****

ATM Forum Document Number: ATM_Forum/96-1761

****

Title: Further Results on UBR+: Effect of Fast Retransmit and Recovery.

****

Abstract:

In a previous contribution, we studied the performance of TCP over UBR+. We showed that TCP perfor-
mance improves by adding EPD and Fair buffer management techniques over UBR. In this contribution we
study the effect of Fast Retransmit and Recovery on these enhancements. In general, Fast Retransmit and
Recovery improves the performance of TCP over UBR. However in some cases, the performance is degraded.
The incremental gain with Fair Buffer Allocation over simple selective drop using per-VC accounting is small.

****

Source:

Rohit Goyal, Raj Jain, Shiv Kalyanaraman, and Sonia Fahmy.
The Ohio State University (and NASA)
Department of CIS

Raj Jain is now at Washington University in Saint Louis, jain@cse.wustl.edu http://www.cse.wustl.edu/~jain/

Seong-Cheol Kim
Samsung Electronics Co. Ltd.
Chung-Ang Newspaper Bldg.
8-2, Karak-Dong, Songpa-Ku
Seoul, Korea 138-160
Email: kimsc@metro.telecom.samsung.co.kr

****

Date: December 1996.

****

Distribution: ATM Forum Technical Working Group Members (Traffic Management)

****

****

1   Introduction

In our previous contribution [12], we studied the performance of TCP over UBR [13]. We studied several enhanced versions of UBR with three buffer management policies–Early Packet Discard, selective drop using per-VC accounting, and Fair Buffer Allocation. A brief summary of the results is given below.

- **TCP achieves maximum possible throughput when no segments are lost.** To achieve zero loss for TCP over UBR, switches need buffers equal to the sum of the receiver windows of all the TCP connections.

- **With limited buffer sizes, TCP performs poorly over vanilla UBR switches.** TCP throughput is low, and there is unfairness among the connections [2, 3, 4, 5, 11, 16]. The coarse granularity TCP timer is an important reason for low TCP throughput.

- **UBR with EPD improves the throughput performance of TCP.** This is because partial packets are not being transmitted by the network and some bandwidth is saved. EPD does not have much effect on fairness because it does not drop segments selectively [11].

- **UBR with selective packet drop using per-VC accounting improves fairness over UBR+EPD.** Connections with higher buffer occupancies are more likely to be dropped in this scheme. The efficiency values are slightly better than those with EPD.

- **UBR with the Fair Buffer Allocation scheme can improve TCP throughput and fairness** [8]. There is a tradeoff between efficiency and fairness and the scheme is sensitive to parameters. We found R = 0.9 and Z = 0.8 to produce best results for our configurations.

- **TCP synchronization is an important factor that effects TCP throughput and fairness.**

In this contribution, we study the effect of Fast Retransmit and recovery on the performance of TCP over UBR.

2   Fast Retransmit and Recovery

TCP congestion control techniques are called slow start and congestion avoidance [18]. TCP Reno introduces an additional congestion control mechanism called Fast Retransmit and Recovery (FRR). FRR is designed for quick recovery from isolated packet losses. Without FRR, every time a packet is lost, TCP waits for the retransmission timeout and then enters slow start mode. As a result, much time is lost waiting for the retransmission timeout which typically has a granularity of 100–500 milliseconds.

Fast Retransmit and Recovery works as follows. When a packet is lost, the TCP destination sends duplicate ACKs upon the receipt of subsequent packets (these packets are out of sequence since there is a missing packet). When the source receives the third duplicate ACK, it sets SSTHRESH to half of the congestion window (CWND), and retransmits the segment that is missing (from the ACK number). It then reduces its congestion window to half its previous value plus three (for the three ACKs it received). Now for every additional duplicate ACK it receives, the source sends an additional packet (if allowed by the maximum window) and also increments CWND by one segment. When it receives a new ACK (meaning that the destination has received the missing segment) the source sets CWND to SSTHRESH (i.e., half the value of CWND before the fast retransmit and recovery began). Since SSTHRESH and CWND are now equal, the source now enters congestion avoidance mode.

Fast Retransmit and Fast Recovery improve TCP performance when a single segment is lost. However, in high bandwidth links, network congestion results in several dropped segments. In this case, fast retransmit and recovery are not able to recover from the loss and slow start is triggered. Moreover, [7] points out that in some cases retransmission of packets cached in the receiver's reassembly queue result in false retransmits.

In this case, the sender goes into congestion avoidance mode when there is no congestion in the network. As a result, Fast retransmit and recovery are effective only in isolated packet losses.

3   The Simulation Experiment

3.1   Simulation Model

All simulations presented in this contribution are performed on the N source configuration shown in Figure 1. The configuration consists of N identical TCP sources that send data whenever allowed by the window. The switches implement UBR service with optional drop policies described in this contribution. The following simulation parameters are used [17]:
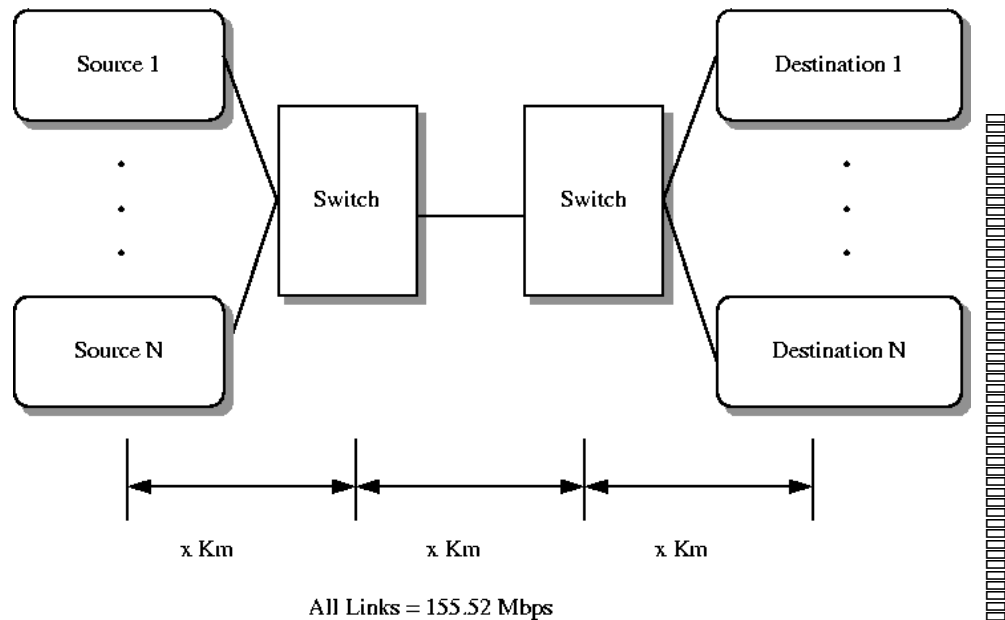

Figure 1: The N-source TCP configuration

- The configuration consists of N identical TCP sources as shown in Figure 1.

- All sources are infinite TCP sources. The TCP layer always sends a segment as long as it is permitted by the TCP window.

- All link delays are 5 microseconds for LANs and 5 milliseconds for WANs. Thus, the Round Trip Time due to the propagation delay is 30 microseconds or 30 milliseconds respectively.

- All link bandwidths are 155.52 Mbps.

- Peak Cell Rate is 155.52 Mbps.

- The traffic is unidirectional. Only the sources send data. The destinations send only acknowledgments.

- TCP Fast Retransmit and Recovery is enabled.

- The TCP segment size is set to 512 bytes. This is the standard value used by current TCP implementations. Larger segment sizes have been reported to produce higher TCP throughput, but these have not been implemented in real TCP protocol stacks.

- TCP timer granularity is set to 100 ms. This affects the triggering of retransmission timeout due to packet loss. The values used in most TCP implementations is 500 ms, and some implementations use 100 ms. Several other studies have used smaller TCP timer granularity and have obtained higher throughput numbers. However, the timer granularity is an important factor in determining the amount of time lost during congestion. Small granularity results in less time being lost waiting for the retransmission timeout to trigger. This results in faster recovery and higher throughput. However, TCP implementations do not use timer granularities of less than 100 ms, and producing results with lower granularity artificially increases the throughput.

- TCP maximum receiver window size is 64K bytes for LANs. This is the default value used in TCP. For WANs, this value is not enough to fill up the pipe, and reach full throughput. In the WAN simulations we use the TCP window scaling option to scale the window to the bandwidth delay product of approximately 1 RTT. The window size used for WANs is 600000 Bytes.

- TCP delay ack timer is NOT set. Segments are acked as soon are they are received.

- Duration of simulation runs is 10 seconds for LANs and 20 seconds for WANs.

- All TCP sources start and stop at the same time. There is no processing delay, delay variation or randomization in any component of the simulation. This highlights the effects of TCP synchronization as discussed later.

3.2   Performance Metrics

The performance of TCP over UBR is measured by the efficiency and fairness which are defined as follows:

$$\text{Efficiency} = (\text{Sum of TCP throughputs})/(\text{Maximum possible TCP throughput})$$

The TCP throughputs are measured at the destination TCP layers. Throughput is defined as the total number of bytes delivered to the destination application divided by the total simulation time. The results are reported in Mbps.

The maximum possible TCP throughput is the throughput attainable by the TCP layer running over UBR on a 155.52 Mbps link. For 512 bytes of data (TCP maximum segment size), the ATM layer receives 512 bytes of data + 20 bytes of TCP header + 20 bytes of IP header + 8 bytes of LLC header + 8 bytes of AAL5 trailer. These are padded to produce 12 ATM cells. Thus, each TCP segment results in 636 bytes at the ATM Layer. From this, the maximum possible throughput = $512/636 = 80.5\% = 125.2$ Mbps approximately on a 155.52 Mbps link.

$$\text{Fairness Index} = (\Sigma x_i)^2/ \ (n \times \Sigma x_i^2)$$

Where x_i = throughput of the ith TCP source, and n is the number of TCP sources

The fairness index metric applies well to the n-source symmetrical configuration. For more general configurations with upstream bottlenecks, the max-min fairness criteria [6] can be used.

4   Results

We performed full factorial simulations for LAN and WAN configurations, with 5 and 15 sources with each of the buffer management policies. For LAN configurations, switch buffer sizes of 1000 cells and 3000 cells were simulated. For WAN configurations switch buffer sizes of 12000 cells and 36000 cells were simulated. Tables 1 and 2 are from our previous contribution [12]. Tables 3 and 4 are the analogous tables with Fast Retransmit and Recovery. The last row of each table gives the column averages of the respective efficiency

and fairness values. This gives a rough quantitative measure of how much each feature of UBR+ improves the efficiency and fairness.

Table 1: UBR+ without Fast Retransmit and Recovery (Efficiency)

Config-uration	Number of Sources	Buffer Size (cells)	UBR	EPD	Selective Drop	FBA
LAN	5	1000	0.21	0.49	0.75	0.88
LAN	5	3000	0.47	0.72	0.90	0.92
LAN	15	1000	0.22	0.55	0.76	0.91
LAN	15	3000	0.47	0.91	0.94	0.95
WAN	5	12000	0.86	0.90	0.90	0.95
WAN	5	36000	0.91	0.81	0.81	0.81
WAN	15	12000	0.96	0.92	0.94	0.95
WAN	15	36000	0.92	0.96	0.96	0.95
Column Average			0.63	0.78	0.87	0.92

Table 2: UBR+ without Fast Retransmit and Recovery (Fairness)

Config-uration	Number of Sources	Buffer Size (cells)	UBR	EPD	Selective Drop	FBA
LAN	5	1000	0.68	0.57	0.99	0.98
LAN	5	3000	0.97	0.84	0.99	0.97
LAN	15	1000	0.31	0.56	0.76	0.97
LAN	15	3000	0.80	0.78	0.94	0.93
WAN	5	12000	0.75	0.94	0.95	0.94
WAN	5	36000	0.86	1	1	1
WAN	15	12000	0.67	0.93	0.91	0.97
WAN	15	36000	0.77	0.91	0.89	0.97
Column Average			0.73	0.82	0.93	0.97

From the average of the efficiency values of vanilla UBR (tables 1 and 3), fast retransmit and recovery seems to improve the efficiency of TCP over UBR (Efficiency = 0.65 without fast retransmit and 0.73 with fast retransmit). However, for the WAN simulations, fast retransmit and recovery hurts the efficiency. This is because in vanilla UBR, congestion typically results in multiple packets being dropped. Fast retransmit and recovery cannot recover from multiple packet losses and slow start is triggered. The additional segments sent by fast retransmit and recovery (while duplicate ACKs are being received) are retransmit during slow start. In WAN links with large bandwidth delay products, the number of retransmitted segments can be significant. Thus, fast retransmit can add to the congestion and reduce throughput. Also, the phenomenon of false retransmits as described in section 2 results in wasted throughput because the source enters congestion avoidance mode by setting the slow start threshold variable SSTHRESH to one-half the congestion window (CWND) value.

The fairness values with fast retransmit and recovery are better for vanilla UBR. This is because, fast retransmit and recovery helps in mitigating the TCP synchronization effects. Sources with fast retransmit and recovery do not wait for the retransmission timeout to trigger, but retransmit the lost packet as soon as they receive duplicate ACKs. Also, for every duplicate ACK received, a new segment is sent if allowed by the window.

The addition of EPD with fast retransmit and recovery results in a large improvement in both fairness.

Table 3: UBR+ with Fast Retransmit and Recovery (Efficiency)

Config- uration	Number of Sources	Buffer Size (cells)	UBR	EPD	Selective Drop	FBA
LAN	5	1000	0.53	0.97	0.97	0.97
LAN	5	3000	0.89	0.97	0.97	0.97
LAN	15	1000	0.42	0.97	0.97	0.97
LAN	15	3000	0.92	0.97	0.97	0.97
WAN	5	12000	0.61	0.79	0.8	0.76
WAN	5	36000	0.66	0.75	0.77	0.78
WAN	15	12000	0.88	0.95	0.79	0.79
WAN	15	36000	0.96	0.96	0.86	0.89
Column Average			0.73	0.92	0.89	0.89

Table 4: UBR+ with Fast Retransmit and Recovery (Fairness)

Config- uration	Number of Sources	Buffer Size (cells)	UBR	EPD	Selective Drop	FBA
LAN	5	1000	0.77	0.99	0.99	0.97
LAN	5	3000	0.93	0.99	1	0.99
LAN	15	1000	0.26	0.96	0.99	0.69
LAN	15	3000	0.87	0.99	0.99	1
WAN	5	12000	0.99	1	0.99	1
WAN	5	36000	0.97	0.99	0.99	1
WAN	15	12000	0.91	0.96	0.99	0.95
WAN	15	36000	0.74	0.91	0.98	0.98
Column Average			0.81	0.97	0.99	0.95

Average efficiency improves from 0.73 without EPD to 0.92 with EPD. Fairness improves from 0.81 to 0.97. Thus, the combination on EPD and fast retransmit can provide high throughput and fairness in configurations similar to those simulated here.

The incremental gain achieved by adding selective drop and fair buffer allocation is small compared to the gain from EPD. Efficiency decreases slightly from 0.92 to 0.89, and fairness improves from 0.97 to 0.99 with selective drop. Fair buffer allocation actually hurts the fairness in one case (fairness = 0.69 for 15 source LAN configuration with 1000 cells buffer size). This is because fair buffer allocation is very sensitive to the parameters, and slightly different parameters could result in much improved performance. In our simulations, we use the parameters that we found best from our previous contribution [12] so that the results can be consistently compared. The combination of early packet discard and fast retransmit and recovery is effective in breaking TCP synchronization, and thus improves fairness and efficiency of the N source symmetrical configurations.

5 Summary

This contribution examines the effect of fast retransmit and recovery on the performance of TCP over UBR+. The following conclusions can be drawn from our simulations of N symmetrical TCP sources.

- Fast retransmit recovers from isolated packet losses faster than slow start. Since it does not depend on a common coarse granularity timer, fast retransmit and recovery helps in breaking TCP synchronization. As a result vanilla UBR performs better in some cases.

- For long links (WAN configuration) fast retransmit actually hurts the throughput. This is because fast retransmit cannot recover from multiple segment losses and transmits segments that are retransmitted by slow start. False retransmits also degrade the throughput performance.

- Early Packet Discard improves the performance of TCP with fast retransmit and recovery, over UBR.

- The incremental gain obtained by adding Fair Buffer Allocation and Selective Drop is much less with fast retransmit and recovery.

The TCP Selective Acknowledgment option (SACK) has been recommended to overcome the shortcomings of fast retransmit and recovery. SACK is expected to improve performance over fast retransmit in cases of multiple packet loss. The effect of SACK on UBR+ is an area of future study.

References

[1] Allyn Romanov, Sally Floyd, "Dynamics of TCP Traffic over ATM Networks."

[2] Chien Fang, Arthur Lin: "On TCP Performance of UBR with EPD and UBR-EPD with a Fair Buffer Allocation Scheme," ATM FORUM 95-1645, December 1995.

[3] Hongqing Li, Kai-Yeung Siu, and Hong-Ti Tzeng, "TCP over ATM with ABR service versus UBR+EPD service," ATM FORUM 95-0718, June 1995.

[4] Hongqing Li, Kai-Yeung Siu, Hong-Yi Tzeng, Brian Hang, Wai Yang, "Issues in TCP over ATM," ATM FORUM 95-0503, April 1995.

[5] Hongqing Li, Kai-Yeung Siu, and Hong-Ti Tzeng, "TCP Performance over ABR and UBR services in ATM," Proceedings of IPCCC'96, March 1996.

[6] J. Jaffe, "Bottleneck Flow Control," *IEEE Transactions on Communications*, Vol. COM-29, No. 7, pp. 954-962.

[7] Janey C. Hoe, "Improving the Start-Up Behavior of a Congestion Control Scheme for TCP," Proceedings of the ACM SIGCOMM, August 1996.

[8] Juha Heinanen, and Kalevi Kilkki, "A fair buffer allocation scheme," Unpublished Manuscript.

[9] Raj Jain, Shiv Kalyanaraman, Rohit Goyal and Sonia FahmySaragur Srinidhi, "Buffer Requirements for TCP over ABR ," ATM Forum/96-0517

[10] Raj Jain, Rohit Goyal, Shiv Kalyanaraman, and Sonia Fahmy, "Performance of TCP over UBR and buffer requirements," ATM Forum/96-0518

[11] Raj Jain, R. Goyal, S. Kalyanaraman, S. Fahmy, F. Lu, and S. Srinidhi, "Buffer requirements for TCP over UBR" ATM FORUM 96-0518, April 1996.

[12] R. Goyal, Raj Jain, S. Kalyanaraman, S. Fahmy, Seong-Cheol Kim, "Performance of TCP over UBR+," ATM Forum/96-1269, October 1996.

[13] Shirish S. Sathaye, "ATM Traffic Management Specification Version 4.0," *ATM Forum/95-0013R10*, February 1996.

[14] Shiv Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, Fang Lu and Saragur Srinidhi, "Performance of TCP/IP over ABR," Proceedings of Globecom. November 1996.

[15] Shivkumar Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, Jianping Jiang and Seong-Cheol Kim, "Performance of TCP over ABR on ATM backbone and with various VBR traffic patterns," ATM Forum/96-1294:

[16] Stephen Keung, Kai-Yeung Siu, "Degradation in TCP Performance under Cell Loss," ATM FORUM 94-0490, April 1994.

[17] Tim Dwight, "Guidelines for the Simulation of TCP/IP over ATM," ATM FORUM 95-0077r1, March 1995.

[18] V. Jacobson, "Congestion Avoidance and Control," *Proceedings of the SIGCOMM'88 Symposium*, pp. 314-32, August 1988.

All our papers and ATM Forum Contributions are available from http://www.cis.ohio-state.edu/~jain