ON THE PERFORMANCE OF TRANSPORT PROTOCOLS OVER ATM NETWORKS

M.Ajmone Marsan[1], M.Baldi[2], A.Bianco[1], R.Lo Cigno[1], M.Munafò[1]
[1] - Dipartimento di Elettronica – Politecnico di Torino – Italy
[2] - Dipartimento di Automatica ed Informatica – Politecnico di Torino – Italy
e-mail: {ajmone,baldi,bianco,locigno,munafo}@polito.it

ABSTRACT

The performance of file transfer applications running over B-ISDN is investigated, taking into account both the impact of the ATM layer protocols and the features of the adopted transport protocol. Two transport protocols in particular are considered: TCP and XTP. The two transport protocols have quite different characteristics: while TCP is a window-based protocol, XTP is rate-based; while TCP contains algorithms for congestion avoidance, XTP includes no such feature. The performance investigation was performed by simulation, using CLASS, a simulation tool for the study of ATM networks at the cell and burst levels.

1 Introduction

After more than a decade of research and experimentation, ATM (Asynchronous Transfer Mode) networks are now becoming available to users both in the local and in the wide area: ATM LAN (Local Area Network) products are reaching stability, and ATM WANs (Wide Area Networks) are becoming commercially accessible in a number of countries. Network operators are thus beginning to provide B-ISDN (Broadband Integrated Services Digital Networks) services, such as video-on-demand, high-speed LAN interconnection, and high-speed file transfer.

While much research is in progress to investigate traffic and performance aspects of ATM networks, so as to correctly engineer B-ISDN services, the performance issues relating to higher layer protocols when used over ATM networks are receiving less attention. In this paper we use a simulation approach to study the performance that can be achieved for file transfer applications over ATM networks with simple topologies. by using either TCP or XTP [1] (Xpress Transport Protocol), the former Xpress *Transfer* Protocol specified by Protocol Engine Inc. [2].

The motivations for considering TCP and XTP are numerous. TCP is an obvious choice, since it is, and it will remain for some time to come, by far the most widely used transport protocol, even though it was not expressly developed for a high bandwidth-delay environment. On the contrary, XTP was designed for high-speed networks, and it is one of the earliest representatives of the class of rate-based transport protocols, contrary to TCP that is window-based. The rate-based versus window-based debate has already consumed many pages of papers on ATM networks, but at a different architectural level; the winning approach within the ATM layer at present seems to be the rate-based paradigm adopted, for example, in the ABR (Available Bit Rate) ATM transfer capability, while it seems that the flow control techniques based on credits, which are related to window-based protocols, have lost momentum. Although the issues in the design of a transport protocol are quite different from those in the design of algorithms at the ATM layer, one might argue that the choice of a rate-based approach also at the transport layer may not be unreasonable. A further significant difference between TCP and XTP concerns the congestion prevention approach. Whereas congestion avoidance algorithms are embedded within the standard versions of TCP, no such feature is included into XTP. The effectiveness of the congestion avoidance approaches of TCP in ATM networks still has to be carefully determined.

The objective of our study is to determine whether the two transport protocols offer significantly different performances in some simple ATM network scenarios, and to assess the effectiveness of cell spacing as a means to improve performance of file transfer applications within the ATM layer.

In our study we did not consider the traffic control mechanisms proposed to efficiently manage best-effort services in ATM networks, namely the closed-loop control mechanism called ABR (Available Bit Rate) proposed by the ATM Forum [3] and the "fast reservation protocol" named ABT (ATM Block Transfer) proposed by the ITU-T [4]. Instead we study the performance of file transfer applications exploiting the simple UBR transfer capability. The main reason is that we are interested in looking at the basic mechanims induced on transport layer protocols by the segmentation process

required at the ATM layer; introducing more sophisticated control tecniques significantly improves the performance figures, but it could mask some of the phenomena we are interested in.

The paper is organized as follows. Section 2 very concisely overviews the main features of CLASS, the ATM network simulation tool used in the study, and illustrates in some detail the implementations of TCP and XTP within the simulator. Section 3 illustrates the ATM network topologies considered in the study, and all the relevant system parameters. Section 4 describes and comments the numerical results obtained with the simulator. Finally, Section 5 presents our concluding remarks, and the future directions of this study.

2 The Simulation Software

The results presented in this paper were obtained with the ATM network simulator CLASS [5]. CLASS has been developed at the Electronics Department of Politecnico di Torino in co-operation with CSELT (Centro Studi e Laboratori Telecomunicazioni, the research center of Telecom Italy) and the Technical University of Budapest. CLASS is a synchronous, time driven, cell- and burst-level ATM network simulator, whose accuracy has been tested against on-field measurements performed in CSELT. This simulator allows the user to describe networks with arbitrary topologies, comprising both connection-oriented and connectionless traffic. The main network entities, i.e., nodes, links and users, can be instantiated with different parameters, in order to simulate heterogeneous networks. Results concerning throughputs and delays can be collected on the overall network as well as on the single connections, and sophisticated statistical analysis is performed to assess the confidence interval and the confidence level of the results. The description of CLASS goes beyond the scope of this Section, but the interested reader can find more information about CLASS, works related to it and how to obtain a version of the simulator on the Web at the URL `http://hp0tlc.polito.it/class.html`.

Modeling complex protocols like TCP or XTP can be a hard job, since an oversimplification of the embedded algorithms may result in behaviors and performances quite different from those of the true protocol. For this reason we decided, instead of creating a specific model, to add a simplified version of two distributed releases of the TCP and XTP protocols to run on top of the AAL layer within CLASS. We chose the officially distributed BSD 4.3-reno release for TCP [6] and the SandiaXTP 1.3 C++ code [7] that implements XTP 4.0 and is released by Sandia National Laboratories.

2.1 The TCP simulation model

Since TCP is a well known protocol, we briefly outline here only the main characteristics of the TCP-reno congestion control algorithm; further details can be found in [6, 8].

TCP is a connection-oriented protocol based on a window mechanism that regulates the flow of data units. The window size at the transmitter is dynamically incremented until either congestion is detected into the network, or a maximum window size, advertised by the receiver, is reached. The receiver sends back acknowledgments (ACKs) of the received data (piggybacking it onto data segments if possible); receiving (or not receiving) ACKs at the transmitter triggers either the transmission of a new segment (if available) and the increase of the window, or the beginning of congestion control procedures.

At the transmitter, a threshold is initially set to half the maximum receiver advertised window size, and the actual window size is set to one segment. The window increases in two phases: the *slow-start* (or *congestion recovery*) phase, where the window grows esponentially, and the *congestion avoidance* phase, where the window increases linearly. During the slow-start phase, two data segments are transmitted for every received ACK; as a consequence, the window size doubles at each *transmission cycle*, defined as the sequence of events corresponding to the transmission of a number of segments equal to the current window size, and to the reception of the corresponding ACKs. During the congestion avoidance phase, the window size is increased by one segment at every transmission cycle; one segment is transmitted for every received ACK, but once every window an ACK causes the transmission of two segments, thus determining the linear increment. The transmitter goes from the slow-start phase to the congestion avoidance phase when the actual window size reaches the current threshold. When the maximum window size is reached, the transmission continues without further increasing the window size.

The source considers the loss of a segment as a symptom of congestion, and reacts reducing the transmission window size. The segment loss can be detected either through a timer expiration (the ACK is not received within a defined delay from the segment transmission), or by the *fast recovery algorithm*. If a retransmission timer expires, the transmission window size is reduced to one segment, the threshold is set to half the current window size and the slow-start phase is entered. Using the fast recovery procedure, the source node infers that a segment was lost if four ACKs referring to the same segment are received; in this case the source reacts by halving the current transmission window size, and switching to the congestion avoidance phase.

Since the BSD 4.3-reno TCP is not tailored to high-speed networks, slight modifications were required to adapt its code to the ATM environment; the most important aspects of this adaptation process are described below. We included practically all the important TCP features, with the exception of the delayed ACK and the selective ACK options. The selective ACK option could significantly modify the efficiency of TCP, specially for long-distance connections where the transmission pipe

is very long and a single segment loss can result in the retransmission of a large number of segments, even if the cumulative ACK of the received data used in TCP avoids a simple go-back-N behavior. We did not consider this option, since most of the running versions of TCP do not include it. The delayed ACK option has a marginal effect in the scenarios we consider.

In order to allow a single connection to grab all the available bandwidth of a high-speed link, specially in a WAN environment, it is necessary to use a window of reasonably large size, hence the maximum window dimension is left as parameter in our implementation.

Another important aspect is related to the timeout setting and to the RTT (Round Trip Time) estimation. Timeouts in TCP are computed as a smoothed estimate based on the RTT average μ_{RTT}, and standard deviation σ_{RTT}, as $t_o = \mu_{\mathrm{RTT}} + 4\sigma_{\mathrm{RTT}}$; the average RTT and its standard deviation are computed following the algorithm proposed by Van Jacobson. In present implementations, the *granularity* of TCP timers (the minimum value to which a timeout can be set) is rather coarse (roughly 500 ms), and it must be reduced if the protocol has to react fast on high bandwidth-delay product networks. This parameter can be chosen by the user in our TCP implementation. As a rule of thumb, the granularity should be smaller than the propagation delay along the connection, and it should be equal for all connections, if fairness among connections sharing the same resources has to be achieved.

2.2 The XTP simulation model

TCP, like other transport protocols presently in use, was designed for networks with low bandwidth-delay product and high error-rates. As LAN and WAN networks evolve and exhibit high bandwidth-delay products and low error-rates, these protocols are being used in environments different from those for which they were originally designed. Starting from this consideration, many protocols called *light-weight transport protocols*, like Delta-t [9], Datakit [10], NETBLT [11], and VMTP [12], were developed, explicitly aiming at the new environment. The Xpress Transport Protocol (XTP) was designed with the same aim, and it incorporates the main contributions brought by the earlier efforts.

Xpress Transport Protocol version 4.0 is independent of the underlying data delivery service, from which it only requires that its *frames* (the data transmisison unit) be delivered to the XTP-equipped destination. Thus, XTP can run directly over MAC protocols in LANs, as well as over AAL5 in an ATM network.

XTP is a connection-oriented protocol; connections are named *associations*, and are uniquely identified at a host by a *key* included in the header of each packet. The status of a connection is kept by each host participating in the connection, and is named *context*.

XTP provides six types of service, spanning from a best-effort datagram service to a reliable multicast stream service. In our simulations we adopt a traditional reliable unicast stream service, and each connection is mapped onto an ATM virtual circuit connection (VCC).

Frames are identified by 64-bit sequence numbers, thus not limiting (actually limiting to very large quantities) the maximum number of unacknowledged data being traveling through the network.

XTP supports both flow and rate control. The former is regulated by the receiver and is based on the amount of space available in its buffer. Rate control is based on two parameters:

1. `burst`: the maximum number of bytes that can be sent consecutively;

2. `rate`: the maximum number of bytes that can be transmitted in each second.

The variable `credit` and `RTIMER` are used by XTP transmitter to control the rate of frames generation: `credit` is initialized to `burst`, and the XTP source entity can transmit up to `credit` bytes; after having transmitted a frame, it decrements `credit` by the number of transmitted bytes and starts the timer `RTIMER` initialized to `burst/rate`. When `credit` reaches zero, no data can be transmitted until `RTIMER` expires; when `RTIMER` expires, `credit` is again set to `burst`.

As we aim at comparing the rate-based control mechanism of XTP with the window-based flow control mechanism of TCP when both run over ATM networks, we would have liked to disable any traffic control mechanism other than rate control. Actually, disabling the XTP flow control (the standard allows to do so) does not guarantee that rate control remains the only mechanism that regulates transmission speed. In fact, when error control is enabled, the sender keeps unacknowledged data in its output buffer to be able to retransmit them, if necessary. When the network is long and congested, a significant percentage of XTP frames is lost, and the output buffer becomes full because ACKs (called *control packets*) are not received; as soon as the transmitter output buffer becomes full, the transmitter must stop until at least one of the outstanding frames is acknowledged, so that the data it carried can be removed from the output buffer, and some space can be freed. Thus, the transmitter output buffer acts as a fixed-size window at the transmitter, introducing a sort of flow control mechanism.

XTP retransmits unacknowledged frames based on either a *go-back-N* or a *selective retransmission* scheme. In both cases not-acknowledgment (NAK) frames (called *error control packets*) carry the sequence number of the next in-sequence byte expected by the receiver. When selective retransmission is applied, each error control packet identifies the subset of bytes, following the last in-sequence one, that have been correctly received; the transmitter retransmits only missing data.

The XTP specification does not state when ACKs and NAKs are to be sent by the receiver. ACKs (NAKs) are requested by the transmitter by setting a bit in the header of the data frame to be acknowledged; the receiver sends back ACKs (NAKs) as soon as it gets a request. Moreover, the transmitter, by setting a different bit in the header of data frames, can request the receiver just to send a NAK if some data are missing (fast NAK option).

After having requested an ACK, the transmitter starts a timer called `WTIMER`; the transmitter expects to receive the ACK (NAK) before the timer expires. `WTIMER` is loaded with a smoothed round-trip time (RTT) estimate plus twice the RTT variance (notice the similarity with the timeout t_o defined in TCP). The smoothed RTT estimate and its variance are calculated (in the SandiaXTP 1.3 implementation) through the same smoothing function used by TCP and proposed by Van Jacobson ([1] pp 27-28). The transmitter measures the actual RTT each time it receives the ACK (NAK) corresponding to the last request it made.

If the control packet does not arrive before the expiration of `WTIMER`, the transmission of data is suspended and a *synchronizing handshake* is started: the transmitter sends a request control packet that contains the status of the transmitter and demands an ACK containing the actual status of the receiver. `WTIMER` is loaded with the usual value. If the reply is not received before the timer expires, a new request control packet is issued, and `WTIMER` is doubled. The process is repeated until an ACK is received by the transmitter before the expiration of `WTIMER`. This terminates the synchronizing handshake; now each one of the two connection end-points precisely knows the status (which data bytes have been sent, which have been received, and which are missing) of the other, and the data flow can resume.

Even though the XTP 4.0 specification does consider the possibility of having one timer per ACK request, it does not impose to implement them: a single timer which is overwritten at each new request can be used and the implementation we use adopt this option. If an ACK is requested before the previous one has been received (and before `WTIMER` has expired), the transmitter reinitializes `WTIMER`; thus, previous ACK requests cannot trigger a synchronizing handshake anymore.

Our XTP transmitter implementation requests ACKs at regular intervals; the number of data frames between two consecutive ACKs, which we call the *ACK request interval*, is a simulation parameter.

3 The Simulation Scenario

The dynamics of protocols like TCP and XTP over ATM networks are clearly influenced by many different parameters, such as the network topology, its span, the buffer size available in the network nodes, and so forth.

We investigate through simulation the performance of both TCP and XTP over ATM networks in different scenarios, trying to isolate behaviors due to the interaction of the protocols under study with the main ATM characteristics, from secondary effects due to other parameters. We base our study on connections in sustained overload resulting from a long file transfer, i.e., we assume that sources always have data to send; data are transmitted as fast as allowed by the control mechanism of the transport protocol.

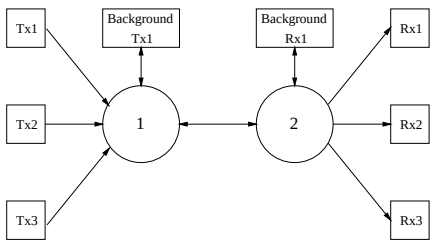

Figure 1: The bottleneck topology

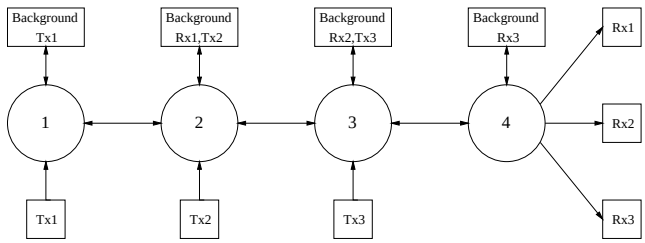

Figure 2: The parking lot topology

The basic topology we refer to in this paper is reported in Fig. 1. Three transmitters are connected to one ATM switch, and the three corresponding receivers are connected to the other one. The transmitter/receiver pairs can adopt either TCP or XTP as their transport protocol. A variable amount of background traffic shares the link between the two ATM switches with the three transport connections. We refer to this topology as the 'bottleneck' topology in the remainder of the paper.

The bottleneck topology allows insight to be gained into the interaction between the ATM network and the considered transport protocols, but it could be claimed that it is far too simple to represent a realistic scenario. To investigate how the network topology influences the interaction between ATM and the upper level protocols, we thus also present results concerning a more complex network topology, depicted in Fig. 2, and generally known in the literature as the 'parking lot' topology. In this scenario, 3 TCP (XTP) connections share the bandwidth of the links connecting four ATM switches with a variable amount of background traffic load; since the average background traffic load is the same in all node-to-node links, the link connecting the ATM switch number 3 with the ATM switch number 4 becomes the system bottleneck. The background traffic loads in the different links are taken to be statistically independent of one another.

In the two considered topologies, the data rate on all channels, both user-node and node-node, is set to 150 Mbit/s, and the size of the buffers inside ATM switches is set to 1000 cells. The data flow of the TCP (XTP) connection is assumed to be unidirectional: transmitters send data segments, and TCP (XTP) receivers return only ACK segments (control packets); TCP ACK segments are assumed to fit in one cell, while XTP control packets are two cells long (ACKs) or more (NAKs). Losses are avoided both in the user transmission buffer and at the receiver's: they can occur only within ATM switches. The connections lengths are taken to be all equal to 1000 km, regardless of the topology. Simulations have also been run for much shorter networks, but the TCP (XTP) behaviors when the measured round trip time is short make these results far less interesting, as explained in Section 4.3, thus they are not fully presented here for the sake of brevity.

The maximum TCP window size and the XTP ACK request rate are taken as variable parameters, as well as the dimension of the "useful data PDU", i.e., the number of user data bytes that are transmitted within each TCP segment or XTP frame. We take as reference value 9140 bytes, that, when the TCP, IP and AAL5 overheads are added makes an AAL5 CS PDU of 9188 bytes, which complies with the suggested maximum segment size for TCP/IP over ATM. With the 32 bytes of the XTP header and the AAL5 overheads, XTP sources generate AAL5 CS PDUs of 9180 bytes.

XTP sources are rate controlled and generate traffic at an average of 50 Mbit/s. The burst length of the rate controller is set to one packet because high burstiness is usually critical in ATM networks; thus an XTP source transmits one data frame of the chosen length and then waits for `RTIMER` to expire. TCP sources generate traffic as fast as their actual window allows. Each simulation is run both with and without a traffic shaper operating at 50 Mbit/s with cell delay variation tolerance (CDVT) equal to zero. The shaping device operates according to a modification of the Generic Cell Rate Algorithm [4].

The XTP flow control was disabled, but the size of the transmission buffer is set to the same value of the TCP window, i.e., it is chosen large enough to allow the three connections to exploit all the available bandwidth when no background traffic is present.

Concerning error control in XTP, the fast NAK option is always used, and both the go-back-N and selective retransmission schemes are used in order to compare their effectiveness in the various scenarios.

The background traffic results from the segmentation of user messages generated according to a Poisson process, with a truncated geometric message length distribution whose mean is equal to 20 cells and whose maximum is equal to 200 cells. The background traffic is then shaped with an allowed peak cell rate equal to 1.2 times the average cell rate, thus introducing only a small burstiness.

4 Numerical Results

The main performance figures we present in this study are the *goodput*, i.e., the effective amount of user data that are transferred from the transmitter to the receiver, discarding all overheads, faulty segments or frames, and possible retransmissions, and the *efficiency*, i.e., the ratio between the goodput and the user offered load. The goodput is a measure of how the end user would perceive the performance of the network. The efficiency, on the other hand, is a measure of how well the network is exploited.

The following subsections present first the results obtained using TCP in the simple bottleneck topology, then those obtained with XTP. Finally, a comparison between the two protocols is attempted, considering also more complex scenarios. If not otherwise stated, the performance figures are averaged over the 3 connections: the exact behaviors of individual connections lie within 1–2% of the average.

4.1 TCP over a 1000 km bottleneck topology

Although a number of analysis of the performance of TCP connections on ATM networks has already appeared [13, 14, 15], the impact of such basic parameters as the maximum window and segment size on the achievable performance has only partially been assessed, to the best of our knowledge. Fig. 3 reports the performance of 3 TCP connections in the bottleneck topology as a function of the background traffic level. Each plot reports the goodput and the efficiency of the connections, both in the case when no shaping is applied (square markers) and in the case when TCP transmitters shape their traffic at 50 Mbit/s (circular markers). The plots of the left column refer to the case when the maximum TCP window size in bytes $W_b = MSS \cdot W$ (where MSS is the maximum segment size in bytes and W is the window size in segments) is set so as to allow each connection to exploit roughly 1/3 of the link bandwidth if the RTT estimate is equal to the propagation delay. Plots in the right column consider a 3 times larger value for W_b, either accounting for the possibility for each connections to exploit the whole link capacity, or to maintain a high throughput even if the queueing delays increase the RTT estimate. The three rows refer to results obtained with different maximum segment sizes in bytes. The upper one refers to the case of $MSS = 9140$ which is the recommended maximum segment size for TCP over ATM networks, the middle row refers to the case of $MSS = 4570$, while the lower row refers to the case $MSS = 1142$; the dimension of the maximum window size in segments W is increased so as to keep the dimension W_b roughly constant.

The curves in the left column show that, when the background traffic is light, and the maximum TCP window is dimensioned so as to allow the link capacity to be utilized when all connections are transmitting, the network resources are very well exploited. The TCP

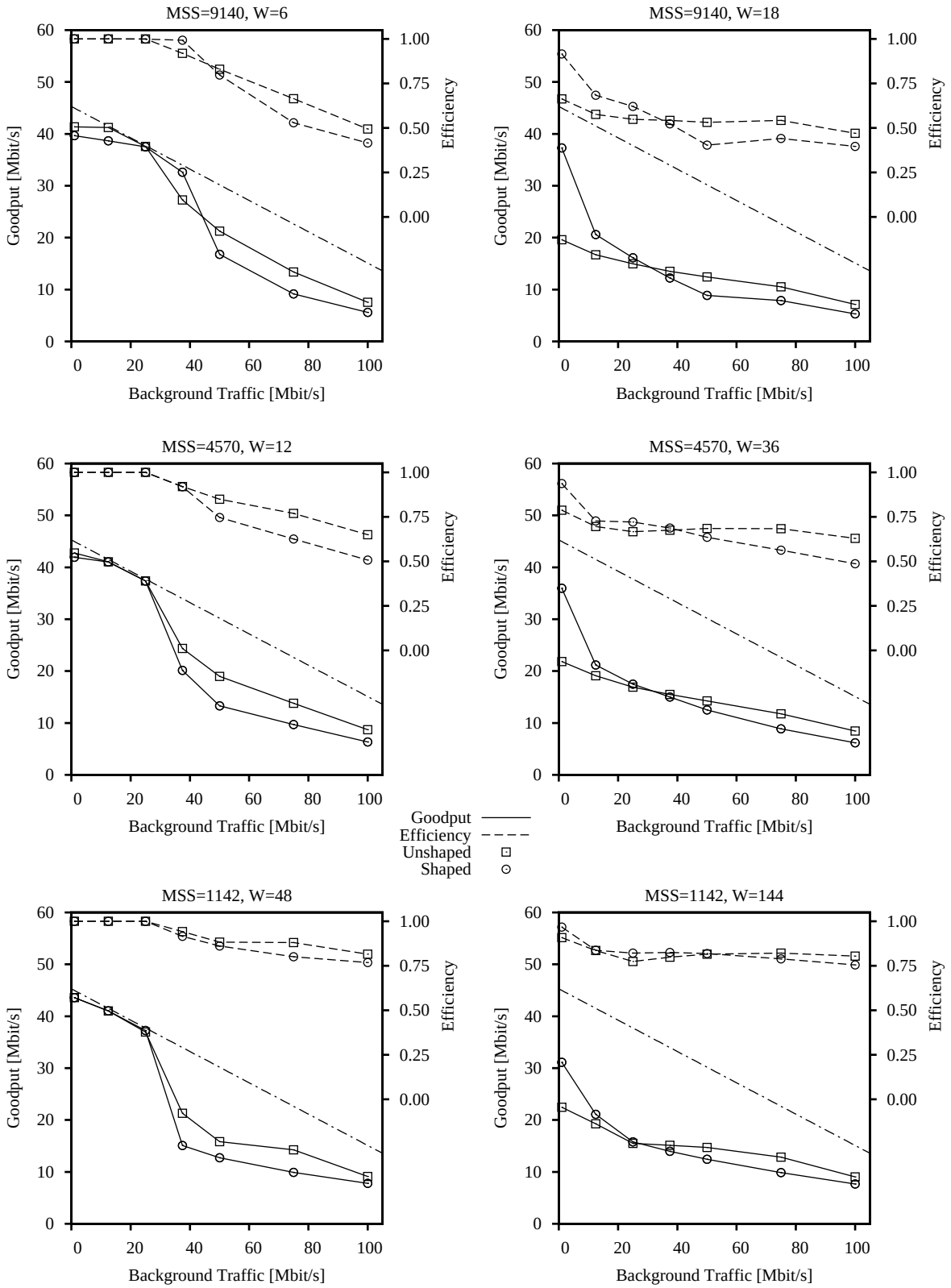

Figure 3: Performance of 3 TCP connections on the 1000 km bottleneck topology, as a function of the background traffic load, the maximum window, and segment sizes

goodput is in fact very close to the available free capacity (one third of the link capacity minus the background load, as indicated by the dot-dashed line), and the efficiency of the transmission is always 1. For higher loads, the network becomes lossy and both the goodput and the efficiency sharply drop.

Comparing the left and right columns, it is quite evident that the maximum window setting has a remarkable impact on the performance that can be obtained by TCP: if W_b is too large, the performance of TCP becomes very poor. The reason for this behavior is the following: a larger window implies the possibility of transmitting more data per transmission cycle, but also a more aggressive policy in the window size increase, since the slow-start phase (the one with exponential window growth) is longer: under these conditions the actual transmission window oscillates at a higher frequency, and its average value (that determines the obtained throughput) is smaller. Besides, it must be noted that very large windows generate congestion even when the background load is very light, a fact that further decreases the performance of the network. A preliminary conclusion is that in order to exploit network resources well, the TCP window must be carefully determined: an overestimation of the window size can be quite dangerous.

Consider now the effect of the maximum segment size. Its impact on the goodput is negligible; however, if we consider efficiency, it is quite clear that the smaller the segments the better the efficiency, specially when the network is overloaded. Clearly, the maximum segment size cannot be reduced too much, otherwise the overheads would become too large. At this point it may be worthwile to stress the importance of efficiency. When considering efficiency, it must be taken into account that a sharp drop indicates most of all that the network is bringing to destination cells that are worthless, either because they belong to damaged segments, or because they are unnecessarily retransmitted. Someone has to pay for the this useless traffic, either the user, or the network operator, but neither receives any advantage from it.

The final consideration on this set of results is that, in this scenario, shaping the TCP traffic does not significantly alter performance, possibly even somewhat worsening them. This is due to two different reasons: i) traffic shaping can improve the performance when short term congestion is the key issue, while in the considered case the congestion is due to a long term effect, ii) spreading the cells of a single segment over time, makes it more likely that one of the cells is lost due to the background traffic fluctuations.

4.2 XTP over a 1000 km bottleneck topology

The first set of numerical results referring to XTP investigates the impact of the XTP frame size on the system performance. Results are presented in the six plots of Fig. 4, assuming that XTP transmitters ask receivers to return one ACK for each frame. Three different sizes for XTP frames are considered (9140 bytes in the upper row, 4570 in the middle row, and 1142 in the bottom row). Both the go-back-N (left column) and the selective retransmission schemes (right column) are studied. Each plot shows curves for both the average goodput and efficiency of the three XTP connections as functions of the background traffic load, in the cases of shaped and unshaped XTP traffic.

First of all, we can observe that the numerical results show that in these cases shaping does not significantly modify the performance of XTP connections. This result is not surprising, since the main characteristic of XTP is that it is rate-controlled at the frame level; it is thus reasonable that a further rate control at the cell level has a minor impact on performance.

It is also immediately evident that in all curves, when the background traffic is low, XTP connections succeed in exploiting most of the available capacity (indicated by the dot-dashed line) without any frame loss (the efficiency equals 1). As the background load grows, both the goodput and the efficiency of XTP connections drop sharply, due to the fact that the only XTP mechanism to react to congestion is the synchronizing handshake. Indeed, XTP sources transmit regularly-spaced frames at the average rate of 50 Mbit/s; when a synchronizing handshake starts, the transmitter stops sending data frames, thus reducing the load on the network. In our scenario, the RTT is very large compared to the time between two consecutive ACK requests, i.e., the minimum time interval between consecutive transmissions of data frames (value loaded in `RTIMER`). Thus, each ACK request causes `WTIMER` to be overwritten, and the synchronizing handshake not to be triggered. In other words, if the output buffer is at least as large as the connection pipe and the network is not congested (no data frame is lost), the transmitter never stops sending data because, as soon as ACKs arrive, the corresponding room in the output buffer is freed, so that such buffer does not fill up.

In our simulations, the output buffers of XTP sources are slightly smaller than the connection pipe, considering empty buffers in the switches; thus, each XTP source transmits a full output buffer of data, and then it stops and waits for the ACKs; after a time period shorter than `WTIMER` has been elapsed, they begin to arrive.

With zero background traffic, the XTP sources can send the amount of data equivalent to a full output buffer without congestioning the network: the efficiency is 1, but since some time is wasted waiting for ACKs, the bandwidth they use is smaller than the raw channel data rate.

When the background traffic is low, each XTP source transmits a full output buffer of data without overflowing the buffer of the first switch (efficiency is one and no frames are lost), but RTT increases, and so does

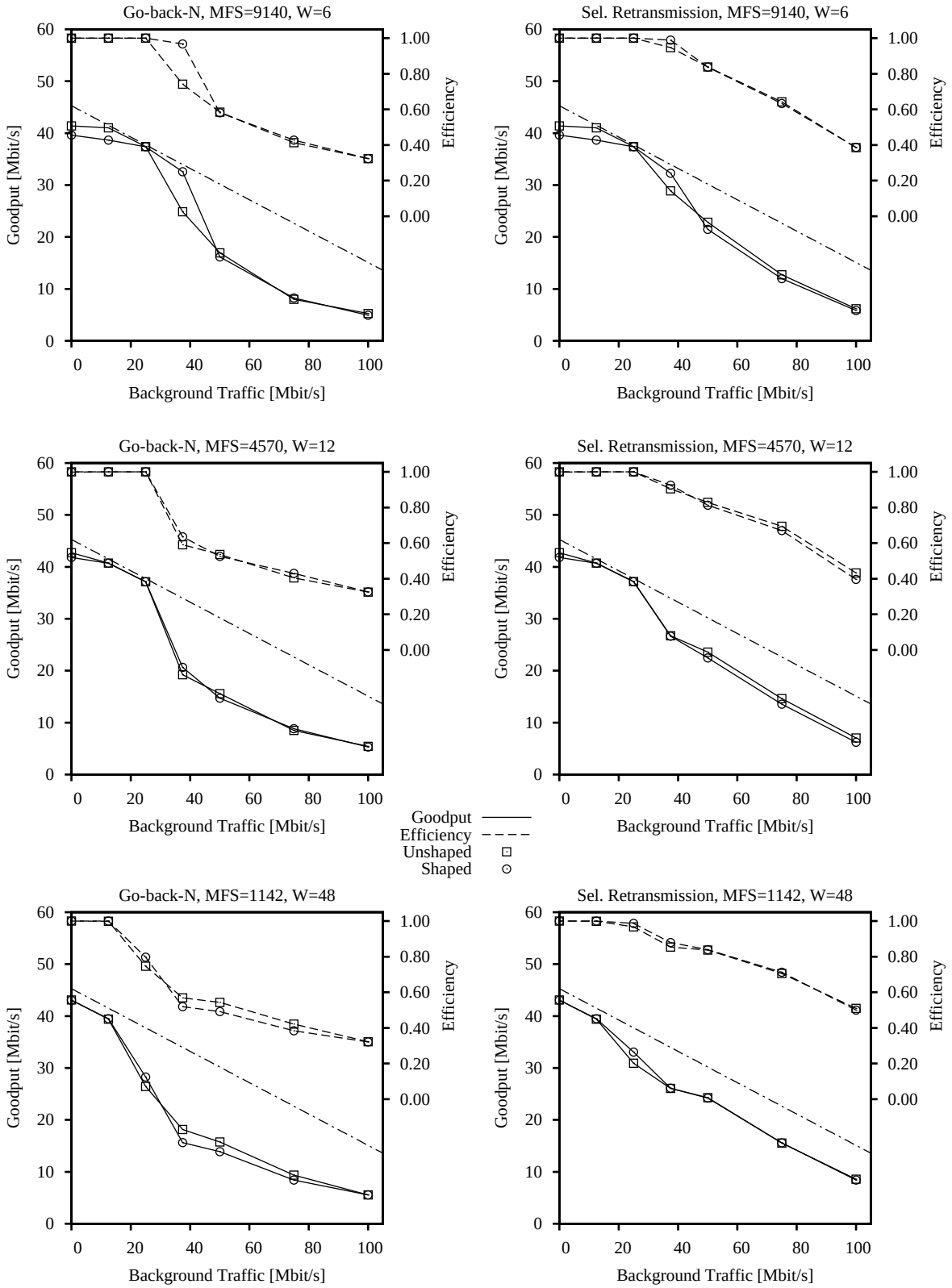

Figure 4: Performance of 3 XTP connections on the 1000 km bottleneck topology, as a function of the background traffic load, the maximum window, and segment sizes; ACKs are requested for each frame; go-back-N in the left colums and selective retransmission in the right column

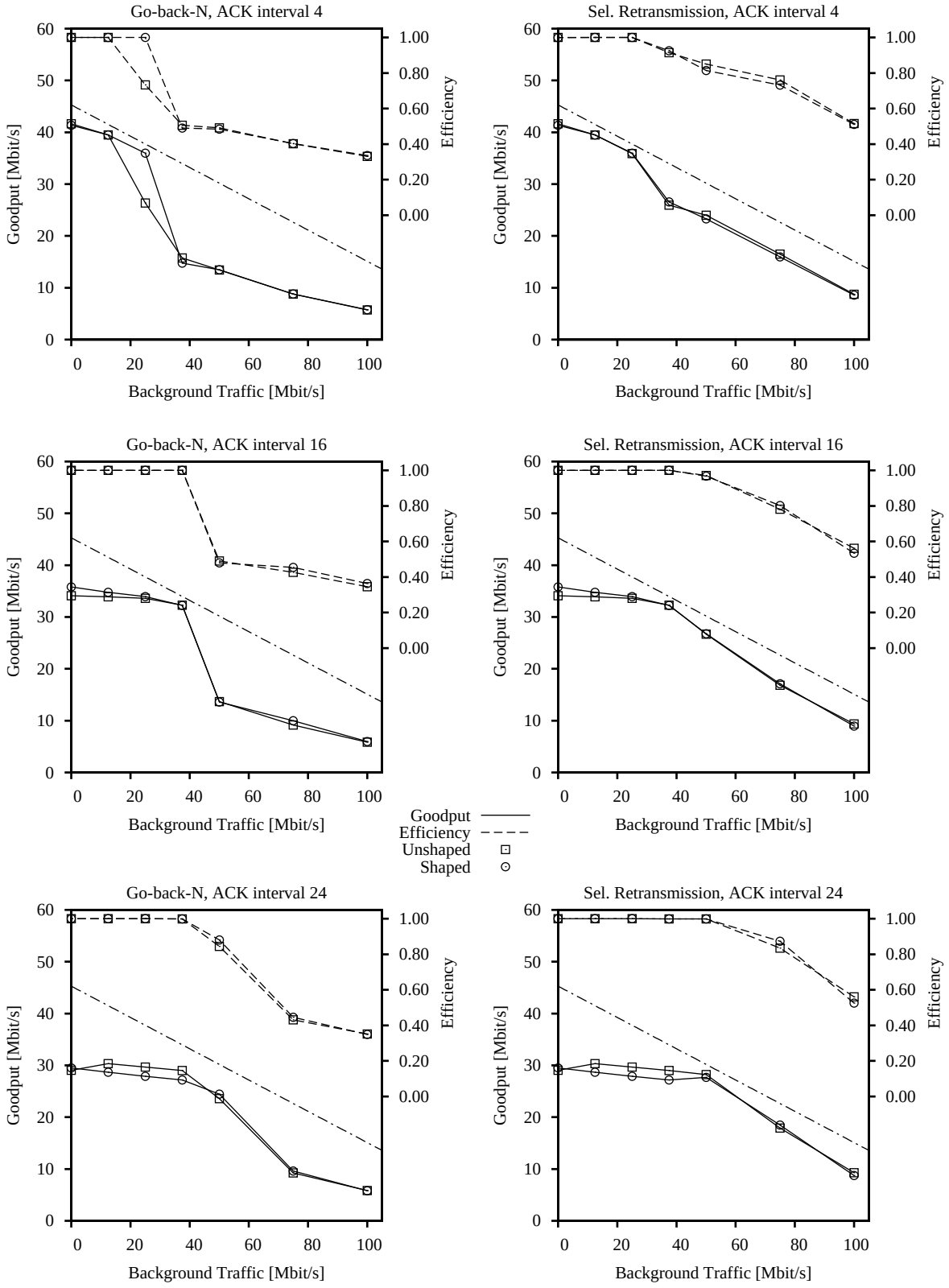

Figure 5: Performance of 3 XTP connections on the 1000 km bottleneck topology, as a function of the background traffic load, and the ACK request interval

the time spent waiting for ACKs. The increase in the amount of time spent waiting for ACKs is beneficial, since it is reflected in a decrease of the average load imposed on the network, thus avoiding congestion. Moreover, it is unlikely that no control packet arrives before the WTIMER expires, and thus synchronizing handshakes can be avoided.

When the background traffic increases, the buffer in the ATM switch cannot store a full output buffer of data from each XTP source, thus some cells carrying XTP data frames are lost. As soon as one data frame is received out of order, the receiver sends back a NAK identifying lost data. The transmitter reacts to the NAK by immediately retransmitting lost data; this further loads the network, especially when the go-back-N scheme is applied. The more congested is the network, the larger is the amount of lost data, hence the time the transmitter spends waiting for ACKs or NAKs; as this time grows, WTIMER eventually expires, and the transmitter at last stops overloading the network for a RTT (needed to complete the synchronizing handshake). This phenomenon generates the sharp drop of goodput and efficiency that appears in the left column plots of Figure 4. When selective retransmission is used, the decrease of goodput and efficiency is smoother, as shown by the plots in the right column of Figure 4.

Thus, the low performance of XTP is mainly due to the inadequacy of the XTP mechanisms to cope with congestion: XTP is not able to promptly react to congestion because it slows down only after having sent a full output buffer of data; this stems from having RTIMER always expiring sooner than WTIMER, and the latter being overwritten.

Figure 5 shows the effect of increasing the ACK request interval on long XTP connections with short data frames (1142 bytes of payload). Growing the ACK request interval, the background traffic range over which XTP achieves efficiency 1 grows, but the goodput at low loads decreases. The transmitter behavior is almost the same as when an ACK is requested for each frame, i.e., a full output buffer of data is transmitted before stopping and waiting for control packets. At this point the XTP transmitter waits for a time that grows with the number of data frames between ACK requests. Waiting reduces the goodput, even though no data frames are lost, because the transmitter buffer is roughly equal to the connection pipe when the network is not loaded.

It can be argued that increasing the transmission buffer would balance the above effect. However if the transmission buffer is bigger, as XTP overwrites WTIMER and never stops transmitting, the network becomes completely overloaded and the system performs very poorly. We do not presents results here for the sake of brevity.

4.3 TCP and XTP over a 10 km bottleneck topology

We have up to now considered the behavior of TCP and XTP separately over a network with a simple topology, but with rather long connections. We now concentrate on the same simple topology, but with very short connections: just 10 km of equivalent length.

The behavior of TCP in this case is far less interesting: if the maximum window dimension W_b is large, TCP connections perform very badly, and connections are sometimes even forced to close down; if, on the other hand, W_b is reduced to a suitable size (e.g. $MSS = 1142$ and $W = 2$), TCP performs very well, grabbing all the available bandwith with efficiency 1, but its behavior is basically the one of a stop-and-wait protocol. Once again we do not present here the numerical results for the sake of brevity.

The behavior of XTP is instead far more interesting. Let's start from the results reported in Fig. 6, where $MSS = 9140$ and $W = 6$, considering only the curve relative to the unshaped connections. Long data frames (9140 bytes) imply a long RTIMER, and both RTT and WTIMER are much shorter than this value. When the network is not overloaded, the ACK for a data frame arrives back to the transmitter before both RTIMER and WTIMER have expired. As the background traffic grows, the network becomes overloaded and the round-trip-time increases; if the cells of a frame arrive at the switch when the buffer is full enough, the frame is delayed, and its ACK is not received by the transmitter before WTIMER expires. A synchronizing handshake must then be started, thus reducing the load on the network. This mechanism reduces the load offered by the XTP associations, but avoids the network congestion. When the overload becomes high (background traffic load higher than 50 Mbit/s), RTT grows quite high, and WTIMER becomes larger than RTIMER; WTIMER is overwritten by new ACK requests and the same phenomenon described for the 1000 km network yields a sharp drop of both goodput and efficiency. It must be noted that in this scenario the difference between the go-back-N and the selective retransmission schemes are not significant, indicating that cell losses have greater impact than retransmissions.

Two considerations arise from the above results: i) when the network is very short both TCP and XTP, in order to obtain good performances, must behave similarly to a stop-and-wait protocol; ii) the conditions that allow XTP to perform well in short networks are very critical.

How critical are the conditions that allow XTP to perform well can be immediately be seen by the fact that, in the same scenario, shaping the connections is enough to completely spoil the performance (see Fig. 6): the increase in RTT due to the slower transmission rate makes WTIMER larger than RTIMER even when the network is lightly loaded.

A situation identical to the one arising when shaping is applied can be due also to shorter frame lengths, since the value of RTIMER is directly proportional to the frame length, while RTT is not. Numerical results are

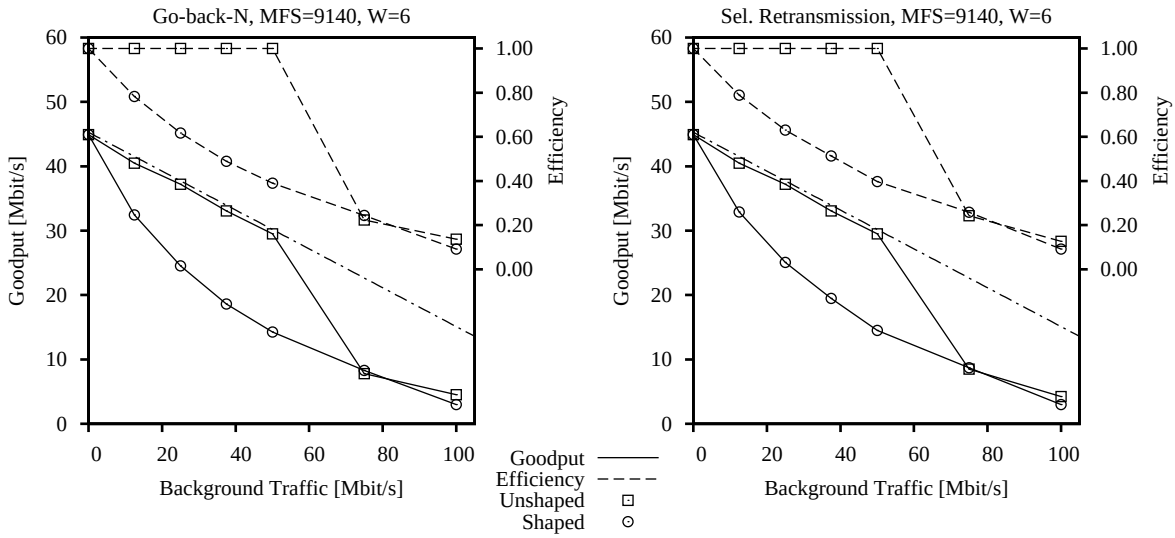

Figure 6: Performance of 3 XTP connections on the 10 km bottleneck topology, as a function of the background traffic load; go-back-N in the left plot and selective retransmission in the right plot

similar to those obtained with the shaped connections and hence they are not shown for the sake of conciseness.

4.4 TCP and XTP over a 1000 km parking lot topology

Up to now we always considered the behavior of TCP and XTP over the simple bottleneck topology depicted in Fig. 1. The next set of numerical results instead aims at determining how the performances change when we consider the more complex parking lot topology, shown in Fig. 2.

In this case the only segment size we consider is 1142 bytes, both for TCP and XTP, and we assume the maximum TCP window dimension to be fixed to 48 segments. The same dimension is used for the XTP transmission buffer; moreover, in the case of XTP we assume that an ACK is requested for each transmitted frame.

Numerical results for TCP are presented in Fig. 7, showing in the left plot the curves obtained without shaping, and in the right plot the curves obtained with TCP sources shaped at 50 Mbit/s.

The curves for XTP are presented in Fig. 8, where the two left plots contain results for the case of no shaping, and the two plots on the right contain the curves for the case of XTP sources shaped at 50 Mbit/s. The two upper plots refer to the go-back-N retransmission scheme, whereas the two lower plots refer to the selective retransmission scheme.

The first general observation that can be drawn is that TCP achieves a somewhat better performance than XTP, specially as far as efficiency is concerned. Indeed, while in Fig. 7 we seldom observe average efficiencies lower than 0.8, in the case of XTP (Fig. 8) the average efficiency of the three associations can drop as low as 0.4. This is essentially due to the lack of congestion avoidance mechanisms in XTP (or at least to their limitations). Indeed, while a TCP source can adapt its transmission rate in response to congestion, and even to increases in the estimated round trip time, this is only partly possible for XTP sources.

The second most evident result is that in the case of TCP with no shaping, a significant unfairness exists among the three connections (see the left plot of Fig. 7). This behavior requires a somewhat elaborate explanation, and is rooted in the fact that the simulation software assumes that cells are brought from the input channels to the output buffers in a random order, selecting input cells according to a uniform distribution encompassing all input channels. The TCP connection that achieves the lowest goodput and efficiency values in Fig. 7 is the one entering the parking lot topology at switch number 3 (see Fig. 2). This switch has five input channels that carry a heavy load: the first three channels are those connected to the three background traffic sources that are directly connected to this switch; the fourth channel is the one arriving from switch number 2, that carries the first two TCP connections, as well as three backgroud traffic flows; the fifth channel is the one arriving from the TCP source number 3, that carries only the cell flow originating from this source. When the node is congested, it is likely that all (or at least several) of these channels carry burts of cells at 150 Mbit/s. If this is the case, due to the random selection, a cell of the TCP source 3 is lost with probability $1/n$, being n the number of channels carrying a burst. Instead, since the flows of the TCP sources 1 and 2 are already multiplexed on one channel, the probability of cell loss for each one of these sources is only $1/(2n)$. Note that when the TCP traffic is shaped, the presence of bursts is almost eliminated, and the three connections achieve the same performance (see the right plot of Fig. 7).

A further interesting comment concerning the performance of the TCP connections can be made comparing

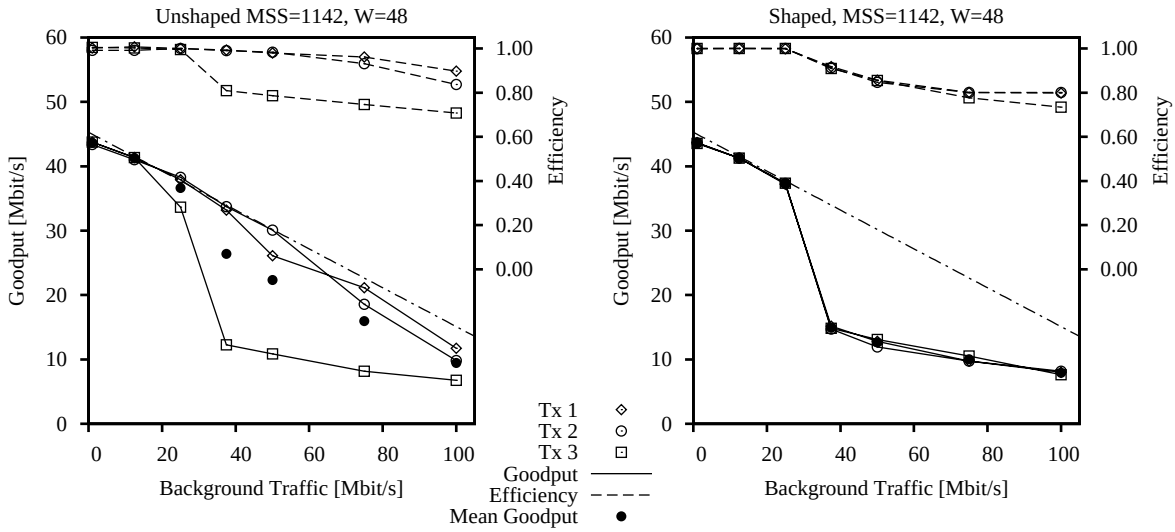

Figure 7: Performance of 3 TCP connections on the 1000 km parking lot topology as a function of the background traffic load

the right plot of Fig. 7 with the curve referring to shaping in the lower left plot in Fig. 3. The comparison allows us to see that the performance achived in the two topologies is almost identical. This is reasonable, if we consider that the only overloaded channel in the parking lot topology is the one connecting the switches number 3 and 4, so that only one channel is actually driving the whole system performance, similarly to what happens in the bottleneck topology.

Coming now to XTP, we observe once more that shaping has a lesser impact on performance, because, as we already noted, the main characteristic of XTP is that of being rate-controlled at the frame level, so that a further rate control at the cell level has only a minor influence on performance. Moreover, in this case we also see that the differences between the go-back-N and the selective retransmission algorithms are marginal. This is an indication that the driving effect is the loss of cells within the buffer in the switch number 3, rather than the retransmission of a large number of frames that were already correctly received.

5 Conclusions

The performance of two transport protocols, TCP and XTP, when used to support bulk transfers over ATM networks, was investigated through simulation, using CLASS, a simulation tool for the study of ATM networks at the cell and burst levels.

The choice of the two transport protocols to be investigated was dictated by the fact that TCP is the de-facto standard for data applications in both LANs and WANs, but it was not specifically designed for a high-speed networking environment, whereas XTP is a significant representative of the category of light-weight transport protocols expressly designed for high bandwidth-delay product networks. Moreover, TCP is a window-based protocol, whereas XTP is rate-based; while TCP contains algorithms for congestion avoidance, XTP includes no such feature.

Numerical results were derived for two different topologies, and for a number of different sets of system parameters. The most general indication that can be obtained from the numerical results is that the performance differences between the two protocols are not striking. This is quite a good point in favor of TCP, whose diffusion is today extraordinarily wide.

It thus seems that the main motivation for a possible replacement of TCP in very high-speed networks should not be rooted in performance issues, but might rather be based on the need for simpler (hence less time consuming) algorithms at the transmitter and the receiver.

References

[1] XTP Forum, *Xpress Transport Protocol Specification Revision 4.0*, Technical Report XTP 95-20, March 1995

[2] Protocol Engine Inc. *Xpress Transfer Protocol Specification Version 3.6*, 1986

[3] ATM Forum/95-0013R8, *ATM Forum Traffic Management Specification*, Version 4.0, October 1995

[4] ITU-TSS Study Group 13, Recommendation I.371, *Traffic Control and Congestion Control in B-ISDN*, Geneve, Switzerland, July 1995

[5] M. Ajmone Marsan, A. Bianco, T.V. Do, L. Jereb, R. Lo Cigno, M. Munafò, *ATM Simulation with CLASS*, "Performance Modeling Tools", Performance Evaluation, 24 1995, pp.137–159

[6] Van Jacobson, *Berkeley TCP evolution from 4.3-tahoe to 4.3-reno*, Eighteenth IETF, Vancouver, BC, Canada, August 1990

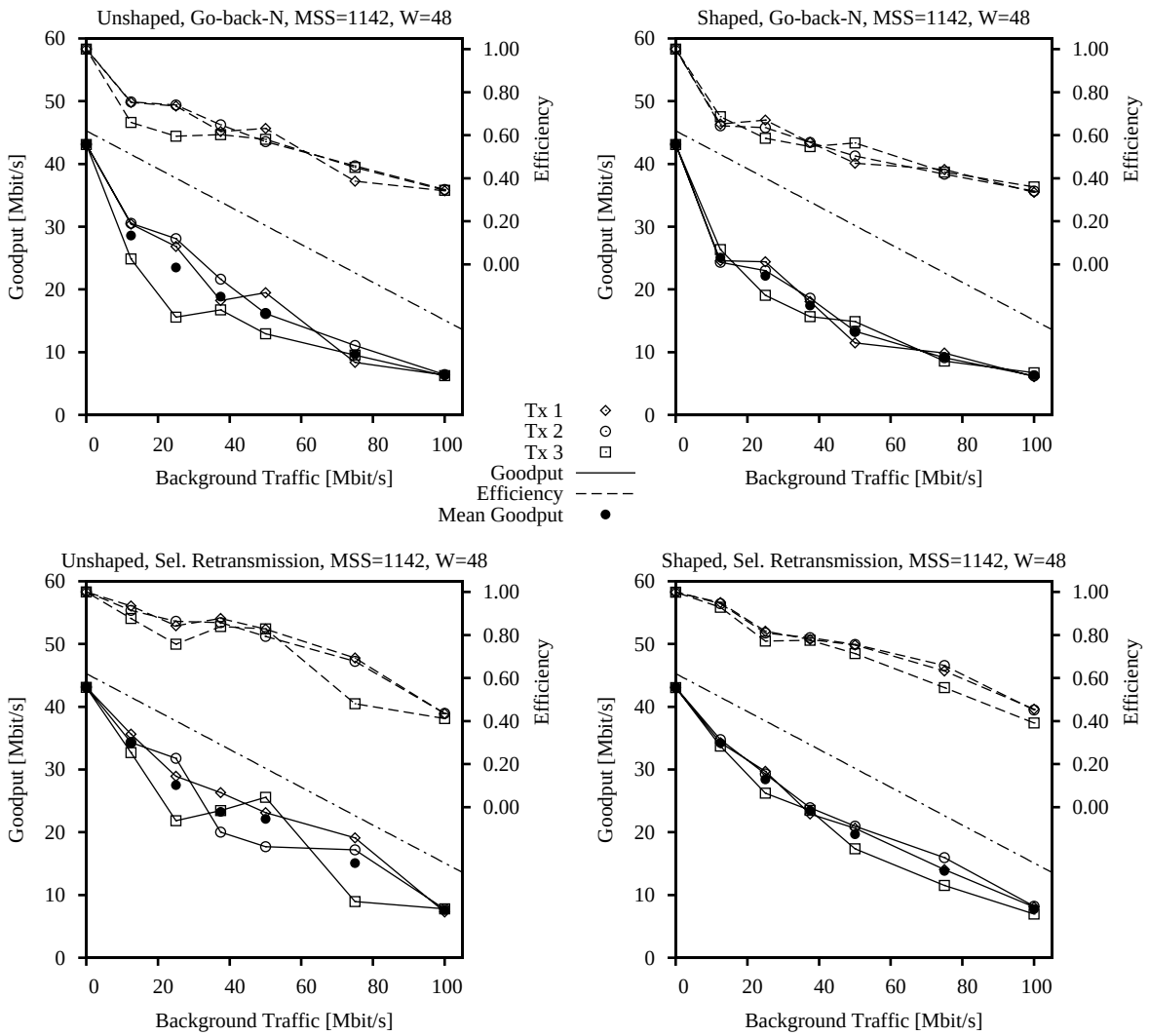

Figure 8: Performance of 3 XTP associations on the 1000 km parking lot topology as a function of the background traffic load

[7] Distributed Systems Research, *SandiaXTP User's Guide Release 1.3*, Sandia National Laboratories, 1995

[8] Van Jacobson, *Congestion Avoidance and Control*, ACM SIGCOMM '88, Stanford, CA, USA, August 1998

[9] *Delta-t Protocol Specification*, Lawrence Livermore Lab., UCID-19293, April 1993

[10] G. L. Chesson. *Datakit Software Architecture*, ICC'79, 1979

[11] L. Zhang D. Clark, M. L. Lambert. *NETBLT: a Bulk Data Transfer Protocol*, (RFC 817), July 1987

[12] D. Cheriton *VMTP: a Protocol for the Next Generation of Communication Systems*, ACM SIGCOMM'86, Stowe, VT, August 1986

[13] A.Romanow, S.Floyd, *Dynamics of TCP Traffic over ATM Networks*, ACM SIGCOMM'94, London, UK, September 1994

[14] A.Bianco, *Performance of the TCP Protocol over ATM Networks*, ICCCN'94, San Francisco, CA, USA, September 1994

[15] M. Ajmone Marsan, A. Bianco, R.Lo Cigno, M.Munafò, *Shaping TCP Traffic in ATM Networks*, IEEE ICT'95, Bali, Indonesia, April 1995