5

Is VBR a solution for an ATM LAN ?*

Olivier Bonaventure[1], Espen Klovning[2], André Danthine[1]
[1] Research Unit in Networking, Institut d'Electricité Montefiore, B-28, Université de Liège, B-4000 Liège, Belgium
[2] Telenor Research and Development, P.O. Box 83, N-2007 Kjeller, Norway
Email : bonavent@montefiore.ulg.ac.be, Espen.Klovning@fou.telenor.no, andre.danthine@ulg.ac.be

Abstract

We present and discuss detailed measurements showing how TCP behaves in an ATM LAN where a Variable Bit Rate (VBR) traffic contract is enforced by a Usage Parameter Control (UPC) mechanism. These measurements show that TCP has a lot of difficulties to adapt its behaviour to such a traffic contract. In our environment, TCP only achieved a 10% utilisation of a 10 Mbps VBR VC. We then discuss and evaluate possible solutions to improve this low utilisation of the reserved bandwidth.

1 INTRODUCTION

Asynchronous Transfer Mode (ATM), selected in 1988 by the ITU as the basis for the B-ISDN has quickly raised a huge interest within the datacommunications community, and it soon appeared as a promising technology for the LAN environment. Many consider ATM's inherent scalability, as well as the provision of QoS guarantees, as its main benefits compared with other high-speed LAN technologies.

However, to quickly benefit from ATM with today's applications and networking protocols, it was necessary to achieve a seamless integration of TCP/IP and other widely used networking protocols within the connection-oriented ATM environment. The connection-oriented nature of ATM does not fit well with most of the current networking protocols which have been developed with a connectionless environment in mind. Both the Internet Engineering Task Force (IETF) and the ATM Forum have proposed solutions to integrate TCP/IP (IETF) as well as other networking protocols (ATM Forum) within the ATM environment. These two solutions, respectively Classical IP over ATM and LAN Emulation, define mainly how the address resolution must be done from an IP or MAC layer address to an ATM address, and how and when ATM Virtual Circuits (VCs) are established between communicating entities.

However, this address resolution problem, while being an important, bootstrap-like, problem, is not the only one that must be solved for a seamless integration of TCP/IP in the ATM environment. The pending issues include notably the suitability of TCP's congestion

* This work was partially supported by the European Commission within the ACTS AC051 OKAPI project.

W. Dabbous et al. (eds.), *Protocols for High-Speed Networks V*

control scheme for the cell-based ATM environment, how to perfom routing efficiently in a mixed IP/ATM network, but also how to benefit from the QoS guarantees provided by ATM.

In this paper, we first review the three types of traffic contracts supported by today's ATM standards. Then, we present and discuss detailed measurements showing how TCP is able to adapt its behaviour to a Variable Bit Rate (VBR) traffic contract, and look at some possible improvements.

2 ATM TRAFFIC CONTRACTS

When ATM was selected by the ITU, it was assumed that the users of the B-ISDN would be able to define precisely their bandwidth requirements through a traffic contract or traffic descriptor. The network would use this traffic contract to perform operations such as routing, Connection Admission Control (CAC) and Usage Parameter Control (UPC). There were two possible approaches for the traffic contract: a statistical and an operational one [DeP93]. The first approach uses stochastic traffic parameters such as an average cell rate or an average burst size. Such parameters can be used to perform CAC, but it would have been difficult to use them for UPC, because long observation times are usually necessary to verify the compliance to a statistical traffic contract while the UPC must react quickly to traffic contract violations to protect the network from misbehaving users. The operational approach defines the traffic contract with one algorithm and a few associated parameters. This approach is suitable for both CAC and UPC as it allows an unambiguous discrimination between the conforming and the non-conforming cells. The algorithm selected by the ITU is known as the Generic Cell Rate Algorithm(GCRA) [I371].

The GCRA uses two parameters : the Cell period T and the allowed Cell Delay Variation (CDV) τ . T is the reciprocal of a Cell Rate and, intuitively, τ corresponds to a maximum allowed short-term deviation from this Cell Rate.

The current UNI signalling specification [UNI30] uses this GCRA to define three types of traffic contracts :

- Constant Bit Rate (CBR)
- Variable Bit Rate (VBR)
- Best Effort (later renamed Unspecified Bit Rate [Sat96])

The CBR traffic contract is suitable for applications which transmit at a constant bit rate during the whole connection duration, e.g. leased line emulation or uncompressed voice or video over ATM [Sat96], but nothing precludes its use by data applications. This traffic contract is defined by GCRA [T_{PCR}, τ], where T_{PCR} is the reciprocal of the Peak Cell Rate reserved for the CBR VC, and τ the allowed CDV. The main characteristic of a CBR VC is that the ATM network will reserve resources according to the Peak Cell Rate (PCR) of the corresponding traffic contract and will guarantee a low (almost 0) cell loss ratio for this VC. However, this traffic contract is seldom used for data applications, as these applications are usually very bursty.

The second traffic contract is the Variable Bit Rate traffic contract. This VBR traffic contract can be used for compressed audio and compressed video applications, but also for response time critical transaction processing (e.g. airline reservation, banking transactions, ...) [Sat96]. It is defined by two GCRAs. The first one, GCRA [T_{PCR}, τ_{PCR}], specifies the peak cell rate for the VBR VC. The second one, GCRA [T_{SCR}, τ_{SCR}], specifies the Sustainable Cell Rate (SCR) and the allowed CDV for this SCR. Intuitively, the SCR is the rate at which the source may transmit during an infinite time. The allowed CDV for the SCR, τ_{SCR}, is not specified directly by the end-system at connection establishment time. Instead, the end-system specifies the Maximum Burst Size (MBS). Intuitively, the MBS corresponds to the largest burst of ATM

cells that can be sent at the Peak Cell Rate. τ_{SCR} can be calculated [UNI30] from the MBS as follows :

$$\tau_{SCR} = (MBS - 1) \times (T_{SCR} - T_{PCR}) \quad (1)$$

The main characteristic of a VBR VC is that the network will reserve resources for this VC according to the specified PCR, SCR and MBS so that if the end-system respects the traffic contract, the cell loss ratio for the VC will be low. The traffic contract will be enforced by a UPC using the two GCRAs operating in parallel on the cell flow. This UPC will either discard or tag (i.e. change the CLP bit from 0 to 1) the non-conforming cells depending on which one of the 3 VBR traffic contracts (table 1) supported by ATM Forum UNI 3.0 is used for the VC. These traffic contracts differ on the treatment of the low priority cells (CLP=1) and also on whether the non-conforming cells should be tagged or discarded by the UPC.

Table 1 : VBR Traffic contracts supported by ATM Forum UNI 3.0 specification

	T_{PCR}	τ_{PCR}	T_{SCR}	MBS	Tagging
A	specified for CLP=0+1	specified for CLP=0+1	specified for CLP=0	specified for CLP=0	not requested
B	specified for CLP=0+1	specified for CLP=0+1	specified for CLP=0	specified for CLP=0	requested
C	specified for CLP=0+1	specified for CLP=0+1	specified for CLP=0+1	specified for CLP=0+1	not applicable

The best-effort contract is the last traffic contract introduced in ATM Forum UNI 3.0. It is defined by GCRA [T_{PCR}, τ]. However, in contrast with a CBR VC, the network does not reserve any resources for a best-effort VC. Thus, there are no QoS commitments for a best-effort VC.

The best-effort traffic contract is used by most ATM LANs today. Unfortunately, by using this traffic contract, these ATM LANs do not benefit from the QoS guarantees that can be offered by the ATM network. These QoS guarantees would surely be useful in environments where mission critical applications (e.g. airline reservation, banking transactions) are used and thus should get a guaranteed minimum amount of network resources, even when the network is heavily loaded. With ATM Forum UNI 3.0[1], the VBR traffic contract appears to be the best solution to support such bursty applications as it allows reservation of resources, as well as a certain amount of burstiness. With the third (C) VBR traffic contracts (table 1), the applications will always get the exact amount of resources specified in the traffic contract. With the first (A) VBR traffic contract, the applications will get the resources specified in the traffic contract when transmitting high priority (CLP=0) cells, but may get additional resources (on a best-effort basis) when they transmit low priority (CLP=1) cells. With the second (B) VBR traffic contract, the applications may get more than the amount of resources specified in the contract, either by transmitting low priority cells, or by sending high-priority cells in excess of the traffic contract and assuming that these cells (which will be tagged by the UPC) will be successfully transmitted if the network is lightly loaded. However, in a heavily loaded network, the VBR VCs will get exactly the resources specified in their traffic contract and enforced by the UPC. To benefit from the VBR VC, the higher layer protocols and the applications should be able to use this specified minimum amount of resources efficiently. In the remaining parts of this paper, we study how TCP is able to use such a minimum amount of resources.

---

[1] Both the ITU and the ATM Forum are working on new traffic contracts (Available Bit Rate and ATM Block Transfer), but it will take some time before they are supported by products and widely deployed. Many ATM networks can be expected to continue to use the UNI 3.0/3.1 specifications for some time.

3 MEASUREMENT ENVIRONMENT

Our measurement environment (figure 1) consisted of two Sparc 10 class workstations and one ASX-200BX ATM switch from Fore Systems.
The two workstations were running SunOS 4.1.3_U1 and were connected to the ASX-200BX with 155 Mbps links. The sending workstation used a Fore SBA-200 ATM adapter, while the receiving workstation used a more recent Fore SBA-200E ATM adapter. They both used release 4.0 of the ATM driver.

The TCP implementation in SunOS 4.1.3_U1 is a subset of TCP[2] Reno. It includes the slow-start and congestion avoidance algorithms, but lacks the window extensions and timestamp options defined in [JBZ92]. The implementation of the fast retransmit algorithm [Ste94] is incomplete in SunOS 4.1.3_U1 [BKD96], but this bug had been fixed for our measurements.

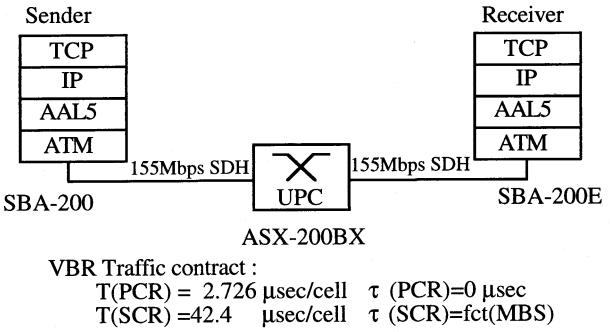

Figure 1 Measurement environment

The ASX-200BX is a 2.5 Gbps bus-based non-blocking output buffered ATM switch. It supports CBR, VBR and best-effort VCs, and contains dual-leaky buckets used for the UPC. For our measurements, we established one bi-directional PVC between the two workstations. There was no other ATM traffic through the switch during our measurements, and thus there was no congestion inside the switch. The traffic contract enforced by the UPC of the ASX-200BX was set to a Peak Cell Rate of 155 Mbps (i.e. one cell every 2.726 µsec which is equal to the physical line rate), no specified allowed Cell Delay Variation for the Peak Cell Rate, a Sustainable Cell Rate of 10 Mbps (i.e. one cell every 42.4 µsec), and we varied the Maximum Burst Size during the measurements.

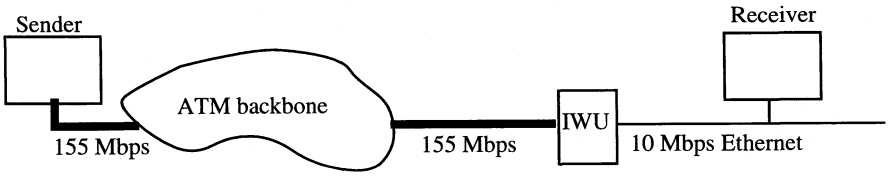

Figure 2 An equivalent target network

This measurement environment will also allow us to evaluate the performance in the network of figure 2 where the sender (e.g. a server) is connected to an interworking unit (IP router or

[2] The reader unfamiliar with TCP may find a good survey in [Ste94].

Ethernet Hub) with a 10 Mbps VBR VC through an ATM backbone and the receiver (e.g. a client) is connected to the interworking unit with a 10 Mbps Ethernet.

4 MEASUREMENTS WITH TCP

Our measurements were performed in the environment shown in figure 1 with the standard `ttcp` [SlM84] measurement tool. Each measurement was reproduced several times, and consisted in the memory-to-memory transfer of 4 MBytes of data in 16 KBytes long buffers. We enabled the TCP_DEBUG option for all the TCP connections to gather some packet traces. The receive and transmit window sizes were always set to 49152 bytes on both the sender and the receiver. By using the same transmit and receive windows on both workstations, we avoid the deadlock problems with SunOS 4.1.3_U1 discussed in [MoG95]. The round-trip-time measured by `ping` with small packets between the sending and the receiving workstation is about 1800 µsec.

During our measurements, we varied the MTU size used by IP over ATM. The selected MTU sizes correspond to the most commonly used MTU sizes for IP over ATM[3].We also varied the Maximum Burst Size enforced by the UPC.

We choose 1000 cells as the minimum value for the MBS as we have already discussed in a previous paper [BKD96] how TCP behaves when a UPC enforces a MBS corresponding to only a few packets.

We choose 4000 cells as the maximum value for MBS as our ASX-200BX (whose CAC algorithm is based on [Gue91]) could not establish (and associate QoS guarantees to) one ATM VC with a MBS much larger than this value.

The results of our first measurements (figure 3) were disappointing. We established one 10 Mbps VBR VC through the switch to benefit from QoS guarantees, and the throughput achieved by TCP with the larger MTU sizes is below 1 Mbps. The throughput achieved by TCP with an MTU size of 552 bytes is slightly higher, but still far from satisfactory. This is obviously a too small utilisation of the resources reserved for the 10 Mbps VBR VC.

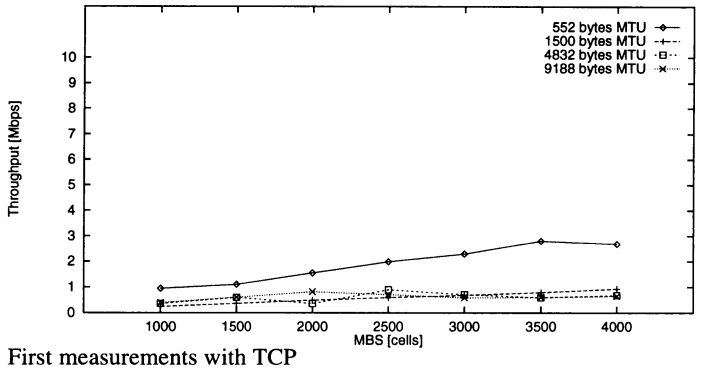

Figure 3 First measurements with TCP

A look at the packet traces gathered during the measurements revealed that TCP sent bursts of packets separated by an idle time of roughly one second. The TCP statistics reported by `netstat -s` showed a large number of retransmitted packets.

[3] 552 bytes corresponds to the default segment size used by TCP in the wide area, 1500 bytes corresponds to an Emulated Ethernet [LAN95], 4832 bytes to an Emulated TokenRing[LAN95], and 9188 bytes is the default MTU size for IP over ATM[Atk94]

The packet losses which caused this large number of retransmissions are due to non-conforming cells being discarded by the UPC. This can be explained by looking at the behaviour of our ATM adapter and at the GCRA. The large idle times, which caused most of the throughput drops, are due to the implementation of TCP's retransmission timer in SunOS and BSD-derived implementations and will be discussed later.

4.1 Explanation for the packet losses

To explain the packet losses, we can consider the packet train model shown in figure 4. This model is close to the behaviour of our ATM adapters. When a TCP packet is sent by the SunOS kernel, it is encapsulated inside one IP packet and passed on to the ATM driver. The driver enqueues this AAL-SDU and then wakes up the firmware of the ATM adapter. The AAL5 segmentation is performed inside the ATM adapter before the transmission of the corresponding ATM cells on the output link. In our environment, the transmission of one AAL-PDU by the ATM adapter took less time than the generation of the TCP/IP packet and the ATM driver processing. Thus, two consecutive AAL-PDUs were always separated by some idle time on the output link.

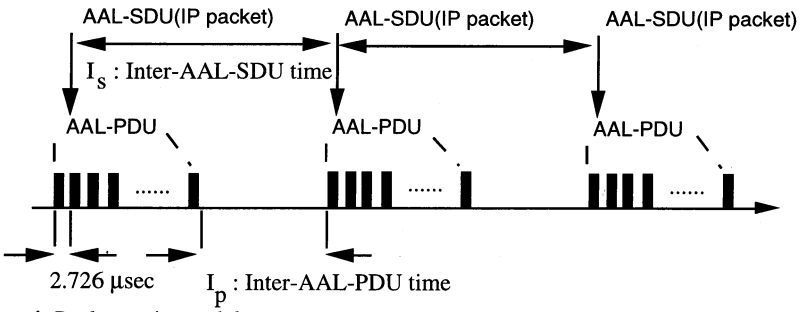

Figure 4 Packet-train model

The ATM traffic was always conformant to the PCR, as we set the PCR of the traffic contract to the PCR of the 155 Mbps link. Thus, all the cells discarded by the UPC were discarded due to a non-conformance with the SCR and MBS parts of the VBR traffic contract. From the specification of the GCRA, and the packet train model, it can be shown that a sequence of p AAL-PDUs separated by an inter-PDU time I_p, with each packet containing c cells, will be compliant with the traffic contract enforced by the GCRA if, considering that the UPC is in the idle state upon arrival of the first cells :

$$\forall\ k = 0 \ldots (p-1), \forall\ j = 0 \ldots (c-1) :$$

$$(k \times c + j) \times T_{SCR} \leq k \times (c \times 2.726 + I_p) + j \times 2.726 + \tau_{SCR} \qquad (2)$$

In an ATM LAN, TCP's behaviour is very close to this packet train model. We were not able to measure the Inter-AAL-PDU time directly, as this would have had required an ATM analyser which was not available where we performed our measurements. Instead we measured the Inter-AAL-SDU time with a modified (but older - release 2.2.9) Fore ATM driver. This modified driver [BKD96] is able to log in a kernel table a timestamp and some related information for each AAL-SDU sent or received by the driver. When we used TCP with this modified driver, the timestamps showed that there were some variations in the Inter-AAL-SDU times. These variations occurred, e.g. when TCP had to process one received acknowledgement or fetch some data from the user process' buffer. However, 600 µsec and

900 µsec appeared as reasonable estimations for the Inter-AAL-SDU times with respectively the 4832 and 9188 bytes MTU sizes . This corresponds to respectively 324 and 376 µsec for I_p, and to raw throughputs[4] of respectively 71 Mbps and 90 Mbps.

The packet traces have shown that TCP transmitted burst of packets separated by an idle time of roughly one second. Figure 5 compares the maximum burst length (in packets) predicted by the packet train model with the average burst size found during the measurements. The model seems to be a good approximation of the measured values. Some of the differences between the model and the measurements can be explained by looking at the packet traces, but we do not discuss these details due to space limitations.

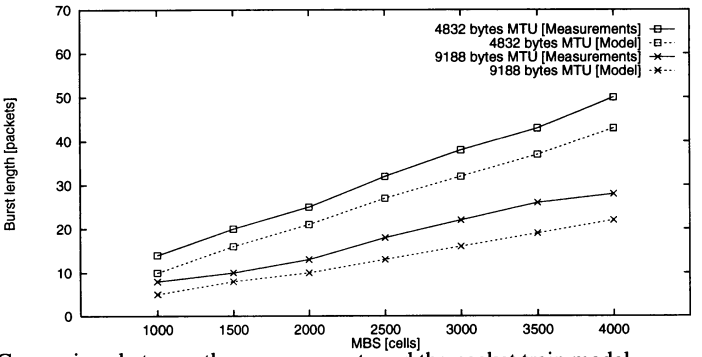

Figure 5 Comparison between the measurements and the packet train model

It should be noted that with a regular traffic such as the one shown in figure 4, once the UPC has accepted a burst of maximum length sent at the PCR, it will discard cells from all the subsequent AAL5-PDUs sent at the PCR until the link has been idle for a sufficient amount of time. The UPC will only accept a new entire AAL-PDU of c ATM cells sent at the PCR if the link has been idle for at least $c \times (T_{SCR} - T_{PCR})$ µsec (e.g. 7.6 msec with the 9188 bytes MTU in our environment). The Inter-AAL-PDU time measured with our ATM adapters is much smaller than this value.

With the smaller MTU sizes, the agreement between the model and the measurements was not so good, especially with the 552 bytes MTU. This is probably due to the fact that as the workstations are more heavily loaded with these small MTU sizes, the assumption that the Inter-AAL-PDU time is constant is not valid.

4.2 Explanation for the large idle times

The large idle times, which resulted in a low throughput, are due to the particularities of the implementation of TCP's retransmission timer in SunOS and BSD-derived implementations [WrS95]. In BSD-derived implementations, this retransmission timer is handled by the `tcp_slowtimo` routine which is called every 500 msec by the kernel [LMK89]. On each invocation, this routine checks for all the active TCP connections whether the retransmission timer has expired or not, and it invokes the `tcp_send` routine for each TCP connection whose retransmission timer has expired.

During our measurements, TCP transmitted a burst of packets. Cells from the last packets of this burst were discarded by the UPC, and TCP had to wait for the expiration of the

[4] By raw throughput, we mean the total throughput sent on the output link, i.e. including the user data, but also the ATM, AAL5, IP and TCP overheads.

retransmission timer to retransmit the lost packets. As the minimum value for TCP's retransmission timer is two 500 msec periods, the sender had to wait for approximately one second between two bursts of packets [BKD96].

A look at the packet traces revealed that sometimes the idle time jumped to two seconds instead of one. These larger idle times were caused by the loss of one retransmitted packet. When one retransmitted packet is lost, TCP's retransmission timer is doubled (exponential back-off), and this explains the larger idle time. The loss of one retransmitted packet was caused by the corruption of the corresponding AAL5-PDU by unreassembled cells from previously transmitted AAL5-PDUs. These losses could have been avoided with the optional AAL5 reassembly timeout (RAS) [I363], but this timeout was not supported by our ATM adapters [Cyp96].

We would like to point out that if we had used several TCP connections multiplexed into one VBR VC instead of a single one, the utilisation of the VC would not have improved significantly. Assuming that these TCP connections start at random times, each connection will transmit one burst of packets and then the loss of the last packets of the burst will force it to wait for the expiration of its retransmission timer. Unfortunately, the retransmission timer will expire at the same time for several connections, and all these connections will start to retransmit together. A few packets will be retransmitted for each connection, before they will all be forced to wait for the expiration of their common retransmission timer. Obviously, such a synchronisation among several connections will not lead to a better utilisation of the VBR VC.

5 POSSIBLE CHANGES TO THE TCP IMPLEMENTATION

The TCP implementation used in SunOS 4.1.3_U1 is a production quality implementation which has been highly optimised during the years [MKK94]. It is representative of a wide range of currently available BSD-derived implementations. However, it does not contain all the TCP modifications which have been proposed recently. We will discuss the impact of some of these modifications in the following sections.

5.1 Reducing the granularity of TCP's retransmission timer

The 500msec granularity for the retransmission timer is obviously too large for the ATM environment. The current release of Solaris uses a 200 msec granularity for the retransmission timer [Ste94], and a 50 msec granularity for the delayed acknowledgement timer. We modified SunOS 4.1.3_U1 to use these smaller timer granularities. The measurements with these lower timer granularities (figure 6) showed that the throughput achieved by TCP almost doubled. With a MTU size of 552 bytes, TCP achieved a throughput of almost 6 Mbps with a MBS of 4000 cells, but only 2Mbps with a MBS of 1000 cells. With the larger MTU sizes, the TCP throughput was lower than 2 Mbps.

In SunOS 4.1.3_U1, any kernel timeout, including TCP's retransmission timer, depends on the `softclock` [LMK89] routine which is invoked by a clock interrupt every 10 msec. Thus, the minimum value for the retransmission timer in SunOS 4.1.3_U1 is 10 msec[5]. Figure 7 presents measurements performed with a 10 msec retransmission timer. For these measurements, we disabled the delayed acknowledgement timer in the receiver because this

[5] Please note that this value is much higher than the 100 µsec value used in some simulation studies [RoF94]. To provide such a 100 µsec retransmission timer in BSD-derived implementations, the kernel should process one clock interrupt every 100 µsec. This would be very difficult with today's workstations and operating systems given the high cost of switching from user space to kernel space [Ous90]. Some kind of hardware support for the retransmission timer would probably be necessary to achieve such a 100 µsec retransmission timer with BSD-derived implementations.

timer should ideally have a value smaller than the retransmission timer in order to avoid inaccuracies in the round-trip-time estimation.

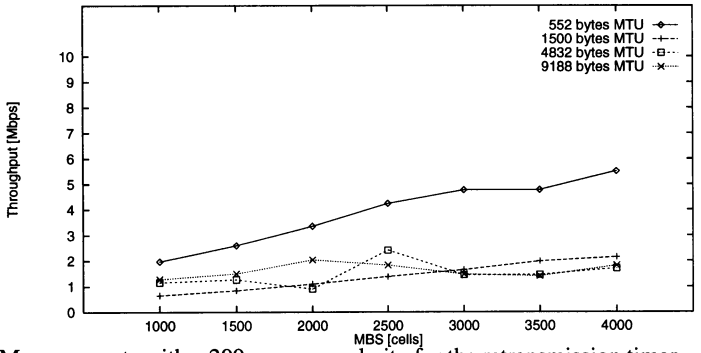

Figure 6 Measurements with a 200 msec granularity for the retransmission timer

From figure 7, one could conclude that the granularity of the retransmission timer should be set to 10 msec and that all the problems will be solved. This is not entirely true. The high throughput achieved with this low granularity is unfortunately accompanied by a huge number of retransmitted packets as shown by figure 8. This figure compares the number of retransmitted packets (reported by `netstat -s`)with the 500 msec, 200 msec and 10 msec retransmission timer granularities and an MTU size of 9188 bytes.

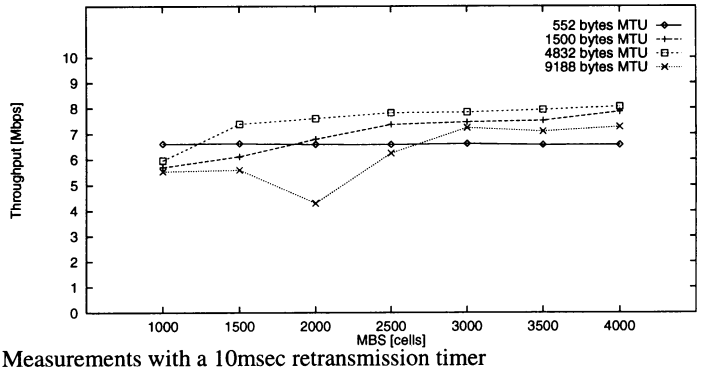

Figure 7 Measurements with a 10msec retransmission timer

We would like to point out that to transmit 4 MBytes of data, TCP must transmit 459 data packets with a MTU size of 9188 bytes. Thus, with the MBS set to 1000 cells and a 10 msec retransmission timer, 80% of the data packets are retransmitted. With the MBS set to 4000 cells, the percentage of retransmitted packets drops to 40 % with the 10 msec retransmission timer, and 20% for the 200 and 500 msec retransmission timers. These high percentages of retransmissions show clearly that TCP, by itself, cannot adapt its behaviour to a VBR traffic contract in an ATM LAN.

The throughput drop which appeared with the MBS set to 2000 cells and a MTU size of 9188 bytes is linked with a kind of race condition. The TCP packet traces revealed an almost periodical behaviour. TCP first sent a burst of about 15 packets. Cells from the last 5 packets of this burst were discarded by the UPC. At that time, TCP's retransmission timer was set to 20

msec (the minimum value). The first retransmitted packet was corrupted with cells from previous incompletely reassembled packets, and thus discarded by the ATM adapter of the receiver. As this retransmission failed, the exponential back-off algorithm set the retransmission timer to 40 msec. The packet sent after the expiration of this timer was acknowledged by the destination. The sender performed slow-start, and successfully retransmitted the other 4 previously transmitted packets. The acknowledgements corresponding to these 4 packets did not update the value of TCP's retransmission timer as specified by Karn's algorithm [KaP89]. Unfortunately the next packets were discarded by the UPC, and thus did not reach the destination. TCP's retransmission timer expired after 80 msec. The first retransmitted packet was discarded by the ATM adapter of the receiver as the corresponding AAL5-PDU was corrupted with cells from the previously transmitted, but unreassembled, AAL5-PDUs. After 160 msec of idle time, TCP attempted a new retransmission. This retransmission was successful, and TCP was able to send a burst of about 15 packets...

With the other MTU sizes and MBS, the retransmission timer did not reach such a high value. This throughput drop would have been avoided with the timestamp options specified in [JBZ92] as when these options are used, TCP can update its retransmission timeout, even when it receives acknowledgements from retransmitted packets.

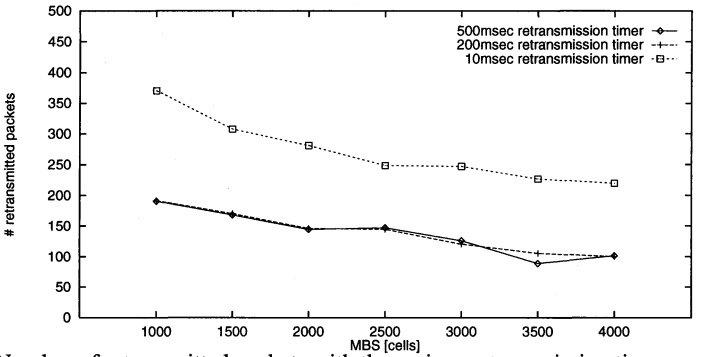

Figure 8 Number of retransmitted packets with the various retransmission timer granularities

5.2 Other retransmission strategies

TCP's retransmission capabilities are probably one of its weakest points. The original specification only supported positive cumulative acknowledgements. Based on these positive acknowledgements, TCP's natural retransmission strategy is go-backn and the retransmissions are triggered by the expiration of the retransmission timer. This is obviously not the best solution in a lossy network. To overcome this limitation, many TCP implementations, including the one we used, support the fast retransmit algorithm. The fast retransmit algorithm [Ste94] assumes that a receiver will immediately send an acknowledgement when it receives an out-of-order data packet. When a TCP sender receives three duplicate acknowledgements pointing to the same data packet, it assumes that the first unacknowledged packet has been lost, and this packet is selectively retransmitted. This solution works well provided that the window size is sufficiently large [BKD96], and that no more than a single data packet is lost within a single window. If these conditions are not fulfilled, TCP will have to wait for the expiration of its retransmission timer, and this may cause large idle times. In our environment, the fast retransmit algorithm was not sufficient to recover from the losses caused by the UPC, even when we set the duplicate acknowledgements threshold to 2. This is due to the fact that there

was no long enough idle time between the last data packet transmitted and the retransmitted packet, and thus cells from the retransmitted packet were also discarded by the UPC.

The other enhancements to TCP's retransmission algorithm proposed recently like TCP Vegas [BOP94] or the new selective acknowledgements option [MMF96] would not have significantly changed the results of our measurements as all these strategies try to retransmit the lost packet as soon as possible. In a wide area network, however, these improved retransmission algorithms would probably be much more useful.

5.3 Traffic shaping

Our measurements have shown that TCP by itself cannot adapt its behaviour to a VBR traffic contract. To fully benefit from the VBR traffic contract with TCP, some additional support would be needed from the ATM driver or adapter. For the third (C in table 1) VBR traffic contract, the solution could be to implement a dual-leaky bucket inside the ATM driver or adapter and to use this leaky bucket to shape the ATM traffic according to the VBR traffic contract. Our ATM adapters did not provide such leaky buckets, but when we set the cell rate of the ATM adapter to 10.000 kbps[6] (i.e. the SCR of the traffic contract), TCP achieved a good utilisation of the VBR VC (figure 9) with the default 500 msec retransmission timer.

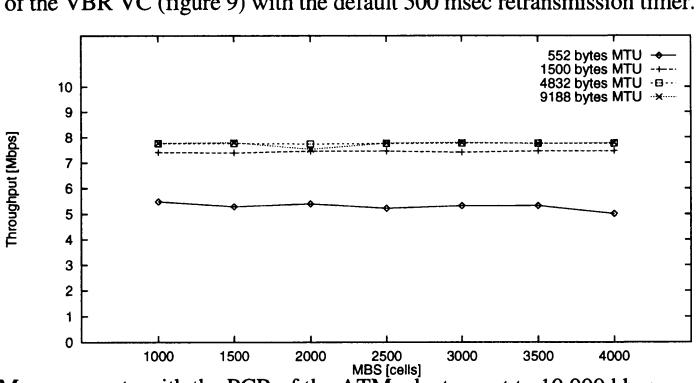

Figure 9 Measurements with the PCR of the ATM adapters set to 10.000 kbps

With the large MTU sizes, the throughput reached about 90% of the theoretical maximum if we take into account the overheads of the ATM cell header, AAL5 trailer and IP and TCP headers. There were no packet losses during the measurements with the 1500, 4832 and 9188 bytes MTU sizes. This is in contrast with the measurements with the 10 msec retransmission timer (figure 7), in which the percentage of retransmitted packets was very high. With a MTU size of 552 bytes, the achieved throughput is below 6 Mbps. This is due to the fact that with this MTU size, the receiving workstation has difficulties to sustain the load, and it drops some received packets. Indeed, 6Mbps is the highest throughput we achieved with TCP, a MTU size of 552 bytes, a window size of 49152 bytes and the default 500 msec retransmission timer in our environment, even when the UPC was disabled.

This traffic shaping could also be implemented inside the ATM driver with a rate control at the AAL5-PDU level. A regular flow of c cells long AAL5-PDUs (figure 4) will be conforming

[6] Of course, by setting the cell rate of the ATM adapter to the SCR of the traffic contract, we do not benefit from burstiness allowed by the VBR traffic contract, but this was the only solution usable with our ATM adapters. In a production network with similar ATM adapters, it may be easier to use the CBR traffic contract instead of the VBR traffic contract when QoS guarantees are required.

with a VBR traffic contract provided that the idle time (I_p) between successive AAL5-PDUs verifies [UNI30] the following equation [7]:

$$c = 1 + \text{floor}\ [\ \min\ (\ I_p - T_{SCR}\ ,\ \tau_{SCR})\ /\ (\ T_{SCR} - T_{PCR})\ ] \qquad (3)$$

It should be noted that the value of I_p given by equation (3) is valid when the source sends a regular flow of c cells long AAL5-PDUs at the PCR during an infinite amount of time. For example, with a MTU size of 9188 bytes (192 cells) and the VBR traffic contract used for our measurements, the minimum value of I_p is 7.6 msec. If the source sends its cells with a cell period larger than T_{PCR}, the minimum value of I_p will be smaller. If the source sends bursts of a few AAL5-PDUs, I_p can be smaller provided that the idle time between successive bursts is large enough. Indeed, the VBR traffic contract allows a source to send bursts of MBS cells at the PCR provided that the idle time between two successive bursts is at least equal to τ_{SCR}. If a rate control mechanism had to be implemented in SunOS, it could be controlled with a routine invoked by `softclock` every 10 msec [KIB96]. If necessary, it could be possible to control both the cell rate of the ATM adapter and the Inter-AAL SDU time.

To fully benefit from the second VBR traffic contract (tagging requested - B in table 1), it would probably be necessary to use some kind of feedback from the receiving ATM adapter or driver. With this traffic contract, the UPC tags the cells which are sent at a rate higher than the rate allowed by the SCR and MBS. If the network is lightly loaded, these cells will reach the destination. If the network is heavily loaded, they will probably be discarded by a congested switch before reaching the destination. Instead of letting the UPC decide which cells should be tagged and which cells should be left unchanged, the sending ATM adapter/driver should already send the cells which are known to be non-conforming with CLP=1 (the last cell of an AAL5-PDU should always be sent with CLP=0 as the loss of this cell will usually corrupt the next AAL5-PDU). With this "preventive" tagging, it will be able to tag entire AAL5-PDUs while still being able to transmit complete AAL5-PDUs with CLP=0 cells. This would help to avoid undesirable situations in which the UPC tags a few cells from each AAL5-PDU. The receiving ATM adapter/driver should also send some feedback to the sending ATM adapter/driver when it notices corrupted CLP=1 AAL5-PDUs, as this might indicate that the network is congested. A similar usage of the high and low priority cells would be necessary to fully benefit from the first (A) VBR traffic contract.

With LAN Emulation and Classical IP over ATM, this could require a new standardised protocol within the ATM or AAL layer. It seems currently unlikely that the IETF or the ATM Forum will work on this subject. A new transport protocol, which uses directly the AAL service (e.g. [AKS96]) could be optimised to adapt its behaviour as a function of the traffic contract of the underlying VC (CBR, VBR or best-effort), but also as a function of the network's congestion level (with the best-effort and second VBR traffic contracts).

6 SHALL WE CHANGE THE UPC ?

In [RoF94], some simulations of a heavily congested ATM LAN have shown that TCP's congestion control had difficulties to adapt to an environment where the ATM switches discard ATM cells and not entire TCP packets when they are congested. This paper has also proposed two modifications, Partial Packet Discard (PPD) and Early Packet Discard (EPD), to the ATM switches. With PPD, an ATM switch should drop the tail of one AAL5-PDU once it has dropped one cell from this AAL5-PDU due to congestion. With EPD, an ATM switch should drop entire AAL5-PDUs when the occupation of its buffer is above some threshold. These discarding strategies have been implemented in some switches, and the next release of the ATM

[7] floor [x] is the largest integer which smaller or equal than x

Forum signalling specification will support a new Information Element that can be used by an end-system to request the network to treat its cell flow as frames (i.e. AAL5-PDUs). It might be tempting to also implement such packet discarding strategies inside the UPC.

PPD could easily be implemented within a UPC at the expense of one additional bit of memory. However, it remains to be seen whether such a modification of the UPC could significantly help protocols such as TCP. Anyway, we believe that it is much better to invest on good traffic shaping capabilities within the ATM driver/adapter than on modifications to the UPC.

It would be more difficult to implement EPD inside a UPC. To implement EPD (i.e. discard entire AAL5-PDUs when they are known to be non-conforming), the UPC should be able to determine whether a complete AAL5-PDU will be conforming upon reception of the first cell from this PDU. However, to do this, the UPC would need to know both the length of the AAL5-PDU as well as the rate at which it will be transmitted. Unfortunately, both these values are unknown when the first cell of one AAL5-PDU is received. With a VBR traffic contract, a UPC implementing EPD could assume that the AAL5-PDU will be sent at the PCR of the contract, and that the AAL5-PDU will be of maximum length (this can be determined from the Maximum CPCS-SDU size parameter found in the AAL Information Element of the signalling messages [UNI30]). However, a UPC which relies on these two assumptions would probably discard AAL5-PDUs (e.g. short AAL5-PDUs) that would have been declared as conforming by a UPC based on the GCRA. This is obviously undesirable.

7 RELATED WORK

The interactions between transport protocols and the CBR and VBR traffic contracts have been studied in several papers. In [BKD96], we shown that in a wide area network where a CBR traffic contract was enforced by a UPC, the conformance of the ATM traffic sent by the adapter was very important. In [KlB96], we discussed similar measurements performed with XTPX (a derivative of XTP) which showed that selective retransmissions are not sufficient to recover from the losses caused by a UPC mechanism when the ATM traffic sent is not entirely compliant with a CBR traffic contract.

Other researchers used simulations to study the interactions between TCP and the VBR traffic contract. [Mis95] reports simulations done in the wide area and in the local area which show similar throughput degradations due to the large granularity of TCP's retransmission timer in most TCP implementations. [EAR95] reports similar simulations. They also propose to replace TCP's slow-start and congestion avoidance algorithms by a leaky bucket and show by simulations that this reduces the number of packet losses. However, this solution is not appropriate from an architectural point of view with both LAN Emulation and Classical IP over ATM. With these two solutions, the TCP entity cannot determine whether the TCP connection uses a VBR VC or not (TCP is not even aware of the existence of ATM). Furthermore, with both LAN Emulation and Classical IP over ATM, several TCP connections can be multiplexed within a single ATM VC. Thus, the traffic shaping must be done on a per-VC basis, within the ATM or AAL layer and not within the TCP layer.

8 CONCLUSION

In this paper, we have presented detailed measurements showing how TCP behaves when it has to adapt its behaviour to a VBR traffic contract enforced by a UPC.

Our measurements have shown that, with the default settings used by most TCP implementations and without shaping at the sender side, TCP has a lot of difficulties to adapt to such a traffic contract. Indeed, with these settings, TCP only achieved a 10% utilisation of a 10 Mbps VBR VC during our measurements. We have also shown that reducing the granularity of

TCP's retransmission timer, even down to 10 msec, did not solve all the problems. With this small granularity, TCP achieved a good utilisation of the VBR VC, but this was at the expense of an unacceptable number of retransmitted packets (more than 80% with an MTU size of 9188 bytes and a MBS of 1000 cells).

In [BKD96], we had already shown that the ATM adapters should provide good ATM-level traffic shaping capabilities to support CBR VCs. The measurements presented in this paper clearly show that these traffic shaping capabilities must also cover the SCR and the MBS, otherwise many applications and legacy protocols will have problems to use a VBR VC. With legacy applications or protocols above LAN Emulation or Classical IP over ATM, this traffic shaping should be performed automatically by the ATM driver or adapter. It should be noted that while our measurements were performed with TCP, they apply to any protocol which does not use some kind of direct or indirect (e.g. with motion-JPEG, there is one burst of cells for each picture and an idle time between two successive bursts) rate control.

9 REFERENCES

[AKS96] Ahuja, R., Keshav, S., Saran, H. (1996) Design, Implementation and Performance of a Native Mode ATM Transport Layer, in *Proc. IEEE Infocom 96*

[Atk94] Atkinson, R., *Default IP MTU for use over ATM AAL 5*, Internet RFC 1626

[BKD96] Bonaventure, O., Klovning, E., Danthine, A. (1996) Behaviour of TCP in the European ATM Pilot, *Computer Communications*, 19(3), 264-275

[BOP94] Brakmo, L., O'Malley, S., Peterson, L. (1994) TCP Vegas: new techniques for congestion detection and avoidance, *ACM Computer Communication Review*, vol. 24, no. 4, pp. 24-35

[Cyp96] Cyphers, R. (1996), Fore Systems technical support, personal communication

[DeP93] De Prycker, M. (1993) *Asynchronous Transfer Mode : Solution for Broadband ISDN*, second edition, Ellis Horwood

[EAR95] Elloumi, O., Afifi, H., Rolin, P., Hamdi, M. (1995) Issues in Improving TCP Performance over ATM, in *1st IFIP conference on ATM Traffic Management*, Paris, December 1995

[Gue91] Guerin, R., Ahmadi, H., Naghshineh, M. (1991) Equivalent Capacity and Its Application to Bandwidth Allocation in High Speed Networks, *IEEE Journal on Selected Areas in Communications*, Vol 9 No 7, Sept. 1991, pp 968-981

[I363] ITU-T (1993a), *B-ISDN - ATM Adaptation Layer specification*, ITU-T Rec. I.363

[I371] ITU-T (1993b), *Traffic control and congestion control in B-ISDN*, ITU-T Rec. I.371

[JBZ92] Jacobson, V., Braden, R., Zhang, L. (1992) *TCP extensions for high performance*, RFC1323

[KaP87] Karn, P., Partridge, C. (1987) Improving Round-Trip Times Estimates in Reliable Transport Protocols, *ACM Computer Communication Review*, vol 17, n. 5, 2-7

[KlB96] Klovning, E., Bonaventure, O. (1996) Behaviour of XTPX in the European ATM Pilot, *European Transactions on Telecommunications*, special issue on "ATM Field Trials and Experiments", Vol. 7, No. 5, September-October 1996

[LAN95] ATM Forum (1995), *LAN Emulation over ATM v 1.0*

[Lau94] Laubach, M. (1994), *Classical IP and ARP over ATM*, RFC 1577

[LMK89] Leffler, S., McKusick, M., Karels, M., Quaterman, J. (1989), *The Design and Implementation of the 4.3 BSD UNIX Operating System*, Addison-Wesley, Reading, Mass.

[Mis95] Mishra, P. (1995), Throughput degradation for TCP in the Presence of Traffic Policing, in: Report and Discussion on the IEEE ComSoc TCGN Gigabit Networking Workshop 1995 (J. Sterbenz, H. Schulzrinne, J. Touch), *IEEE Network Magazine*, July-August 1995

[MKK94] Moldeklev, K., Klovning, E., Kure, O. (1994) TCP/IP behaviour in a high-speed local ATM network environment, in *Proc. 19th Conference on Local Computer Networks*, Minneapolis, USA, pp. 176-185

[MMF96] Mathis, M., Mahdavi, J., Floyd, S., Romanow, A. (1996) *TCP Selective Acknowledgement Option*, Internet Draft, work in progress

[MoG95] Moldeklev, K., Gunningberg, P. (1995), Deadlock Situations in TCP over ATM, in *Protocols for High Speed Networks* (G. Neufeld and M. Ito, eds.), Chapman&Hall

[Ous90] Ousterhout, J. (1990) Why aren't operating systems getting faster as fast as hardware ?, in *Proc. of the summer 1990 USENIX conference* - similar measurements with recent workstations have been performed by H. Schulzrinne and may be found at http://www.fokus.gmd.de:80/step/hgs/bench/summary.html

[RoF94] Romanow, A., Floyd, S. (1994) The Dynamics of TCP traffic over ATM Networks, Proc. ACM SIGCOMM'94, *ACM Computer Communication Review*, vol. 24, no. 4, 79-88

[Sat96] Sathaye, S. ,Ed. (1996), *ATM Forum Traffic Management Specification Version 4.0*, ATM Forum Traffic Management Working Group, Letter ballot version

[SlM84] Slattery, T., Muss, M. (1984), `ttcp.c`, available as ftp://ftp.brl.mil/pub/ttcp.c . See also ftp://www-run.montefiore.ulg.ac.be/pub/soft/ttcp2.c for the version we used.

[Ste94] Stevens W.R. (1994), *TCP/IP Illustrated, volume 1 : the Protocols*, Addison-Wesley, Reading, Mass.

[UNI30] ATM Forum (1993), *ATM User Network Interface Specification -Version 3.0*, Prentice-Hall

[WrS95] Wright, G. and Stevens, W.R. (1995), *TCP/IP Illustrated, volume 2 : the Implementation*, Addison-Wesley, Reading, Mass.

10 BIOGRAPHY

Olivier Bonaventure graduated from the University of Liège in 1992. Since then, he has worked as a research engineer in the research unit in networking and has been involved in the ESPRIT OSI95 project and the RACE CIO and ACCOPI projects. He is currently working on transport protocols for the ATM environment within the OKAPI ACTS project.

Espen Klovning received the M.S degree from the Norwegian Institute of Technology in 1992. He is currently a research scientist at Telenor Research and Development, Kjeller, Norway, where he has worked since 1993 with high performance communication systems.

André Danthine, Professor at the University of Liège since 1967, created a research unit in networking (RUN) in 1972. In 1995, the ATM LAN of RUN has been connected, at 34 Mbps, to the European ATM Pilot for functional and performance evaluations (RACE Project CIO). Prof. Danthine is the chairman of the COST237, member of ACM, of IEEE, Editor of "ETT", of several books and proceedings. He is Governor of ICCC since 1982. CRB Fellow in 1960, he received the Melchior Salier Prize in 1961, the "Bell Telephone-100th Anniversary" prize in 1983 and the IFIP Silver Core in 1986. In 1993-1994, he was Francqui Professor at VUB. He is Doctor Honoris Causa of the University of Kent (1991) and of the Université Paul Sabatier in Toulouse (1996).