## Copyright Notice



Institute of Communication Networks and Computer Engineering
University of Stuttgart
Pfaffenwaldring 47, D-70569 Stuttgart, Germany
Phone: ++49-711-685-68026, Fax: ++49-711-685-67983
Email: mail@ikr.uni-stuttgart.de, http://www.ikr.uni-stuttgart.de

# A Detailed Experimental Performance Evaluation on TCP over UBR

LAURENT JAUSSI
Swiss Federal Institute of Technology (EPFL), Telecommunications Laboratory
1015 Lausanne, Switzerland
Email Laurent.Jaussi@epfl.ch

MARTIN LORANG
University of Stuttgart, Inst. of Communication Networks and Computer Engineering
Pfaffenwaldring 47, 70569 Stuttgart, Germany
Email lorang@ind.uni-stuttgart.de

JORDI NELISSEN
Alcatel Alsthom Corporate Research Centre, Network Architecture, Traffic Technology Team
Francis Wellesplein 1, B-2018 Antwerp, Belgium
Email jordi.nelissen@ alcatel.be

**Abstract :** The ATM service category UBR 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 that applies stricter traffic specifications 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, to study the influence of the TCP implementation, and in a final step, to relate the measurements to simulation results.

**Keywords :** TCP, ATM Traffic Control, Unspecified Bit Rate, Early Packet Discard.

## 1 Introduction

TCP over UBR [1] has been subject to many simulation studies [2-6]. The main outcome of those studies is that large UBR buffers (larger than one TCP window size) are required to achieve reasonable performance, that packet discard mechanisms (such as EPD) improve the effective usage of the bandwidth and that more elaborated service and drop strategies than FIFO+EPD, such as per VC queuing and accounting, might be required at the bottleneck to ensure both high efficiency and fairness. Though all studies clearly indicate that a buffer size not less than one TCP window size is required, the amount of extra buffer required naturally depends on the packet discard mechanism used and it is still an open issue. Most of the studies made to date are based on simulation results and, according to the authors' knowledge, no experimental studies are available that clearly demonstrate the performance of TCP over UBR and UBR with EPD when several connections are multiplexed. As important simulation studies are, as incomplete the outcome is, because a lot of the dynamics of real equipment is neglected. Several research groups however conducted experimental work on TCP over ATM focusing either on end system and TCP parameter tuning [7], [8], or focusing on specific control functions related to ATM [9-11]. 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 within the framework of the ACTS project EXPERT. The main goals of the experiment campaign presented here were (1) to measure the performance of TCP over plain UBR, UBR with PPD and EPD, in terms of efficiency and fairness as a function of the switch buffer size and EPD threshold, (2) to estimate experimentally the amount of buffering required to achieve good efficiency and fairness, and (3) to measure the impact of various TCP/IP parameters such as MTU size Fast Retransmit and Recovery threshold.

This paper is organised as follows: Section 2 starts with a brief overview of TCP and explains Early Packet Discard, a widely implemented buffer acceptance mechanism in ATM switches. In section 3, the testbed configuration and measurement set-up are explained. Section 4 presents the outcome of a vast TCP measurement campaign studying both the influence of ATM and TCP parameters and control functions. Finally, section 5 compares and complements the experimental results with simulations.

## 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* [12]. 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 shaping capabilities on the network interface card. For this reason, the UBR service category is a natural choice for TCP traffic, in particular if it is enhanced by additional traffic control functions such as EPD, per-VC queuing, per-VC accounting and Weighted Fair Queuing [13] for example. 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 error control 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 imposed by 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 [12].

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 causes an exponential increase of the Congestion Window controlled by the returning ACKs. This exponential increase slows down to a linear one when the Slow Start Threshold, *sstresh*, is crossed. Then the Congestion Avoidance Phase is entered

In addition, 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 as at the start-up of a connection and retransmits a data segment starting from the first unacknowledged byte.

In order to provide a more efficient retransmission mechanism in networks with moderate packet losses, the Fast Retransmit and Recovery algorithms were proposed in 1990 and are now integrated in most TCP versions in use, although still with some mistakes (see [14] for a discussion of different TCP implementations). With Fast Retransmit, TCP uses the fact that the receiver is sending duplicate ACKs for each out-of-sequence segment, thus indicating that a segment might be lost. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then initiates a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire. After Fast Retransmit sends what appears to be the missing segment, TCP changes to a Congestion Avoidance instead of another Slow Start phase. This is the Fast Recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows. The reason for not performing slow start in this case is that the receipt of duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver usually only generates a duplicate ACK when another segment is received, that segment has left the network and is in the receiver buffer. Hence, there is still data flowing between the two connection endpoints, and TCP does not need to reduce the flow abruptly by going into Slow Start.

According to [15], TCP should also implement a delayed acknowledgement strategy, 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.

Furthermore, 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 [16] have

been proposed to allow the receiver to request the retransmission of only the lost segments, which is expected to be much more efficient. However, these extensions have not been widely deployed in current systems and have not been used for the experiments described here.

### *2.2 Early Packet Discard*

Previous studies of TCP throughput behaviour on ATM networks reported that poor performance of TCP over plain ATM can be caused by the well-known problem of fragmentation. When an ATM cell is dropped at a switch, the rest of a higher layer packet is still transmitted to the destination thus wasting bandwidth. Consequently if a cell of an AAL5 frame must be dropped because of buffer overflow, there is no reason for transmitting subsequent cells of this corrupted frame. Consequently the switch may apply Partial Packet Discard (PPD) [2] to discard the remainder of the frame.

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 have been proposed, but usually EPD uses a fixed, global threshold. Romanow and Floyd [2] reported a positive impact of EPD on TCP performance, however, measurement campaigns covering a large range of parameters have never been carried out.

## 3 Configuration and Measurements for the TCP over UBR Experiments

### *3.1 Testbed Configuration*

The ATM LAN configuration, depicted in Fig. 1, is based on a FORE ASX-200 ATM switch. The ASX-200 implements delay priorities for three service classes (CBR, VBR and ABR-UBR), programmable buffer size, a flavour of per-VC queuing, VP shaping and Early Packet Discard with a programmable EPD threshold. The EPD mechanism implemented in the ASX-200 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 PDU arrives, the incoming AAL5 frame will be fully discarded except for its last cell containing the EOM field which allows to delimit different frames. In the case a cell is lost in the middle of an AAL5 PDU, the remaining cells of that PDU are discarded, i.e. subject to PPD. Four Solaris 2.6 workstations are configured as senders and two HP-UX 9.05 together with two HP-UX 10.20 workstations are configured as receivers. All the senders behave as greedy sources sending bulk TCP traffic to the receivers simultaneously during 180 seconds, a large value compared to [2] and [4] which typically simulate the system for 10 to 20 seconds. Only ACKs flow in the reverse direction. These four TCP connections share a bottleneck of 37.44 Mbit/s minus the overhead of ATM, AAL5, RFC1483 [17] and TCP/IP. The bottleneck rate is limited by means of VP shaping. The protocol stack involved in the experiments is also shown in Fig. 1. In all measurements presented here, the user data buffer size is set to 16 kB and the TCP maximum window size to 52224 byte.

### *3.2 Measurements and Performance Metrics*

While measuring the performance of TCP over UBR dependent on the switch buffer size and the MTU, Early Packet Discard has been disabled or enabled with several buffer thresholds and TCP's Fast Retransmit and Recovery threshold varied. Furthermore, the influence of the TCP implementation on TCP throughput efficiency has been studied.

The benchmark tool *ttcp* generates and receives TCP traffic at the application layer and measures the memory-to-memory throughput, also called goodput, averaging out 10 runs with identical parameters. Throughout this paper, TCP performance is assessed in terms of throughput efficiency, defined as the ratio between aggregate TCP goodput and theoretical achievable goodput under loss-less conditions.

Results for EPD in this paper will often be presented as a function of the EPD normalised excess buffer capacity, *b*, i.e. the buffer capacity in excess of the EPD threshold normalised to the MTU size. The EPD threshold *EPDth* and normalised buffer excess capacity b are related by: *EPDth = Switch Buffer - b . MTU**, where MTU* is an upper bound to the MTU size (in cells). Note that if the EPD threshold equals the switch buffer size, EPD is equivalent to PPD.

The Cell Loss Ratio (CLR) is measured at the ATM

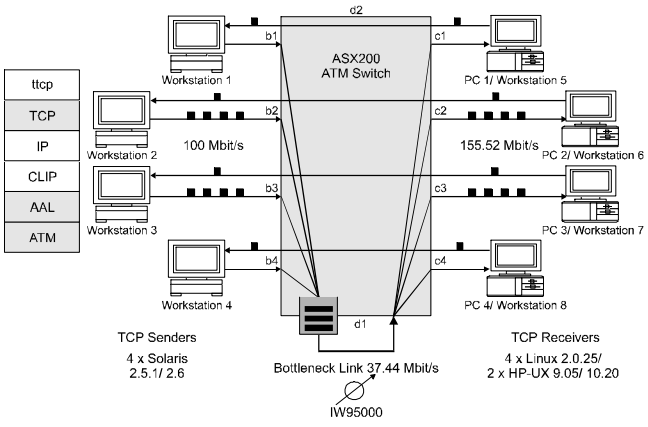


Fig. 1: Configuration for the TCP over UBR experiments

switch as the ratio between the number of incoming and outgoing ATM cells and the AAL5 Frame Error Ratio (FER) is measured at the end-systems as the ratio between the number of correctly received frames and the total amount of received frames.

Other metrics like Dead Cell Ratio and Idle Cell Ratio will also be used throughout this paper. The Dead Cell Ratio corresponds to the ratio between the cells belonging to corrupted AAL5 frames which are transported to the destination and the total amount of cells, whereas the Idle Cell Ratio quantifies the link idle time.

In [4], a metric for the assessment of fairness among different TCP sources has been proposed. The *fairness* is quantified through the fairness index defined as:

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

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

## 4 Experimental Results

This chapter deals with the most important results of two separate experimental campaigns with different TCP implementations. In both scenarios, the performance of TCP over UBR plus EPD is measured dependent on a wide range of parameters in all layers involved. Variable MTU, different switch buffers, duplicate ACK threshold, and timer granularity values have been configured. By means of using different TCP implementation, it can be demonstrated that especially TCP's reactivity to segment loss is critical for the performance of TCP over UBR with EPD.

### *4.1 Performance of Default TCP over UBR with Partial and Early Packet Discard*

In this section, the performance of TCP over UBR is measured as a function of the switch buffer for plain UBR, UBR with PPD and EPD with different EPD thresholds. Fig. 2 shows TCP efficiency as a function of the switch buffer for an MTU of 4352 byte (for EPD, the normalised buffer excess capacity $b$ is equal to 1). PPD and EPD significantly improve the efficiency for small buffers and as the buffer size is increased beyond 1088 cells (approximately equal to one TCP window size), the three curves saturate. With a buffer of 1088 cells, TCP achieves approximately 87% efficiency with plain UBR, 91.5% with PPD and close to 93% with EPD. With an MTU of 4352 byte and EPD with $b$ equal to 1, TCP does not encounter any fairness problems (not shown here). The fairness index is larger than 0.95 with all buffers also used in Fig. 2. With PPD, fairness is not as good as for EPD with $b$ equal to 1 but usually the fairness index is still larger than 0.95 apart from a few exceptions between 0.75 and 0.95. On the other hand, fairness is not satisfactory for plain UBR with buffers smaller than 1088 cells but at least good with larger values (index larger than 0.9). Note that the efficiency observed with a buffer of 2176 cells corresponds to 99.5% of the efficiency measured with a buffer that ensures a loss-free operation of TCP, which means that close to full efficiency is already achieved with a switch buffer of one window size (since the efficiency does not increase very much between 1088 and 2176 cells).

For an MTU of 9180 byte (results not shown here but available in [18]), the same observations remain generally valid. In terms of efficiency, EPD with $b$ equal to 1 performs better than PPD which excels plain UBR as expected. On the other hand, the fairness is not satisfactory up to a buffer of 1360 cells. Buffers larger

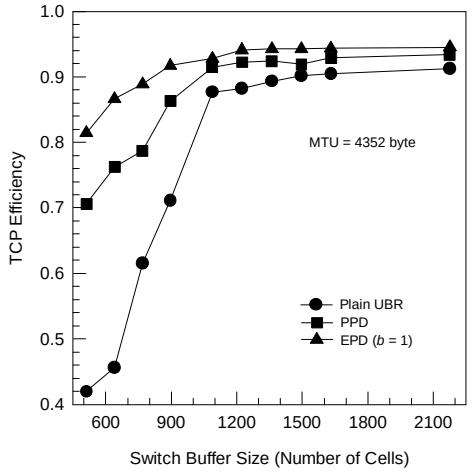


Fig. 2: Measured TCP efficiency for plain UBR, PPD and EPD, $b$ = 1.

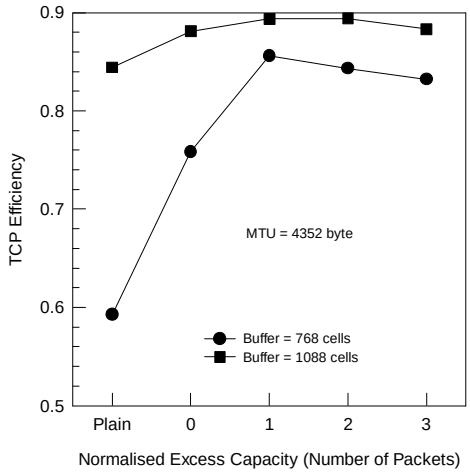


Fig. 3: Measured TCP Efficiency for plain UBR, PPD and EPD with different thresholds, MTU 4352 byte.

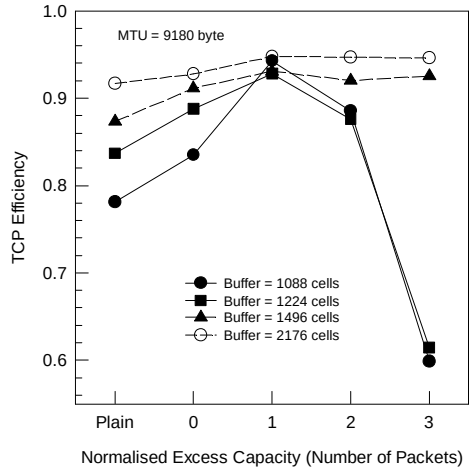


Fig. 4: Measured TCP Efficiency for plain UBR, PPD and EPD with different thresholds, MTU 9180 byte.

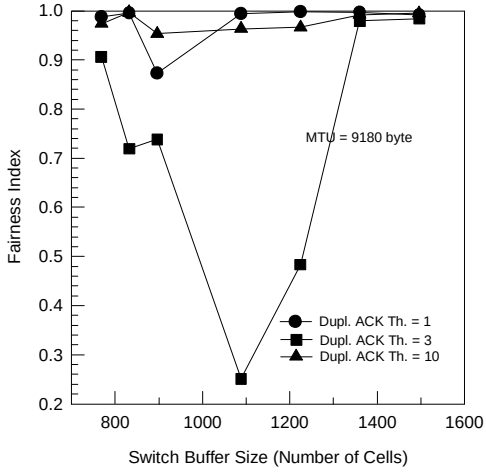


Fig. 6: Fairness as a function of the switch buffer for different values of the duplicate ACK threshold.

than 1360 cells provide good fairness and efficiency.

Fig. 3 and Fig. 4 compare the efficiency achieved by TCP for plain UBR, PPD and EPD with three different EPD thresholds and different MTU. Best efficiency is observed for EPD with *b* equal to 1. For a larger *b*, efficiency decreases with small switch buffers because an increased excess capacity is obviously rather seen as a reduction of the effective buffer which results in increasing segment loss. This is especially critical for the large MTU of 9180 byte since the excess capacity is set proportional to the MTU. For large buffers, an increase of the normalised excess buffer capacity does not decrease efficiency because it already saturates with a smaller effective buffer.

Some authors [3], [4] explicitly state that TCP over UBR actually requires buffers equal to the sum of the maximum TCP window size to achieve zero loss and consequently good performance. Our results show that zero loss is not necessary to achieve good bottleneck utilisation. For example, the frame loss ratio is between 6.7% and 4.8% with buffers between 1088 and 2176 cells in Fig. 2 for EPD with *b* equal to 1. Almost maximum efficiency and fairness can already be observed for a much smaller buffer size (than the sum of the TCP window size of the involved connections) when EPD is used. A switch buffer size slightly larger than one TCP window size already ensures good performance. However, TCP's reactivity to segment loss, i.e. its time-out and retransmission strategy is a critical element in the considered configuration. With small buffers, i.e. smaller than one TCP window, when losses occur frequently, FRR alone is not capable of avoiding retransmission time-outs and consequently PPD and EPD can be efficient because they concentrate cell loss to fewer frames. For a buffer larger than one TCP window size, FRR turns out to be very effective in avoiding time-outs and the difference between plain UBR, PPD and EPD is only marginal.

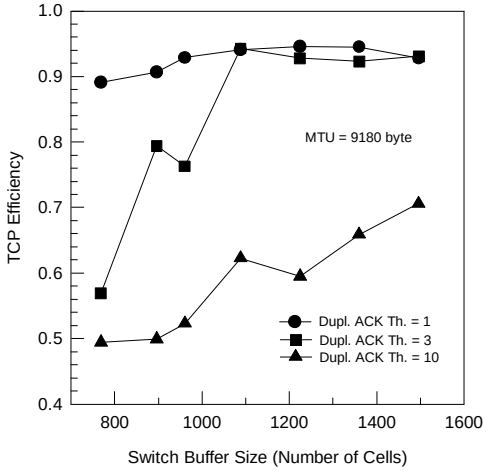


Fig. 5: TCP Efficiency as a function of the Switch buffer size for different value of the duplicate ACK threshold.

### *4.2 Effect of TCP's Fast Retransmit and Recovery Threshold*

In order to evaluate the impact of FRR further, the number of duplicate ACKs which have to be received (this parameter is called duplicate ACK threshold) before Fast Retransmit is triggered has been varied. Fig. 5 and Fig. 6 present efficiency and fairness respectively as a function of the switch buffer size for different values of the duplicate ACK threshold and an MTU of 9180 byte. Note that for a socket buffer of 52'224 byte and an MTU of 9180 byte, the maximum TCP window size corresponds to less than six segments which means that a duplicate ACK threshold of 10 practically disables FRR (retransmission occur in this case only after time-outs).

Fig. 5 demonstrates that FRR has a significant impact on efficiency. With a duplicate ACK threshold of 1, a single duplicate ACK triggers FRR thus avoiding time-outs. Thanks to the sequence integrity guaranteed by ATM, the number of false retransmissions remains limited. Conversely, a threshold of three duplicate ACKs is sometimes out of reach since a TCP sender cannot always send enough segments into the network due to the sending window constraint (especially with Delayed Acknowledgement strategies) Of course, this happens frequently with small buffers when the Congestion Window tends to be smaller. Note that results for smaller MTUs are qualitatively similar.

Fig. 6 shows the fairness index as a function of the switch buffer size for the different duplicate ACK thresholds. Fairness is high for a threshold of 1 whereas it is potentially very low with a threshold of 3. If time-out triggered retransmissions (threshold of 10) become dominant, fairness is generally good at the expense of efficiency. A more detailed analysis of the impact of FRR on the fairness is given in [18].

As a last comment it should be noted that FRR, when effective, allows to mitigate the trade-off between efficiency and fairness as observed when retransmission are handled with time-outs as can be seen in Fig. 6 and as is reported in literature [3] (results for small MTUs support this [18]).

### *4.3 Influence of the TCP Implementation*

EPD is expected to clean the frame loss process in a way that one or more connections survive a congestion period without multiple loss and subsequent idling. As will be shown below, unusual, sometimes faulty implementation details can prevent TCP from recovering quickly from loss. Though this does not necessarily result in worse performance, it further affects the potential benefit of EPD.

In the original configuration depicted in Fig. 1, Solaris 2.5.1 instead of Solaris 2.6 workstations act as senders and Linux 2.0.25 PCs replace the HP-UX workstations as receivers. The systems involved are perfectly homogeneous, however, this special combination holds some shortcomings.

First, a Linux receiver delays the acknowledgement based on an operating system timer and thus acknowledges a bunch of segments when this timer expires. Even if more than two MSS byte of in-sequence data have arrived, this timer is still used to asynchronously send the acknowledgement, which yields a delay of approximately 4 ms in the configuration under consideration. Furthermore, new acknowledgements are merged with the pending acknowledgements. Since the number of acknowledgements is reduced significantly in this way, such a „stretch ack“ policy [14] slows down the growth of the congestion window of TCP and, in addition, makes it difficult to trigger Fast Retransmit at the sender.

Second, Solaris 2.5.1 does not handle time-out triggered retransmissions properly. When several segments are lost, Solaris 2.5.1 doubles the retransmission time-out every now and then even if an outstanding segment has just been acknowledged. Besides, incoming acknowledgements are often processed incorrectly which results in a long phase of time-out triggered, partly unnecessary retransmissions. As Fast Retransmit is practically disabled due to the Linux acknowledgement mechanism, this faulty implementation of TCP changes the system behaviour drastically.

Fig. 7 indicates that EPD hardly improves efficiency with these TCP implementations. Since the Linux receivers usually cannot generate sufficient duplicate acknowledgements to trigger Fast Retransmit and Recovery, TCP is not able to recover quickly from multiple losses and ends up with a Solaris 2.5.1 typical long phase of time-out triggered retransmissions. As illustrated in Fig. 8, a lower timer granularity of 20 ms

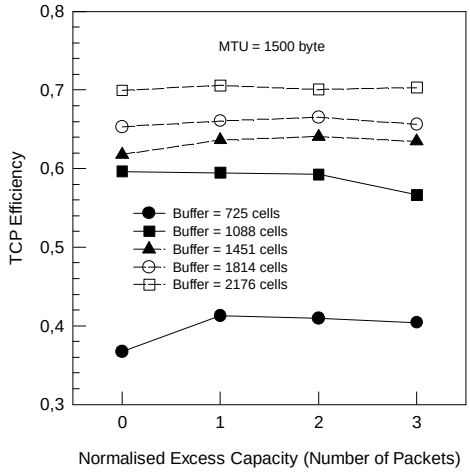


Fig. 7: Solaris-Linux TCP goodput as a function of the normalised excess buffer capacity, MTU 1500 byte.

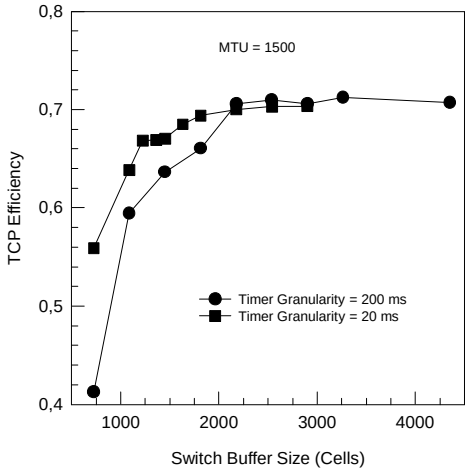


Fig. 8: Solaris-Linux TCP efficiency as a function of the buffer size for different timer granularity values, EPD *b*=1.

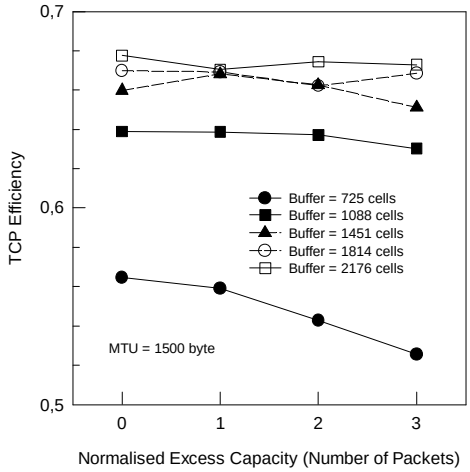


Fig. 9: Solaris-Linux TCP goodput as a function of the normalised excess buffer capacity *b* for MTU = 1500 byte for a timer granularity of 20 ms.

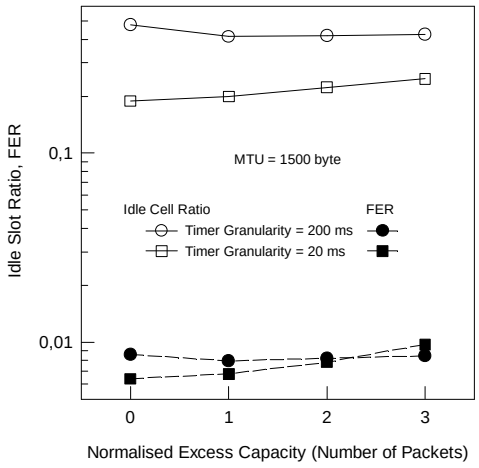


Fig. 10: Solaris-Linux Idle Cell Ratio and FER as a function of the excess capacity *b* for a switch buffer of 725 cells and different timer granularity values.

may result in a noticeable efficiency gain with small buffers. On the other hand, EPD still does not improve anything at all, see Fig. 9. In fact, the idle cell ratio at the bottleneck switch port shown in Fig. 10 is orders of magnitude higher than the FER at the receivers (which can be regarded as a sufficiently safe upper bound of the dead cell ratio). Hence, in this situation, larger performance improvements can be achieved by improving TCP's reactivity to losses with a lower timer granularity rather than by reducing the proportion of dead cells with EPD.

## 5 Relation with Simulation Studies

The experiments described in the previous sections show that TCP over UBR performance may benefit from EPD, however, it is very sensitive to many parameters such as (a

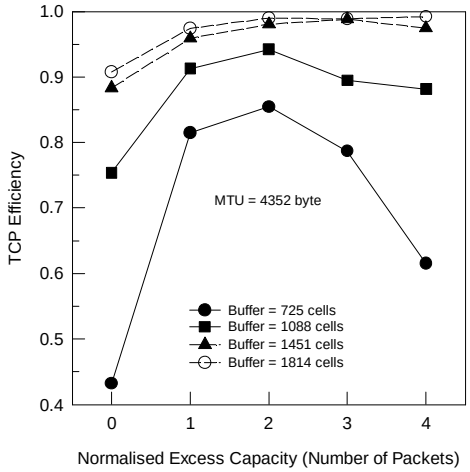


Fig. 11: Simulated TCP efficiency as a function of the normalised excess capacity.

possibly heterogeneous) packet size, the delayed (stretch) acknowledgement strategy, the reaction and reactivity of TCP to multiple loss and probably other effects not covered in this paper. Of course, any experimental configuration does not scale well and suffers from a limited flexibility and controllability as opposed to simulations. For this reason, this chapter relates experimental to simulation results and discusses the deviation of this study from previous simulation work reported in [2].

### *5.1 Simulated Performance of Default TCP over UBR*

The simulation tool used, models the experimental configuration from Fig. 1. In particular, the TCP version includes the specifications from [15] and [12]. Throughout this section, the MTU is 4352 byte, the duplicate ACK threshold for Fast Retransmit/Recovery is set to 3, and TCP works with a timer granularity of 200 ms.

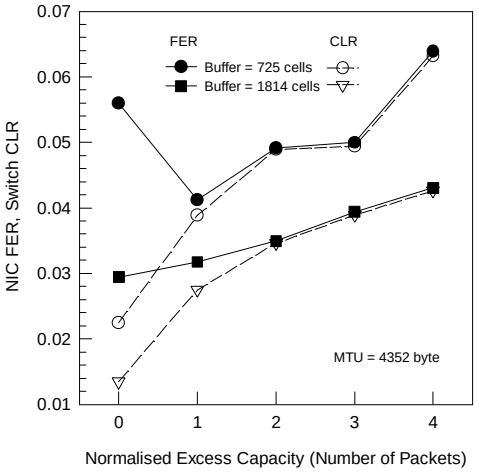


Fig. 12: Simulated Frame Error Ratio at the receivers and CLR at the switch as a function of *b*.

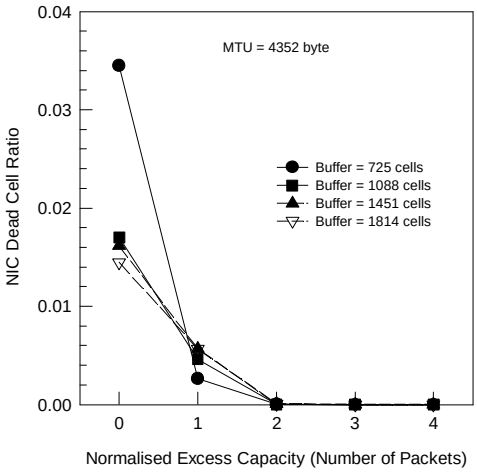


Fig. 13: Simulated Dead Cell Ratio at the receivers as a function of the normalised excess capacity.

Fig. 11 clearly shows that the chosen parameter subset is appropriate to demonstrate the potential benefit of EPD similar to section 4.1 and 4.2 as well as [2]. With small buffers, which is the most interesting case, EPD achieves the highest performance improvement with a normalised excess capacity of 2. With this setting, the CLR at the switch approaches the FER at the receivers, as can be seen in Fig. 12, i.e. the dead cell ratio approaches 0. A further increase of the normalised excess capacity beyond 2 leads to a performance decrease with small buffers. In this case, the FER gets higher again because EPD with a normalised capacity of 2 fulfils its assigned role of avoiding dead cells almost perfectly (Fig. 13) and a further increase of the excess capacity is rather seen as a reduction of the effective buffer size. However, with the optimum threshold settings, EPD really cleans the packet loss process compared to PPD, as has been shown by a more detailed trace-driven analysis in [19]. The likelihood increases that at least one connection survives loss periods without multiple loss and thus can recover quickly. As a result,

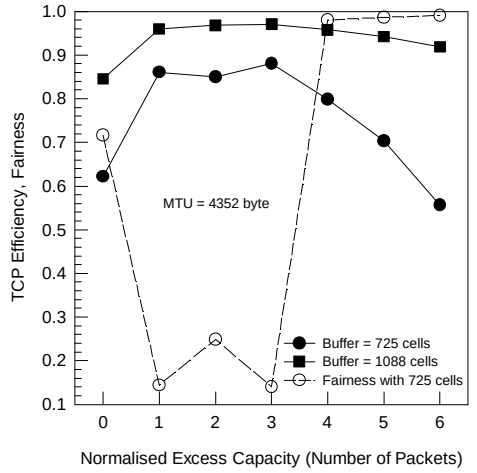


Fig. 14: Simulated TCP efficiency and fairness with 16 sources as a function of the normalised excess capacity

usually at least one connection is active. Without EPD, often all TCP connections enter a time-out triggered retransmission phase when congestion occurs and thus show a tendency to idle synchronously.

### *5.2 Increasing the Number of Connections*

With small buffers, 725 and 1088 cells, maximum performance is achieved for an excess capacity of 2. This is different from [2], where performance of EPD increases monotonously. However, the respective configurations deviate from each other since the parameter sets used throughout this paper were the result of often very practical considerations as well as careful end system performance measurements. First, the experimental configuration introduced in Fig. 1 comprises fewer sources than normally used in simulation studies. By means of shaping, the bottleneck capacity at the bottleneck link is reduced to 37.44 Mbit/s in order to ensure that each individual TCP connection is able to exploit the bandwidth with any arbitrary parameter combination used in this paper. The ASX-200 switch schedules the UBR per-VC queues with a round robin discipline as opposed to the FIFO strategy simulated in [2]. In addition, timer granularity is either set to 20 ms or 200 ms (default) and not to 0.1 ms as in [2]. In a local environment, a large timer granularity causes TCP to over-estimate the round-trip time. This results in a too large retransmission time-out value and waste of bandwidth.

At first glance, the number of connections might be the most important difference. This section therefore investigates the performance of TCP over UBR plus EPD in simulations with 16 sources. The main results are presented in Fig. 14. With a buffer of only 1088 cells, the efficiency of TCP is quite robust against a change of the excess capacity but still non-monotonous performance behaviour can be observed. Whereas also perfect fairness is achieved with that buffer size (not shown in Fig. 14), the system with a switch buffer size of 725 cells gets rather unfair. If both fairness and efficiency are taken as performance criteria, PPD or an excess buffer capacity of 4 should be favoured. However, the complete different behaviour for the respective buffer size renders any profound dimensioning impossible.

### *5.3 Discussion of the Idling TCP Configuration*

In section 4.3, experimental results with a TCP configuration composed of Solaris 2.5.1 senders and Linux 2.0.25 have been summarised. In this scenario, TCP over UBR only benefits little from EPD. This section tries to validate those result by approximating the known unusual TCP implementation details. Among others, the simulator was subject to the following modifications:

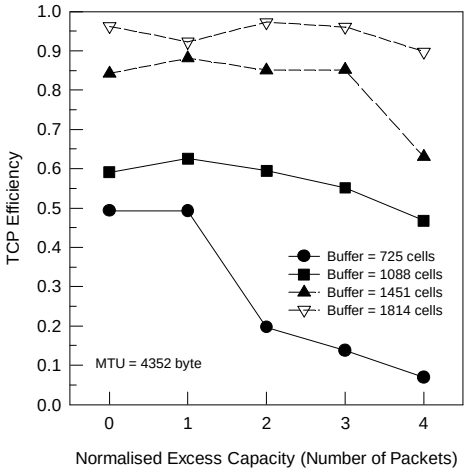


Fig. 15: Simulated TCP efficiency as a function of the normalised excess capacity (Modified TCP).

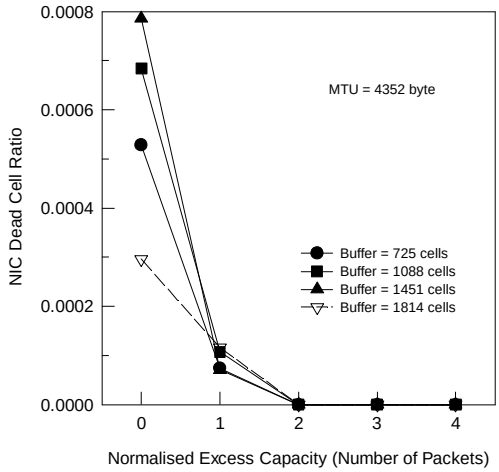


Fig. 16: Simulated Dead Cell Ratio at the receivers as a function of *b* (Modified TCP*)*.

- Exponential backoff is only reset when all retransmitted segments have been acknowledged in order to emulate the extensive idling after multiple loss.
- The stretch acknowledgement delay is set to 4 ms.
- While waiting for the return of an acknowledgement for a segment which was retransmitted by Fast Retransmit, the congestion window must not be increased. This is a conservative approximation of a Solaris 2.5.1 implementation detail reported in [14].
- The duplicate acknowledgement counter is not reset when a retransmission time-out occurs [14].

As discussed in section 4.3, the number of acknowledgements decreases dramatically thus practically disabling Fast Retransmit/Recovery. At the same time, the amount of data acknowledged at once increases which will lead to more bursty packet arrivals. In fact, these changes make the TCP over UBR plus EPD system ineffective (Fig. 15). Even with an excess capacity of 0 (PPD), only a few dead cells can be observed in Fig. 16 (compared with Fig. 13). As pointed out above, EPD is beneficial only in conjunction with Fast Retransmit because it concentrates cell loss on fewer segments thus enabling at least some TCP connections to avoid retransmission time-outs. Since this does not work here, EPD only reduces the effective buffer size, and performance gets even worse.

## 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.

An efficient co-operation of EPD with TCP's retransmission strategy has been found to be the key requirement for good TCP over UBR performance with relatively small buffers, compared to the TCP window size. In the local network configuration, the Fast Retransmit and Recovery algorithms work sufficiently well. EPD cleans the loss episode by concentrating cell loss to a lower number of segments and, in this way, helps some TCP connections to avoid retransmission time-outs.

However, in experiments with different TCP implementations, which were characterised by a dominance of time-out triggered over FRR retransmissions, EPD has little impact. In this case, only a reduction of the timer granularity makes TCP more reactive to segment loss and thus reduces idle time. The ratio of cells belonging to corrupted frames but still arriving at the receiver (dead cell ratio) alone is only an indicator for good EPD performance if it is compared to the values measured with plain UBR and PPD.

Moreover, good performance in terms of both efficiency and fairness for TCP over UBR with EPD can be achieved with a switch buffer slightly larger than one TCP window size. Zero loss is not necessary to obtain good performance but, as stated above, losses must be reduced such that many lost segments can be retransmitted efficiently with Fast Retransmit/Recovery. EPD helps to meet this requirement with a smaller buffer size, compared to plain UBR and PPD, provided that the threshold is set appropriate. The latter condition can be hard to meet in a more heterogeneous scenario.

In this work, both experiments and simulations have been carried out in order to validate the results but also to test hypotheses about the behaviour of the real systems. At the end, a qualitative correspondence between simulation and measurements was achieved.

Selective Acknowledgement [16] may also enable TCP to bypass retransmission time-outs and to recover quickly from congestion with multiple loss. The investigation of this option in the context of UBR and EPD is a worthwhile subject for future experimental work.

**Acknowledgements**

The achievements made within the project EXPERT and the comprehensive development work undertaken was only possible with the conscientious co-operation of all partners especially of Work Package 4.2, together with the support of the Project Officer and funding from the Commission of the European Union, the Swiss Bundesamt für Bildung und Wissenschaft and other national bodies. The contribution of all these players is therefore hereby gratefully acknowledged.

**References**

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

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

[3] S. Kalyanaraman, R. Jain et al, "Performance and Buffering Requirements of Internet Protocols over ATM ABR and UBR Services", *IEEE Communications Magazine*, January 1997.

[4] R. Goyal, R Jain et al, "Improving the Performance of TCP over the ATM-UBR Service", *proceeding of ICC'97*, Montreal, June 1997

[5] H. Li, K.-Y. Siiu et al, "TCP Performance over ABR and UBR services in ATM", *Proceeding of IPCCC'96*, Phoenix, March 1996.

[6] Lakshman T. V., Neidhardt A., Ott T.J.," The Drop from Front strategy in TCP and in TCP over ATM", *proceedings of IEEE Infocom*, 1996.

[7] 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.

[8] 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.

[9] Bonaventure O., Klovning E., Danthine A., "Behaviour of TCP in the European Pilot," *Computer Communications, Special Issue on algorithms for ATM Networks*, 1996.

[10] Bonaventure O., Klovning E., Danthine A., "Is VBR a Solution for an ATM LAN", *in 5th IFIP Workshop on Protocols for High-Speed Networks*, 1996.

[11] R. Morris and H. T. Kung, "Impact of ATM Switching and Flow Control on TCP Performance: Measurements on an Experimental Switch", *proceedings of Globecom'95*, 1995.

[12] W. R. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms", *IETF, Network Working Group*, RFC 2001, january 1997

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

[14] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics", *Ph.D. Thesis* UCB/CSD-97-945, LBNL-40319, April 1997

[15] B. Braden, ed., "Requirements for Internet Hosts -- Communication Layers," RFC 1122, Oct. 1989

[16] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options", *IETF Network Working Group*, RFC 2018, October 1996

[17] Heinanen J., "Multiprotocol Encapsulation over ATM Adaptation Layer 5", *IETF, Network Working Group*, RFC 1483, July 1993

[18] L. Jaussi, "Report on TCP over UBR Experiments on an ATM LAN", *EPFL/TCOM Internal Research Report*, No 1998/X, Lausanne, April 1998.

[19] M. Lorang, "Discussion of TCP over UBR Results from Measurements, Simulations and Analytical Studies", *ACTS 094 Contribution AC094_UST 42_008.02_CD_CC*, Stuttgart, March 98.