Experiments on TCP over UBR Performance in the EXPERT Testbed

EGIL AARSTAD

Telenor Research & Development, P.O. Box 83, 2007 Kjeller, Norway

BART GIJSEN, NICKY VAN FOREEST

Koninklijke PTT Nederland Research, Leidschendam, The Netherlands

LAURENT JAUSSI

Swiss Federal Institute of Technology (EPFL), Telecom Lab, 1015 Lausanne, Switzerland

MARTIN LORANG

University of Stuttgart, Inst. of Communication Networks and Computer Engineering, Pfaffenwaldring 47, 70569 Stuttgart, Germany

JORDI NELISSEN

Alcatel Corporate Research Center, Francis Wellesplein 1, B-2018 Antwerp, Belgium

Abstract. The UBR service is intended for non-real-time applications that do not require guaranteed QoS commitments. With additional, relatively inexpensive control functions such as packet discard schemes, UBR could become a cost-effective alternative for the transmission of data traffic, offering a straightforward and flexible solution as opposed to nrt-VBR and GFR applying traffic contracts as well as ABR with its sophisticated and complex rate-control protocol. This paper presents the results obtained from a comprehensive set of experiments with TCP over UBR, comprising measurements taken on different protocol layers. The goal is to experimentally investigate the performance of UBR to carry TCP traffic, to evaluate the performance gain achievable by packet discard schemes and TCP parameter tuning, and in a final step, to relate the measurements to simulation results.

1 Introduction

Though the Available Bit Rate (ABR) service category [1] with its sophisticated rate control and the recently proposed Guaranteed Frame Rate (GFR) service category [2] seem to be superior to UBR at first glance, Unspecified Bit Rate (UBR) [1] with additional, relatively inexpensive control functions such as shaping or intelligent packet discard schemes can be a cost-effective alternative for the transmission of TCP (Transmission Control Protocol) data traffic. TCP over UBR has been subject to many simulation studies. As important such studies are, as incomplete the outcome is, because a lot of the dynamics of real equipment is neglected. Therefore also some experimental work on TCP over ATM has been conducted focusing either on end system and TCP parameter tuning [3, 4], or focusing on specific control functions related to ATM [5, 6, 7].

So far, neither the mutual influences of both have been investigated nor statistics from different layers have been shown in detail. This paper evaluates the TCP over UBR performance for a large set of parameters both at the TCP and ATM layer in a local ATM network. Section 2 starts with a brief overview of the TCP flow control mechanism. The local testbed configuration is described in section 3 and section 4 addresses the outcome of a vast measurement campaign studying the influence of ATM layer and TCP parameters and control functions, ranging from the ATM switch buffer size, Early Packet Discard (EPD) with different thresholds and TCP built-in mechanisms such as Fast Retransmit/Fast Recovery, Delayed Acknowledgement, and Timer Granularity. Section 5 the relation between these results and simulation studies, either found in literature or carried out to investigate several effects observed in the experiments, is discussed.

2 Overview of TCP and Early Packet Discard

TCP provides end-to-end window based flow control by means of several congestion control algorithms such as *Slow Start*, *Congestion Avoidance*, *Fast Retransmit and Recovery* (RFC2001). All these algorithms assume that packet loss indicates congestion in the network which has to be resolved by an appropriate rate decrease. Since TCP increases its transmission rate until congestion occurs, TCP applications can hardly specify a traffic contract unless they access directly shaping capabilities in the network interface card. For this reason, UBR service is a natural choice for TCP traffic, in particular if it is enhanced by additional traffic control functions such as EPD, per-VC queueing, per-VC accounting, Weighted Fair Queueing [8] etc. This section surveys TCP over UBR and points out the issues which give the justification for the measurement campaign carried out and presented in this paper.

2.1 Transmission Control Protocol

TCP's flow control and packet retransmission provides a reliable byte stream service for data transfers over a best effort network. Two types of flow control can be distinguished, flow control imposed by the receiver and flow control imposed by the sender. Flow control imposed by the receiver is achieved by the so called sliding window mechanism of TCP, where the maximum amount of outstanding data in the network is limited by the receiver's advertised window, merely to avoid buffer overflow at the receiver side. The combination of *Slow Start*, *Congestion Avoidance*, *Fast Retransmit and Recovery* (FRR) realise flow control at the sender by maintaining and updating a second window, Congestion Window (*cwnd*), which forces a rate decrease when congestion is indicated by the loss of segments.

At connection set-up, the Congestion Window, maintained in byte, is set to one TCP Maximum Segment Size (MSS). At the receipt of a segment, the TCP receiver returns an acknowledgement (ACK) to the sender. Each time the sender receives an ACK, the Congestion Window is increased by one MSS thus doubling the Congestion Window every round-trip time (RTT). This mechanism provides an exponential increase regulated by the returning ACKs. This exponential increase of the Congestion Window slows down to a linear increase when the Slow Start Threshold, *sstresh*, is crossed. The Congestion Avoidance Phase is entered.

TCP measures the Round Trip Time regularly and uses this measurement to adjust a Retransmission Time-Out value RTO. When this timer expires before the peer acknowledges new data, the sending TCP enters Slow Start again and retransmits starting from the first unacknowledged byte until all the lost segments are acknowledged.

Besides time-out, the receipt of several duplicate ACKs also indicates packet loss. An ACK is sent by the receiver to acknowledge received data, but also to request the next segments. Once a segment is lost, the receiver will send duplicate requests, hence duplicate ACKs. Modern TCP implementations execute Fast Retransmit and retransmit the lost packet after the receipt of normally three (assuming that sequence integrity within the network is not guaranteed but the usual case) duplicate ACKs, without awaiting a time-out. Fast Retransmit is immediately followed by Congestion Avoidance instead of Slow Start (Fast Recovery).

According to RFC1122, TCP should also implement delayed acknowledgement, but an ACK should not be excessively delayed. In particular, the delay must be less than 0.5s (ensured by a timer in the receiving TCP) and there should be an ACK for at least every second segment. By combining ACKs with window updates or data in one TCP segment, the Delayed Acknowledgement strategy substantially reduces protocol processing overhead by decreasing the total number of packets to be produced by the receiver.

With the limited information contained in cumulative acknowledgements, currently realised in most TCP implementations, a sender can only learn about a single lost packet per round-trip time. Especially in connections over large distances, this uncertainty leads to dramatic performance drops and unnecessary retransmissions. In order to avoid these undesired effects, Selective Acknowledgement (SACK) options (RFC2018) have been proposed to allow the receiver to request the retransmission of only the lost segments, which is believed to be much more efficient.

2.2 Early Packet Discard

If a cell of an AAL5 frame must be dropped because of buffer overflow, there is no reason for transmitting subsequent cells of this frame (even if buffer space has become available again) because the destination AAL5 cannot reassemble the frame correctly anymore. Consequently the switch may apply Partial Packet Discard (PPD) [9] to discard the remainder of the frame except of the EOM cell at the end of the AAL frame, which is necessary at the destination to delineate the beginning of a new frame. The frame discard is partial, because, at the time of cell loss, some cells of the affected frame may already have been stored in the switch buffer or even transmitted.

Early Packet Discard (EPD) enforces a switch to drop entire frames prior to overflow. If the first cell of an AAL5 frame arrives at a switch when the buffer exceeds a certain threshold, the first as well as all subsequent cells of the frame are discarded.

Several approaches for setting the threshold can be distinguished. Usually, EPD uses a fixed, global threshold, e.g. [9]. In [10], EPD uses a variable threshold, which is a function of the number of already accepted frames. Lakshman, Neidhardt, and Ott [11] apply Random Early Detection (RED) to EPD although this scheme was proposed for gateways in IP networks. When RED is employed, the ATM switch associates a drop probability with each buffer occupancy level. When the first cell of an AAL5 frame arrives, it is dropped with exactly this probability. Turner [12] introduces hysteresis into the selection process for frame discards. In [13, 17], EPD is combined with per-VC accounting and per-VC queueing in order to improve fairness. When the EPD threshold is reached, no new frame of a VC exceeding its fair buffer share is allowed to enter the buffer. Fair Buffer Allocation [14] is a similar selective drop policy with a smoother rejection function.

This paper focuses on basic PPD and EPD with a fixed, global threshold. Romanow and Floyd [9] reported a positive impact particularly of EPD on TCP performance, however, measurement campaigns covering a large range of parameters have never been carried out. The different enhancements to these schemes, as promising they might be, have not been implemented on any of the switches available at the EXPERT testbed site.

3 Testbed Configuration for the TCP over UBR Experiments

Four Sun SPARC 20 workstations, four Pentium PCs and a Fore ASX200 ATM switch build the local ATM network based on 100 Mbit/s TAXI and 155.52 Mbit/s optical multimode interfaces depicted in figure 1. Both the workstations and the PCs are equipped with network interface cards, administration commands of which allow to encapsulate IP packets in AAL5 and to route them to particular IP destinations in Permanent Virtual Connections (PVC). The benchmarking tool *ttcp* accesses the TCP/IP protocol stack through the socket API provided by Solaris 2.5.1 on the workstations and Linux 2.0.25 on the PCs respectively and measures the TCP memory-to-memory throughput. *ttcp* controls the size of two kinds of buffers which are significant to measured performance, namely the socket buffer size and the user data buffer size. In the basic configuration, four TCP connections share a bottleneck of 37.44 Mbit/s minus the overhead of ATM, AAL5, RFC1483 and TCP/IP. The Solaris 2.5.1 workstations are configured as senders and the Linux PCs are configured as receivers. All the senders behave as greedy sources sending bulk TCP traffic to the receivers simultaneously during 180 seconds. Only ACKs flow in the reverse direction. In all measurements presented here, the user data buffer size is set to 16 kB and the TCP maximum window size is set to 52224 byte (which corresponds to about 1000 ATM cells. Previous

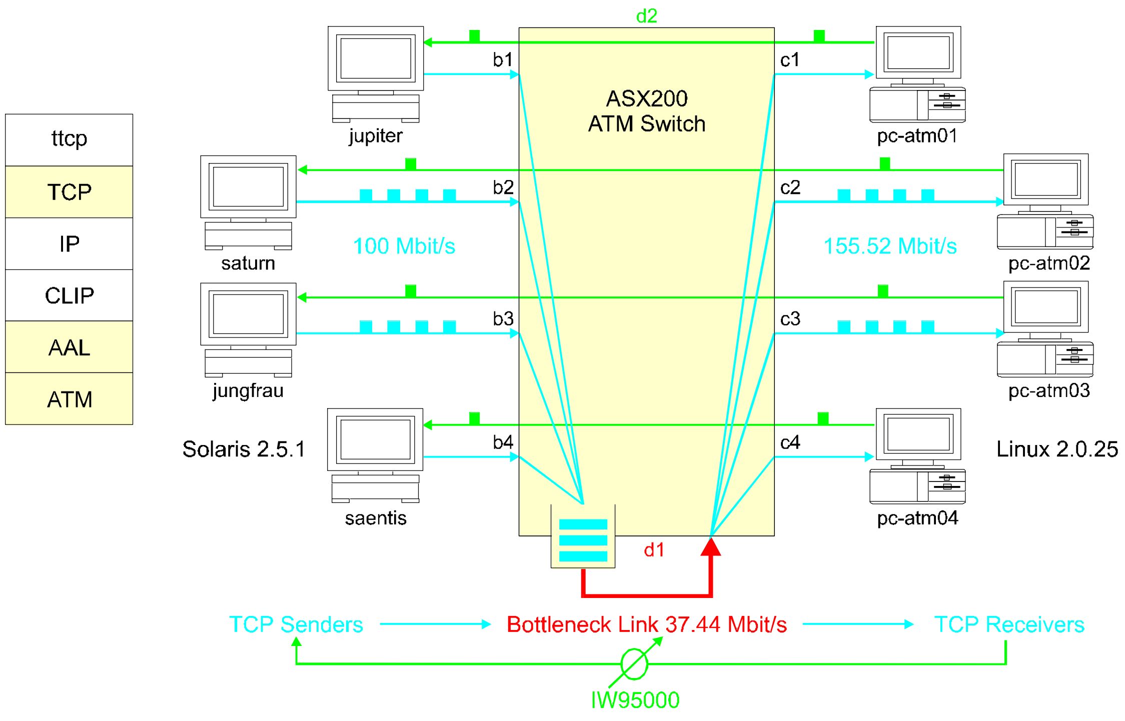


Fig.1: Configuration for TCP over UBR experiments

measurements [15] have shown that this allows each connection to use the full bottleneck capacity on its own if there is no contention.

In order to analyse the flow of data and acknowledgements, the ASX 200 redirects the connections through GN Nettest's interWATCH95000 network analyser.

The ASX200 implements per VC queueing (C-series network modules) with a weighted round robin service discipline within the service categories (Constant Bit Rate, Variable Bit Rate, Available Bit Rate - Unspecified Bit Rate). Additionally, buffer space can be reserved for exclusive usage by the different service categories. The remaining buffer space not reserved is shared among all connections. The EPD mechanism implemented in the ASX200 works as follows: The EPD threshold, *EPDth*, is defined on the shared buffer space. Whenever the fill level of the shared buffer space is greater than *EPDth* when the first cell of an AAL5 frame arrives, the incoming AAL5 frame will be fully discarded except of its last cell containing the EOM field which allows to delimit the different AAL5 frame. In the case a cell is lost in the middle of an AAL5 frame, the remaining cells of the frame are discarded, i.e. subject to Partial Packet Discard (PPD).

4 Experimental Results

In the scenario depicted in figure 1, four parallel TCP sessions contend for the same UBR switch output buffer. As mentioned above, TCP congestion control assumes that packet loss indicates congestion in the network which has to be resolved by an appropriate rate decrease. Waiting for time-outs or duplicate acknowledgements after packet loss can cause TCP to stop the transmission, and therefore, the switch buffer for UBR has to be dimensioned appropriately to ensure that packet loss can be kept within reasonable bounds. A number of experiments have been carried out to obtain guidelines for this buffer dimensioning. Performance is assessed in terms of aggregate TCP goodput and fairness among TCP connections.

- The *efficiency* is measured in terms of the aggregated goodput achieved by the involved sessions.
- The *fairness* is quantified through the fairness index defined as:

$$Fairness\_index = \frac{(\sum gi)^2}{N * \left(\sum gi^2\right)} \tag{1}$$

where N is the number of connection (N = 4 here), and g_i is the goodput of connection i (i = 1...4). This fairness index [16, 17] is well suited for symmetric configurations and relatively simple.

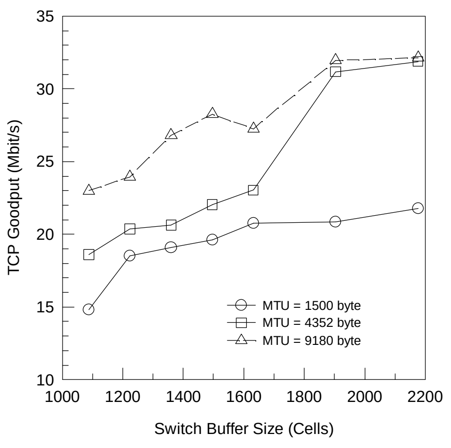
- The *Cell Loss Ratio* (CLR) is measured by the switch on both connections (total CLR).

The *AAL 5 Frame Loss Ratio* (FLR) is measured for each connection separately.

While measuring the performance of TCP over UBR dependent on the switch buffer size and the MTU with these same performance measures, Early Packet Discard has been disabled or enabled with several buffer thresholds, Delayed Acknowledgement been switched off and on, and the TCP's Timer Granularity has been reduced far below the default value.

4.1 TCP Performance over Plain UBR

TCP achieves the maximum goodput when no packets are lost, which means that the switch buffer size at the multiplexing point has to be equal or larger than the sum of the receiver windows of all TCP connections involved. In this case, the goodput is maximal and the bandwidth is shared almost perfectly among the connections. Figure 2 shows the goodput, cell and frame loss ratios (CLR and FLR) for a switch buffer size between one and two TCP windows and for different Message Transfer Units (MTU) in the case of two TCP connections sharing the bottleneck link. Here, a configuration different from figure 1 with four hosts and only two TCP connections has been used. As before, Solaris 2.5.1 workstations act as senders, however, SunOS 4.1.4 workstations, and not Linux PCs, are configured as receivers.



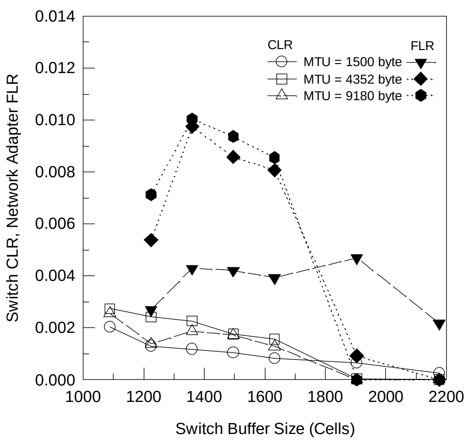
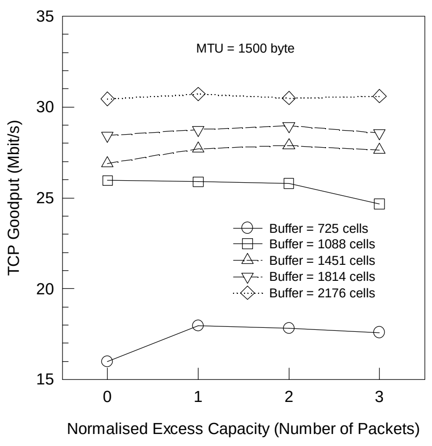


Fig. 2a: TCP goodput as a function of the buffer size *Fig. 2b: FLR and CLR as a function of the buffer size*

As expected, the goodput increases and the CLR (FLR) decreases as a function of the switch buffer with increasing buffer size up to the point where the buffer is large enough to prevent any cell losses (see CLR for buffer 2176 cells and MTU 4352 and 9180). As opposed to simulation results [9], the goodput is larger with a larger MTU. This is caused by the relatively high processing overhead for the transmission of smaller packets. This processing overhead compensates the fact that connections with larger frames are more affected because the loss of a single cell already corrupts a whole frame. The processing overhead is but naturally bound to the specific end systems and different behaviour can and has been observed in different hardware and software configurations. Figure 2 shows as well that a larger switch buffer is required to ensure zero loss for smaller packets because the effective maximum TCP window depends on the MTU (and Maximum Segment Size, MSS) as discussed in [15]. The FLR is significantly larger than the CLR since, as mentioned above, the loss of a single cell within a frame inevitably leads to the discard of the whole AAL5 frame at the destination. Only if all cells of a corrupted frame where dropped at the switch (e.g. using EPD), the CLR in the switch would equal the FLR at the receivers (leaving the EOM cells out of consideration). The difference between the FLR and CLR represents the ratio of cells that belong to corrupted frames and which are still transmitted to the destination. Note that this ratio is for small to medium buffer larger than the CLR itself, which means that more cells of corrupted frames are sent to the destination than cells lost at the switch.

4.2 TCP Performance over UBR with Early Packet Discard

As observed in the previous section, a significant ratio of cells that belong to corrupted packets arrive at the destinations. Thus, performance improvements can be expected by using EPD. In order to quantify the goodput improvements that can be achieved by using EPD, experiments have been carried out with four TCP connections, switch buffers ranging from 725 to 2176 cells and with a variable normalised buffer excess capacity. The latter, the normalised buffer excess capacity *b*, is



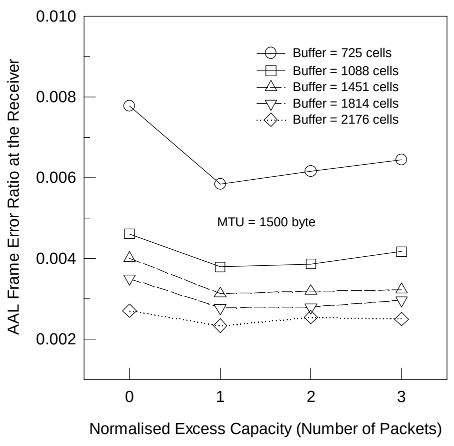


Fig. 3a: TCP goodput as a function of the normalised excess buffer capacity b for MTU = 1500 byte

Fig. 3b: FLR at the receivers as a function of the normalised excess capacity b

defined by the following relation: *EPDth = Switch buffer - b MTU*. In the experiments described below, a normalised excess capacity of 0, 1, 2 and 3 has been used. An excess capacity of 0 is equivalent to PPD since no buffer capacity is available to buffer incoming cells from partially transmitted packets and thus, because the EPD threshold is equal to the buffer size, cell discard is triggered upon buffer overflow.

Figure 3 shows the goodput and FLR as a function of the normalised buffer excess capacity for different switch buffer size values. As can be observed, EPD is only marginally improving the goodput (apart from a 725 cells buffer where a slightly better improvements is seen). Furthermore, any attempt to increase the normalised excess capacity beyond 1 results in either a marginal increase (for buffers of 1451 and 1814 cells) or even a decrease (buffers of size 725 and 1088 cells) of the goodput.

These marginal improvements and even goodput decrease respectively may be explained by means of figure 3b which shows that the AAL5 frame loss ratio actually increases when the normalised excess capacity is increased beyond 1. For all curves in figure 3, the total buffer size is kept constant while the EPD threshold is decreased to increase the normalised capacity. Therefore, figure 3 indicates that the increase of the normalised excess capacity is rather seen as a reduction of the effective buffer than as an improvement of the efficiency of EPD to filter out corrupted packets. The resulting performance drop is even more obvious for a MTU of 9180 byte, as shown in figure 3c. In this case, the reduction of the effective buffer already yields in a goodput decrease for a normalised capacity *b* equal to 1.

Another effect, and probably the most important one, which might prevent significant performance improvements with EPD, is the link idle time due to TCP's retransmission time-out. In the measurements of figure 3, the value of the retransmission timer granularity is set to 200 ms, the default value for Solaris. When losses are such that Fast Retransmit and Recovery (FRR) is not able to retransmit the missing segments and to recover quickly from packet loss, a considerable amount of time is wasted waiting for a time-out. To test this hypothesis, the series of experiments has been repeated with a lower timer granularity of 20 ms. Figure 4 summarises the main results for TCP goodput as a function of the buffer size for both values of the timer granularity (20 and 200 ms).

As can be observed, using a lower value for the timer granularity improves the goodput. On the other hand, the impact of EPD (not shown here) is again only marginal, if not negative, with the same typical behaviour as show in figure 3. This indicates that the bottleneck link is more often idle than occupied by corrupted packets and therefore EPD cannot improve the situation. In this case, goodput performance improvements can be obtained by improving TCP’s reactivity to losses (lower timer granularity, selective retransmission for example) rather than ATM layer packet discard schemes.

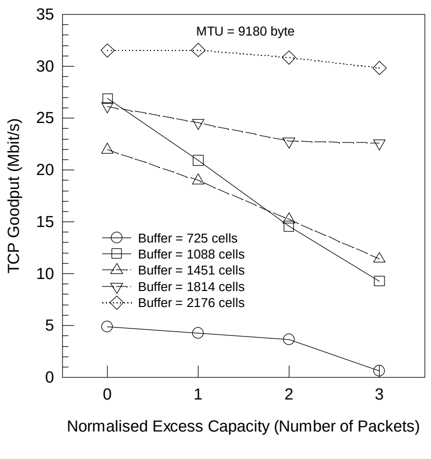


Fig. 3c: TCP goodput as a function of the normalised excess buffer capacity b for MTU = 9180 byte

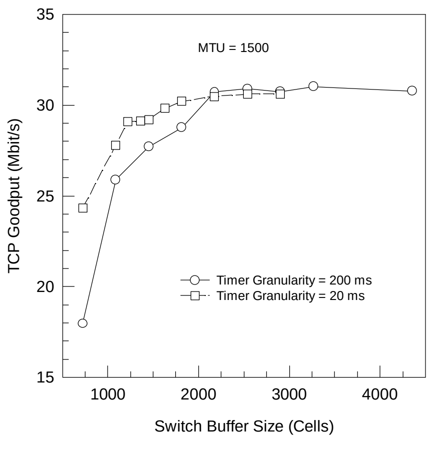


Fig. 4: TCP goodput as a function of the buffer size for different timer granularity values

4.3 Effect of TCP's Fast Retransmit and Recovery Threshold

Currently, there is no way for the receiving TCP entity to tell the peer that a segment is missing unless selective acknowledgement (SACK) options are supported. Since SACK is not included in most TCP implementations, this case is not considered here. Furthermore, TCP cannot acknowledge out-of-order data. All the receiver can do is to continue sending the acknowledgement of the last segment received in the correct order. Instead of waiting for the retransmission time-out and then entering another slow start phase, modern TCP implementations perform the Fast Retransmit/Fast Recovery algorithm. Since in router based IP networks duplicate acknowledgements may either be caused by the arrival of out-of-order IP packets or by packet loss, only three or more duplicate acknowledgements in a row are interpreted as a strong indication that a segment has been lost. Slow Start is not needed in this case because the receipt of the duplicate acknowledgements also shows that there is still data flowing between the two ends. Unlike Slow Start and Congestion Avoidance, the Fast Retransmit algorithm follows a concept of conservation of packets. This means that a new segment will not be injected into the network until an old segment has left. When finally an acknowledgement confirms the successful transmission of new data (including the retransmitted segment), the Fast Retransmit phase is finished and Congestion Avoidance is entered (Fast Recovery). For a more detailed description of the congestion control of TCP refer to RFC2001 or [18]. With ATM, TCP can benefit from the frame sequence integrity guaranteed by ATM. Hence, Fast Retransmit can already be invoked on the receipt of the first duplicate acknowledgement.

In order to investigate the potential of Fast Retransmit/Fast Recovery, the duplicate acknowledgement (*dupack*) counter (which triggers the Fast Retransmit) was set to 1, 3, and 10 respectively (10 being the maximum value for the duplicate acknowledgement counter in Solaris 2.5.1) .

Figure 5 shows the impact of the value of the *dupack* counter on the TCP throughput for a MTU

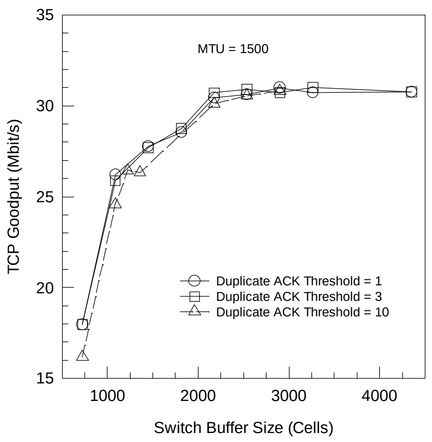


Fig. 5: TCP goodput as a function of the buffer size for different FRR thresholds with EPD (b = 1)

size of 1500 byte and in the case of EPD with an normalised excess buffer of 1. The measurements do not point out a performance increase when the *dupack* counter is reduced from 3 to 1 for an MTU of 1500 byte. For a MTU of 9180 byte, a significant performance increase with the more aggressive TCP settings, i.e. where *dupack* equals 1, has been observed. Due to temporary problems with a duplicate ACK counter of 10 in this case, these results are not yet presented here. However, with a MTU of 9180 byte and a TCP window size of 52224 byte, the effective TCP window is only about six TCP segments (due to the delayed acknowledgement algorithm). This reduces the amount of TCP segments that can potentially trigger three subsequent duplicate ACKs at the receiver. Consequently FRR is triggered less often and TCP throughput is lower. With an MTU of 1500 byte, the effective TCP window is equal to about 33 segments which is largely sufficient to trigger three subsequent duplicate ACKs.

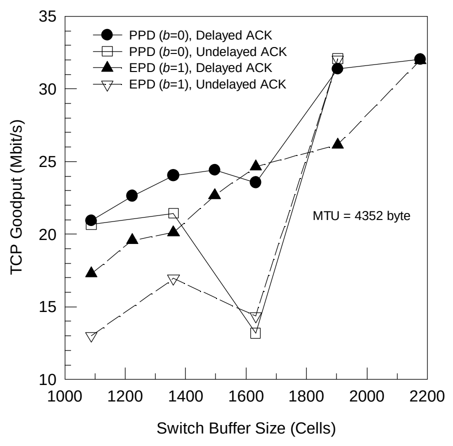
From our simulations, using the same configuration and parameter settings, we learn however that a duplicate ACK counter of 10 does not switch off the FRR mechanism with a MTU of 1500 byte as it does with an MTU of 9180 byte.

4.4 Effect of the Delayed Acknowledgement Algorithm

In bulk TCP transfers, returning acknowledgements do have a strong influence on the segment stream sent by a TCP host because they move on the window of the sending TCP. The more bytes a receiver acknowledges at once, the more data can be sent as a reaction, possibly yielding a small burst of TCP segments. Enabling the Delayed Acknowledgement strategy as proposed in RFC1122 thus might have a negative influence on the packet drop and discard behaviour at the bottleneck switch. In order to investigate the impact of Delayed Acknowledgement, measurements with four workstations, two Solaris 2.5.1 sending and two SunOS 4.1.4 receiving workstations, have been carried out (the same configuration as for section 4.1). The default SunOS TCP refrains from sending acknowledgements until the receiver window has slid more than two maximum packets or 35 % of the maximum TCP window (delayed acknowledgement, DACK). Additional timer generated acknowledgements limit the delay to 200 ms which however only plays a minor role in the configuration this paper concerns. Since SunOS TCP never acknowledges above-sequence segments immediately as it should, Fast Retransmit/Fast Recovery is not effective in this section.

The comparison of TCP with and without DACK both for PPD and EPD in Fig. 6a and 6b reveals that TCP without DACK suffers from relatively low goodput and a significantly lower fairness index. Obviously, TCP without DACK allows a dominant session to conserve or even increase its resource share in the network. Conversely, DACK inserts delay and randomness to the transmission of packets and thus the acknowledgement driven congestion window increase is smoothed. Since the ACK for every second packet is deferred, DACK prevents the synchronous sequence: „packet-ACK-new packet“. The information that a segment of its own has left the network does not return immediately back to the source. And when it returns, the sending TCP does not increase the congestion window twice as it would if two subsequent acknowledgements had arrived: TCP is less aggressive with DACK.

If the buffer space exceeds the size of the TCP window, which is the case for a switch buffer of 1088 cells in figure 6a and 6b, a connection starting first may in any case occupy buffer corresponding to its window size thus ensuring that its TCP congestion control works very efficiently compensating the low goodput of the other connection. TCP without DACK tends to maintain synchronous, deterministic periods and thus this dominance. Acceptable overall goodput despite unfairness is possible, as can be seen for a switch buffer of 1360 cells. Without DACK, TCP's congestion control works inefficiently when the buffer is increased further to 1632 cells and the second connection comes into play. The higher fairness for EPD suggests that EPD adds randomness to the system similar to DACK.



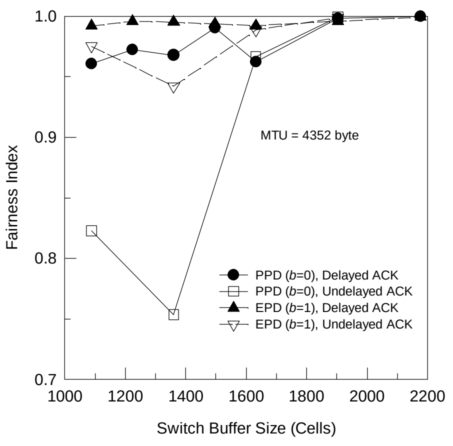


Fig. 6a: TCP goodput for delayed/undelayed ACK *Fig. 6b: TCP fairness for delayed/undelayed ACK*

5 Relation with simulation studies

Several papers have reported results from simulations on the performance and buffer requirements of TCP over UBR [9, 11, 19, 20, 21]. These papers mainly concentrate on TCP throughput performance, efficiency and fairness both in LAN and WAN networks. The authors of this paper also have been studying TCP over UBR performance with independent simulations of the experimental configuration with four sources using the same parameter settings as during the measurements. It is interesting to notice that all of these simulations, including the authors', show results which are more optimistic than real-world measurements. Thus a validation of simulation tools by means of measurements is an open issue. Nevertheless, simulations are particularly interesting when trying to study isolated effects of a specific parameter setting, but one should be very careful when trying to extrapolate simulation results to real-world networks.

6 Conclusions

This paper summarises the most important results of a vast measurement campaign to study the performance of TCP over UBR. Packet length, switch buffer, Partial Packet and Early Packet Discard with different thresholds, and many TCP protocol parameters have been tuned or varied to gain insight into the manifold relevant parameters impacting the performance in a local ATM network.

TCP only benefits little from the most important UBR control feature already implemented in ATM switches, Early Packet Discard. This is contrary to what has been reported in many simulation studies with similar configurations as the one used here. A reduction of the timer granularity to increase the reactivity of TCP apparently has a stronger influence on the performance than the modification of the EPD threshold.

Of course, Early Packet Discard might be more beneficial in larger networks with several hops, however the dicrepancy between measurements and simulations in the local configuration is at least so significant that it needs further investigation. Also bugs or unusual implementation details in the respective TCP versions have to be identified and studied. The authors of this paper will continue using both public available and self-developed simulators to find final explanations for the results obtained for TCP's Delayed Acknowledgement and Early Packet Discard. Future work will also focus on an experimental comparison between TCP (with possibly modified congestion control) over ABR and default TCP over UBR

References

[1] ATM Forum: *Traffic Management Specification Version 4.0*. AF-TM-0056, ATM Forum Technical Committee, April 96.

[2] R. Guerin, J. Heinanen: *UBR+ Service Category Definition*. AF-TM 96-1598, December 1996

[3] K. Moldeklev, E. Klovning, Ø. Kure: *The Effect of End System Hardware and Software on TCP/IP Throughput performance over a local ATM network.* Telektronikk, Vol. 91, No.2/3 - 1995, pp. 155-165, Kjeller, Norway, 1995.

[4] K. Moldeklev, P. Gunningberg: *How a Large MTU Causes Deadlocks in TCP Data Transfers.* IEEE/ACM Transactions on Networking, Vol. 3, No. 4, pp. 409-422, August 1995.

[5] Ewy, Evans, Frost, Minden: *TCP Experiences in the MAGIC Testbed*. Telecommunications & Information Sciences Laboratory, University of Kansas, http://www.tisl.ukans.edu/Workshops /ATM_Performance, 1996.

[6] Lazarou, Frost, Evans, Niehaus: *Using Measurements to Validate Simulation Models of TCP/IP over High Speed Wide Area Networks*. Telecommunications & Information Sciences Laboratory, University of Kansas, http://www.tisl.ukans.edu.

[7] Report from COAST phase 1: *Packet Discard Experiments with Cisco LS1010*. Available via anonymous ftp at robur.slu.de, ftp/pub/SUNET/fas1c.rapport.ps, September 1996.

[8] H. Zhang: *Service Disciplines for Guaranteed Performance Service in Packet Switching Networks*. Proceedings of the IEEE, Vol. 83, N-10, Oct. 1995.

[9] A. Romanov, S. Floyd: *Dynamics of TCP over ATM Networks*. IEEE JSAC, vol. 13, No 4, pp.633-641, May 1995.

[10] K. Kawahara, K. Kitajima, T. Takine, Y. Oie: *Performance Evaluation of Selective Cell Discard Schemes in ATM Networks.* Proceedings of the IEEE INFOCOM'96, Vol.. 3, pp. 1054-1061, San Francisco, March 1996.

[11] T. V. Lakshman, A. Neidhardt, T. J. Ott: *The Drop from Front Strategy in TCP over ATM.* Proceedings of the IEEE INFOCOM’96, Vol. 3, pp. 1242-1250, San Francisco, March 1996.

[12] J. S. Turner: *Maintaining High Throughput during Overload in ATM Switches.* Proceedings of the IEEE INFOCOM’96, Vol. 1, pp. 287-295, San Francisco, March 1996.

[13] H. Li et al.: *On TCP Performance in ATM networks with per-VC early packet discard mechanisms*. Computer Communications 19, November 1996.

[14] J. Heinanen, K. Kilkki: *A Fair Buffer Allocation Scheme.* Available via anonymous ftp://lohi.dat.tele.fi/atm/articles/heinanes-fair-buffer.ps.gz.

[15] EXPERT WP4.2: *First Results from Trials of Optimised Traffic Control Features*. ACTS 094 Deliverable 10, March 1997.

[16] R. Goyal et al.: *Further results on UBR+: Effect of Fast Retransmit and Recovery*. AF-TM 96-1761, December 1996.

[17] R. Goyal et al.: *Performance of TCP over UBR+*. AF-TM 96-1269, October 1996.

[18] R. W. Stevens: *TCP/IP Illustrated, Vol. 1: The Protocols*. Addison Wesley Publishing Company, Reading, Massachusetts, 1994.

[19] R. Jain et al.: *Performance of TCP over UBR and Buffer Requirements*. AF-TM 96-0518, April 1996.

[20] R. Goyal et al.: *UBR Buffer Requirements for TCP/IP over Satellite Networks.* AF-TM 97-0616, 1997.

[21] H. Li et al: *A Simulation Study of TCP Performance in ATM Networks with ABR and UBR Services in ATM LANs*. IEICE Trans. Comm., Vol. E79-B, No 5, pp.658-667, May 1996.