrary to the results achieved in Section 5, Tahoe-TCP achieves higher throughput than Reno-TCP. This was caused by the packet drop behavior noticed at the user network interfaces.

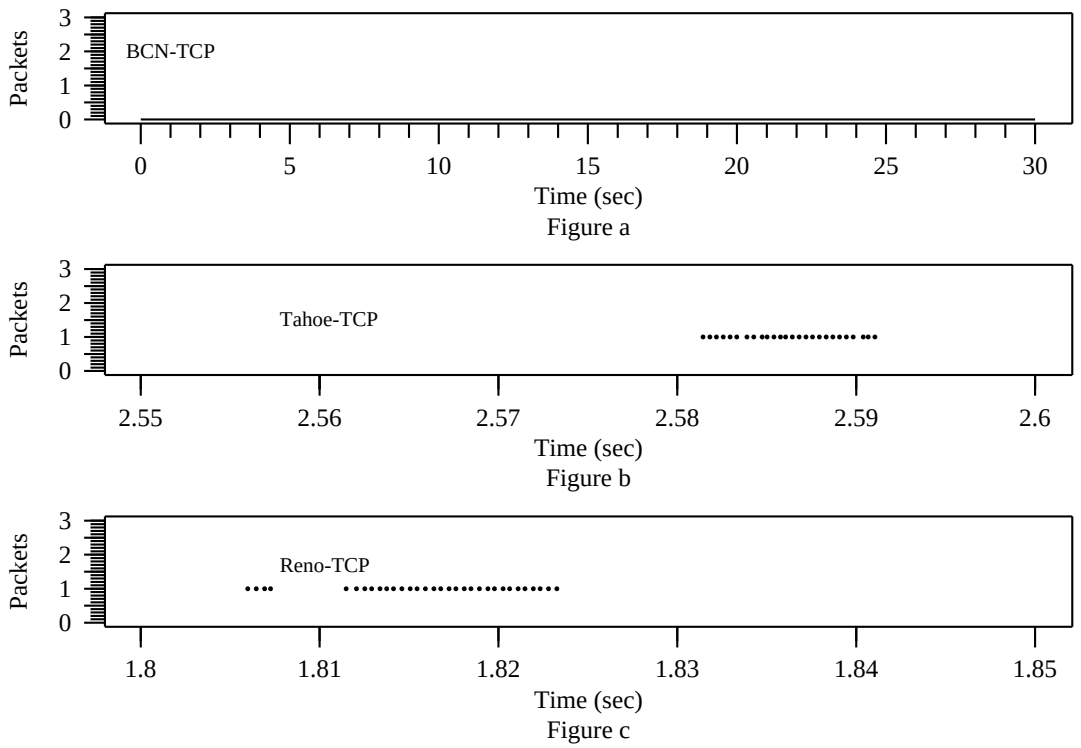

Figure 9: Number of packets dropped during a congestion period at the UNI for Tahoe-, Reno- and BCN-TCP with 25 kbytes UNI-buffer

Zhang et. al [18] show that with a simple configuration like the one used here, the number of discarded packets at a congested link during one epoch equals the amount by which the transmission window was increased during the same epoch. We define epoch to be the time required

to acknowledge a complete window. That is, during the slow start phase the loss would equal an entire window. During the congestion avoidance period in which the window is only increased by approximately one packet size per round trip the loss would be only one packet during each congestion phase.

However, during our simulations multiple packet losses in one epoch were noticed even during the congestion avoidance periods. From Fig. 9.b and 9.c we can see that during a period lasting for around 0.01 sec which is smaller than the round trip delay of the tested connections more than 20 packets get dropped for both Reno-TCP and Tahoe-TCP. This can be explained as follows: During the uncongested periods both the TCP connections and their UNIs have the same rate, the peak link rate. During the congested states the UNI decreases its allowed cell rate with every received RM cell with CI=1. A TCP connection, however, can keep on sending with a higher rate and additionally increases its window. The packets that can not be directly serviced at the UNI are then buffered. When the buffer reaches its capacity arriving packets will be discarded. Due to the UNI rate reduction the difference between the size of the increased window after an epoch is larger than the number of the packets that can be serviced during this epoch with the new reduced rate. This leads then to the multiple packet discard. As an example for this, consider the following case: At some point the UNI has a rate of 10 packets/second and the TCP source has a window of 10 packets. The connection has a propagation delay of 1 second. At this moment the UNI is notified to reduce its rate to 8 packets/second. As the last 10 TCP packets were sent at a rate of 10 packets/second their acknowledgments will arrive at the same rate. Thereupon, the TCP source will send 11 packets during one second. As only 8 packets can be serviced during the same period, 3 packets will have to be buffered or discarded. This scenario will repeat itself until the buffer capacity is reached and the arriving packets are discarded.

After acknowledging a retransmitted packet Tahoe-TCP sources start sending data packets from the last acknowledged packet. Thereby, retransmitting all lost packets without taking any further actions and the effects of the multiple losses are therefore minimized. For the Reno-TCP sources using the fast recovery scheme this is entirely different. Reno-TCP is optimized for the case of a single loss per epoch. For the case of a multiple drop, one of the following scenarios are applicable [19, 20]:

- Fast recovery as well as fast retransmission are triggered for every lost packet. This can cause the window and the value of the slow start threshold (ssthresh) to be reduced in half a number of times. Due to the small ssthresh the window will then grow slowly with the congestion avoidance scheme.

- The loss could not be detected with the fast retransmission scheme. That is, less than three packets arrived at the destination after the second dropped packet, so the fast retransmission scheme could not be triggered. In this case the recovery will even take longer as the source will have to wait for a retransmission timeout before proceeding with the transmission.

Due to the multiple losses Reno-TCP achieves worse utilization than both BCN-TCP and Tahoe-TCP. Introducing binary congestion notification in this case would likely result in substantial performance improvements. Fig. 9.a shows that, here, using BCN-TCP results in a packet loss ratio of zero. Thereby, the combination of Reno-TCP and binary congestion notification would sustain the advantages of fast recovery for packet drop networks and benefit from the bit marking schemes proposed for ATM networks.

7 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, whereas each source is the origin of three similar connections. The distance between two neighboring 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.

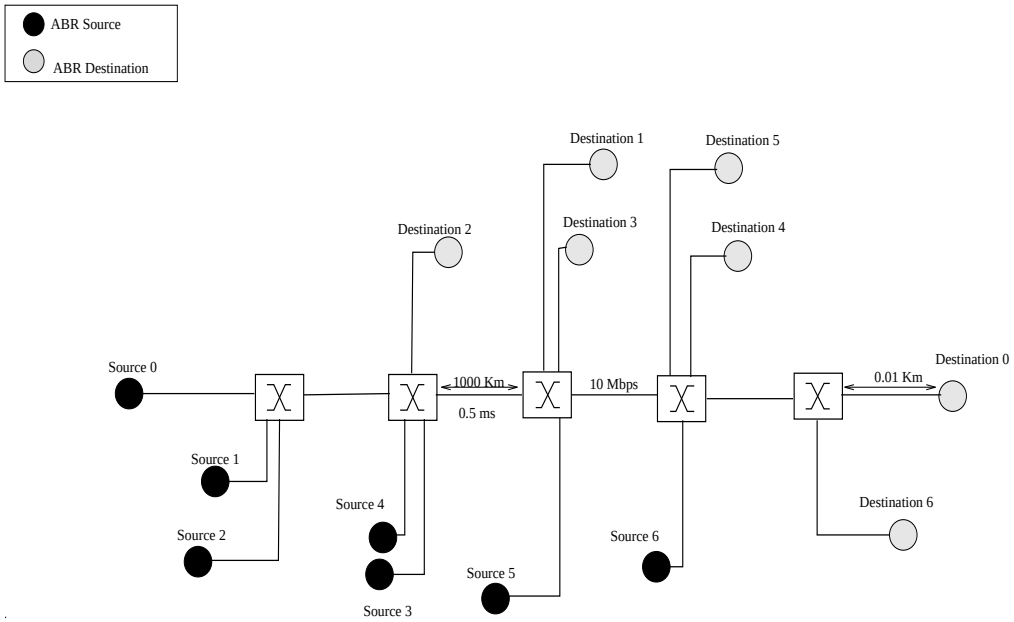

Figure 10: Test topology

The switches are RED switches with a buffer size of 500 kbytes and a minimum threshold (Max_{th}) of 50 kbytes and a maximum one (Max_{th}) of 200 kbytes.

Again here, persistent sources are used with a maximum packet size of 500 bytes and a multiplicative decrease factor of 0.975. As a fairness criterion we use the max-min fairness definition as it was described in the ATM Forum traffic management specification 4.0 [1]. A connection's fair bandwidth share is at the minimum

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

and maximally

$$\frac{\text{Bandwidth of link} - \sum \text{Bandwidth of connections bottlenecked elsewhere}}{\text{Number of connections} - \sum \text{Number of connections bottlenecked elsewhere}}.$$

Where 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 [21] suggests using the so called fairness index.

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

Version	connection							fairness index
	0	1	2	3	4	5	6	
BCN-TCP	0.2	0.63	2.5	1.8	0.64	2.5	3.1	0.65
Tahoe-TCP	0.32	0.68	2.3	1.6	0.69	2.3	3	0.87
Reno-TCP	0.36	0.7	2.3	1.6	0.68	2.2	2.9	0.86
Fair share	0.83	0.83	1.66	0.83	0.83	1.66	2.5	1

Table 3: Average achieved throughput in Mbps for each connection compared to the calculated fair share with the max-min criterion

Table 3 contains the fairness indexes for the three schemes examined and the fair bandwidth share for each connection according to the max-min fairness criterion. 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 both connections will be discarded, marked for BCN-TCP, 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 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 larger reduction factors leads to a more oscillating traffic and possibly lower throughput, see [12].
- In our second approach each connection uses a multiplicative decrease factor in accordance with the number of switches it passes through. 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 acknowledgment 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 initialized 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 acknowledgment packets the source can calculate the number of passed switches 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.

8 Summary and Open Issues

While the simulation results presented here suggest the usefulness of binary congestion notification especially when combined with Reno-TCP there are still a number of open issues that must be addressed in future work.

- The efficiency of binary congestion notification schemes in general 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 utilization 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 [22, 11, 17]. 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 have shown that using the explicit rate indication mode in ATM results in much better utilization 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.

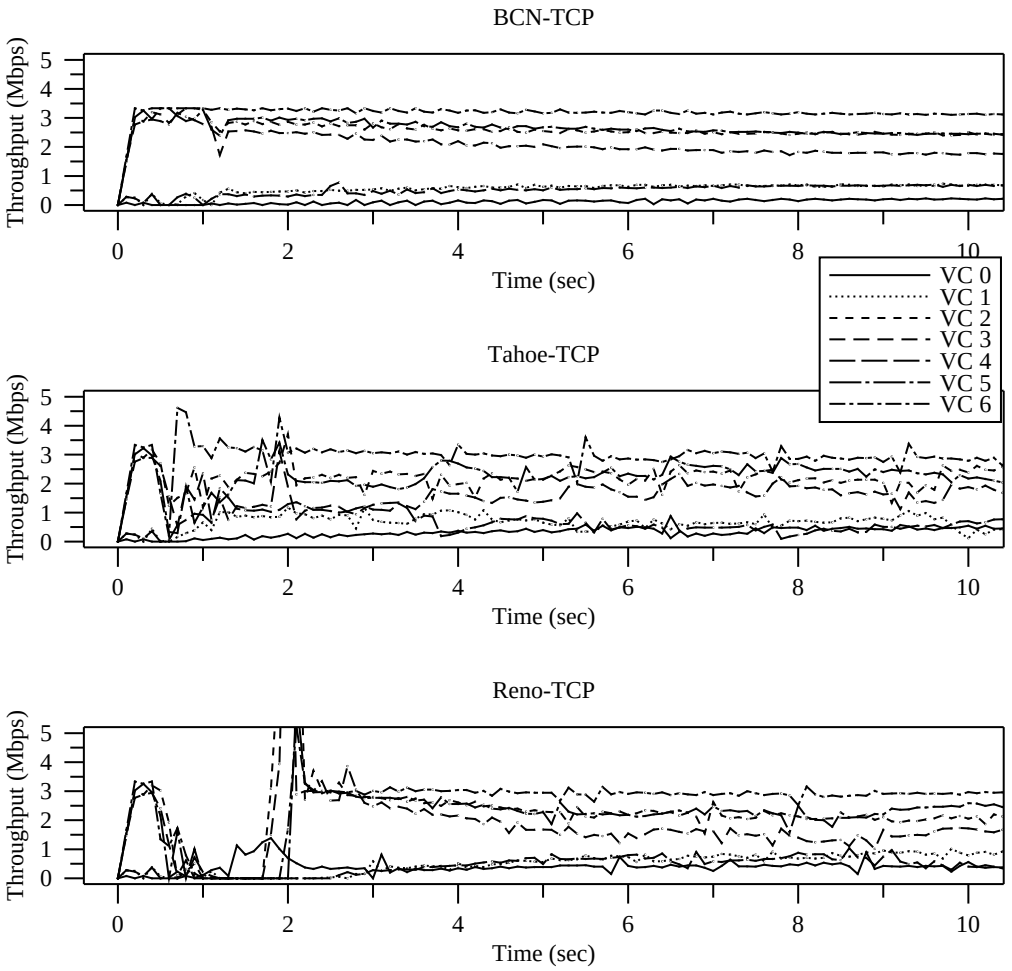

Figure 11: Throughput achieved with Tahoe, Reno and BCN-TCP

References

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

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

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

[4] 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.

[5] P. Calhoun, "Congestion control in IPv6 Internetworks," internet draft, May 1995.

[6] 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," ATM Forum 94-0438, May 1994.

[7] 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.," ATM Forum 94-0119, Jan. 1994.

[8] 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.

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

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

[11] H. Hsiaw, K. Fendrick, J. Alpan, L. Roberts, 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," ATM Forum 94-0438R2, Sept. 1994.

[12] D. Sisalem, "Rate based congestion control and its effects on TCP over ATM," Master's thesis, School of Engineering, TU Berlin, Berlin, July 1995.

[13] 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.

[14] D. E. Comer, *Internetworking with TCP/IP*, vol. 1. Englewood Cliffs, New Jersey: Prentice Hall, 3rd ed., 1995.

[15] "The Almagest." A PostScript version of this document is available from ftp://ptolemy.Berkeley.EDU, 1994.

[16] 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).

[17] 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.

[18] S. Shenker, L. Zhang, and D. D. Clark, "Some observations on the dynamics of a congestion control algorithm," *ACM Computer Communication Review*, pp. 30–39, Oct. 1990.

[19] C. Villamizar and C. Song, "High performance TCP in ANSNET," technical note, Advanced Network & Services, Inc. (ANSNET), Sept. 1994.

[20] S. Floyd, "TCP and successive fast retransmits." A PostScript version is available from ftp://ftp.ee.lbl.gov/papers, May 1995.

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

[22] T. Chen, S. Liu, V. Samalam, J. Ormord, and N. Yin, "Examples of switch mechanisms," ATM Forum 95-0345, Apr. 1995.