Binary Congestion Notification in TCP*

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

June 25, 1996

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 rate 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. 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.

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 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 section 4. The effects of integrating TCP and ABR are described in section 5. 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.

2 Related Work and Previous Proposals

Probably the most important work done in the area of binary congestion notification in TCP was done by Sally Floyd [1]. In her paper [1], 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 BCN does not trigger the sending or retransmitting of any data packets.

In another work by Calhoun [2] the importance of using binary congestion notification is stressed as well. The author proposes a response scheme similar to some extent to older bit marking schemes that were proposed for ATM [3]. 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

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

will play an important role. Many current applications are based on the currently widespread TCP/IP infrastructure. Different studies [4] 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.

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 TCP's congestion control 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^* 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 (UNI) of the source 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.

- **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 by setting a congestion indication (CI) bit in the backward RM cells.

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 [6, 7].

3 Implementing Binary Congestion Notification in TCP

Our aim in this study is to show the performance achieved 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 source can inform the intermediate switches of its willingness to use binary congestion notification. The congestion bit can be added, for example, by using special options, the flow label in IPv6 or ICMP Source Quench messages [8].

3.1 Source Behavior

According to the received congestion information the source works in one of the following two modes:

- **Binary Notification Mode:** Upon receiving an acknowledgment packet with the congestion bit active and there were no packets waiting to be acknowledged after a retransmission or a timeout the source goes into this mode. This mode lasts 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.

  In this mode a variable called the slow start threshold (ssthresh) is set to half the current transmission window (cwnd) and the transmission window is reduced multiplicatively by a multiplicative decrease factor (MDF). During this mode, different 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 MDF.
  2. Three duplicate acks are received. In this case the lost packet is retransmitted and the window is further decreased by MDF.
  3. An acknowledgment packet is received with the congestion bit unset. The window is simply advanced by the acknowledged amount.
  4. The acknowledgment for the last sent packet before entering the binary notification mode is received and the source can leave this mode. Depending on the value of the transmission window reached the source can increase its transmission window exponentially with the slow start scheme as long as ($cwnd < ssthresh$). Afterwards, the window is increased with the congestion avoidance scheme.
  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. See [8] for a more detailed description of the taken responses.

Fig. 1 shows an example for the source behavior.

4 A Comparison of TCP and BCN-TCP

We have chosen a test topology consisting of nine connections sharing a single link. All the hosts have the same parame-

$^*N$ is usually set to 32 [5]

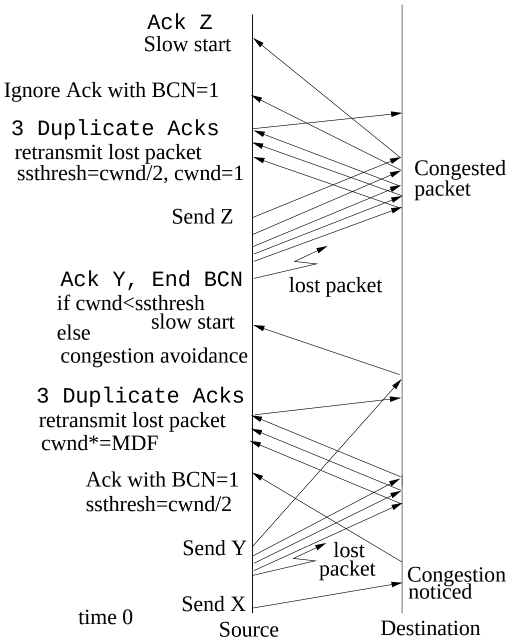

Figure 1: An example of the packet exchange with TCP-BCN

ters and share the same buffer of the congested switch. The shared link has a propagation delay of τ seconds. The link rate was set to 10 Mb/s.

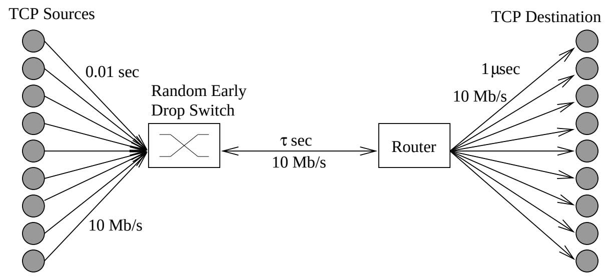

Figure 2: Simulation test topology

For simulating the behavior of TCP we used the PTOLEMY [9] simulation tool to write a TCP model [6]. We implemented two versions of the 4.3BSD TCP: Tahoe-TCP with fast retransmission, delayed acknowledgments and round trip calculation and the Reno-TCP version with the fast recovery scheme as well. Both versions use maximum transmission windows larger than the 64 kbytes allowed with the current implementations of TCP. 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 imprecise timeout values. Therefore, we used here a clock granularity of only 200 msec.

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 for BCN 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 [10]. A RED gateway detects incipient congestion by computing the average queue size. When the average queue size exceeds a preset minimum threshold the switch marks each incoming packet with some probability. Exceeding a second maximum threshold 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 of 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.

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. This is especially evident for the Tahoe version without the fast recovery scheme.

Delay	Buffer	Tahoe	Reno	BCN
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

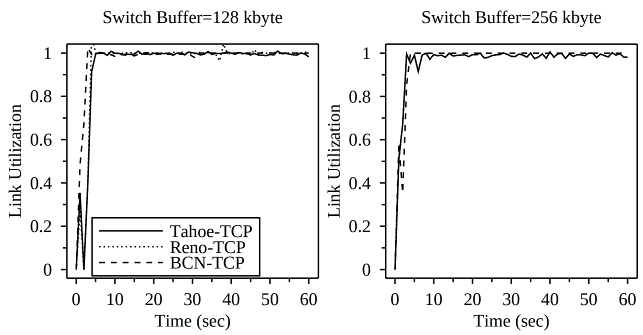

Figure 3: Link utilization for Tahoe-, Reno- and BCN-TCP with a propagation delay of 40 msec

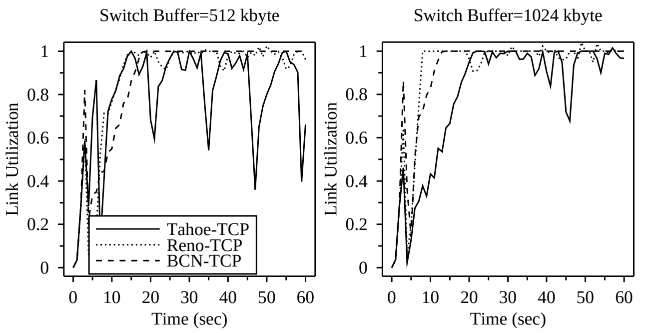

Figure 4: Link utilization for Tahoe-, Reno- and BCN-TCP with a propagation delay of 220 msec

5 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 simple testing configuration, see Fig. 5, 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.

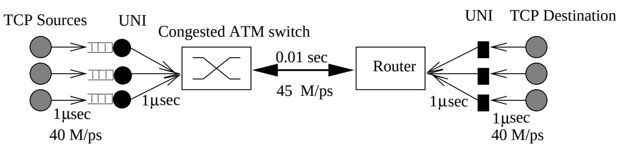

Figure 5: Simulation test topology for the integration of TCP and ABR

The switch determines which packets to mark using the enhanced proportional rate control algorithm (EPRCA) [5]. EPRCA computes a mean allowed cell rate (MACR) using exponential weighted averaging.

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

Here, α is the exponential averaging factor and is usually set to 1/6 [11]. 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.

Buffer	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 for different UNI buffer sizes

Fig. 6 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 link utilization remains the same for the different used buffer values, the utilization achieved with the Tahoe and Reno versions depends to a great extent on the available buffer at the UNI. This can be explained as follows: With the larger UNI buffers the sources can achieve higher transmission windows before detecting the packet loss. This yields a higher slow start threshold (ssthresh) and a smaller recovery period after the retransmission. Thus, higher utilization can be achieved with the provision of larger UNI buffers.

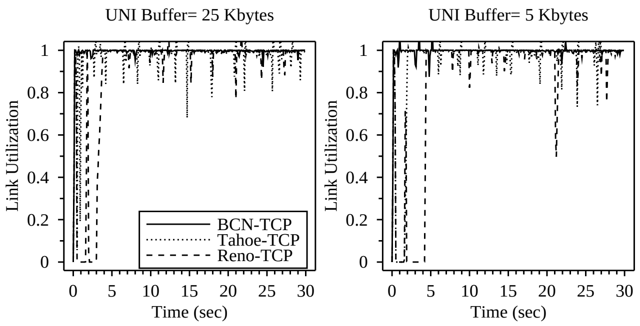

Figure 6: Achieved link utilization with Tahoe-, Reno- and BCN-TCP over ABR for different buffer sizes

Secondly, on the contrary to the results achieved in Section 3, Tahoe-TCP achieves higher throughput than Reno-TCP. This was caused by the packet drop behavior noticed at the user network interfaces.

Zhang et. al [12] 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 an 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. For example we noticed that more than 20 packets got dropped for both Reno-TCP and Tahoe-TCP during a period lasting for around 0.01 sec which is smaller than the round trip delay. 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 and the number of the packets that can be serviced during this epoch with the new reduced rate is larger than one. 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. Reno-TCP on the other hand is optimized for the case of a single loss per epoch. For the case of a multiple drop fast retransmission and fast recovery might be triggered for every lost packet resulting in a long recovery period. On the other side, if the loss was not detected with the fast retransmission scheme the source has to wait for a timeout until it can proceed with the transmission [13, 14].

Due to the multiple losses Reno-TCP achieves worse utilization than both BCN-TCP and Tahoe-TCP. 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.

6 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. While there have been several proposals for switch algorithms for the ABR services, see [11], 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 BCN in heterogeneous environments.
- The possibility of using explicit rate indication instead of BCN is a very interesting enhancement to TCP, especially as different studies have shown that using the explicit rate indication mode in ATM results in much better utilization and fairness results compared to BCN schemes.
- The fairness of the scheme has to be investigated more thoroughly. Various studies [15] have shown that using BCN often results in a bad fairness behavior.

References

[1] 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.

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

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

[4] 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.

[5] A. Roberts, "Enhanced PRCA proportional rate control algorithm," Tech. Rep. 94-0735R1, ATM Forum, Aug. 1994.

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

[7] 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.

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

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

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

[11] R. Jain, "Congestion control and traffic management in ATM networks: Recent advances and a survey," *Computer Networks and ISDN Systems*, Feb. 1995.

[12] 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.

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

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

[15] D. Sisalem and H. Schulzrinne, "End-to-end rate control in ABR," in *IFIP WG 6.2 Broadband Communication*, (Paris), IFIP, Dec. 1995.