Simulation Analysis of TCP and XTP File Transfers in 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*
[2] *Dipartimento di Automatica e Informatica, Politecnico di Torino*
Corso Duca degli Abruzzi 24, 10129 Torino – Italy
email: {ajmone,mbaldi,bianco,locigno,munafo}@polito.it

Abstract

In this paper we study the TCP and XTP transport protocols in light of their possible use for applications running over B-ISDN in a UBR-like context. In particular, we investigate the performance of file transfer applications taking into account both the impact of the ATM layer protocols and the features of the adopted transport protocol. The investigation is performed by simulation, using CLASS, a simulation tool for the study of ATM networks at the cell and burst levels, and considering quite a simple topology, known as the bottleneck topology, where TCP or XTP connections share a bottleneck link with some interfering background traffic, that is taken to have either Poisson or ON-OFF characteristics. Although the two transport protocols have quite different features, their performances are rather similar in almost all the considered scenarios. However, when XTP and TCP connections are mixed within the same network, the TCP performance is dramatically worse than that achieved by XTP, due to a more aggressive policy of XTP in the access to the transmission medium: a behavior that should raise some concern about the performance achievable when mixing different transport protocols on the same network.

1 INTRODUCTION

Much research is in progress to investigate traffic and performance aspects of ATM networks, so as to correctly engineer B-ISDN services. While recently a lot of attention has been devoted to the performance of TCP [Transmission Control Protocol – see Van Jacobson (1990)] when used over ATM networks [see Romanow and Floyd (1994), Bianco (1994), Ajmone Marsan et al. (1996), Lakshman et al. (1996), Li et al. (1996)], to our knowledge no study was yet published about the

*This work was supported in part by a research contract between CSELT and Politecnico di Torino, in part by the EC through the Copernicus Project 1463 ATMIN, and in part by the Italian Ministry for University and Research.

performance of XTP [Xpress Transport Protocol – see XTP Forum (1995)] over ATM networks. The comparison of TCP and XTP is an interesting subject of research when we consider data transfer applications, where the performance seen by the end-user not only depends on the ATM and the AAL (ATM Adaptation Layer) layers, but it is also greatly influenced by the behaviour of the transport protocols.

The interest in XTP is justified by the fact that XTP is a rate-based transport protocol designed for high-speed networks, in contrast with the window-based approach adopted by TCP. This important difference could have a significant impact on the performance figures obtained by the two protocols. A further major 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 objective of our study is to determine whether the two transport protocols offer significantly different performance when used in the context of ATM networks. 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 in ATM Forum (1995) and the "fast reservation protocol" named ABT (ATM Block Transfer) proposed in ITU-TSS (1995). The main reason is that we are interested in looking at the basic behaviors induced on transport layer protocols by the segmentation process required at the ATM layer; introducing more sophisticated control techniques significantly improves the performance figures, but it could mask some of the phenomena we are interested in. Our scenario bears resemblance with the UBR service capability, although we do not consider here, for simplicity, techniques like selective cell discarding which are widely accepted in the UBR context.

In this paper we adopt a simulation approach to study the performance that can be achieved by using either TCP or XTP for file transfer applications over ATM networks. We focus on the so-called bottleneck topology, where file transfer connections share a bottleneck link with a variable amount of background traffic. We first present results in a simpler scenario with either only TCP or only XTP connections, sharing the bottleneck link with Poisson background traffic, to highlight the main characteristics of the two protocols. Then we concentrate on a more bursty ON-OFF background traffic; later we modify the network span by considering variable length connections to assess the ability of the two protocols to deal with connections with different round-trip times. Finally, we consider a mixed scenario, in which some file transfer connections are carried by the TCP protocol, while others rely on XTP, to discuss the interaction between the two protocols when sharing network resources.

2 THE SIMULATION MODEL

The results presented in this paper are obtained with the CLASS ATM network simulator [see Ajmone Marsan et al. (1995), Lo Cigno and Munafò (1996)] that was developed at the Dipartimento di Elettronica 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.

In order to keep our simulation models as close as possible to the original behaviour of the protocols, instead of developing an abstract model for TCP and XTP, we adapted a simplified version of two distributed releases of the TCP and XTP protocols to run on top of the AAL5 layer within CLASS. We chose the officially distributed BSD 4.3-reno release for TCP [see Van Jacobson (1990)] and the SandiaXTP 1.3 C++ code [see Distributed Systems Research (1994)] released by Sandia National Laboratories.

We do not outline here the main characteristics of the TCP-reno congestion control algorithm, since the protocol is well-known [see Stevens (1994)], but we discuss the modifications required to adapt its code to the ATM environment.

Our implementation includes all the important features of TCP reno, with the exception of the delayed ACK and the selective ACK options. Although the selective ACK option could have a remarkable influence on the simulation results, we do not include this option in our model, 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, and it is disabled.

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 a parameter in our implementation.

In present TCP implementations, the *granularity* of timers (the minimum value a timeout can assume) is rather coarse (roughly 500 ms). This granularity impacts the timeout setting and the RTT (Round Trip Time) estimation, and it must be reduced in order to allow the protocol to react fast on high bandwidth-delay product networks. 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 network resources has to be achieved. We left the granularity as a parameter in our model and we followed the above guidelines to set its value in all our simulation experiments.

The Xpress Transport Protocol (XTP) was specifically designed aiming at high bandwidth-delay product and low error-rate networks. Since XTP is not a widely used protocol like TCP, we briefly outline here the main featuress of XTP version 4.0, with particular emphasis on the aspects that are most relevant from the modeling point of view.

XTP supports both flow and rate control. Flow control is regulated by the receiver, and it is based on the amount of space available in its buffer. Rate control, one of the main feature of XTP, is enforced by the transmitter, through the use of the timer `RTIMER`, which is used to schedule the transmission of frames (the term used to refer to packets in the XTP specifications). An XTP source can initially transmit up to `burst` bytes; then it has to wait until `RTIMER` expires before being again allowed to transmit up to `burst` bytes, and so on; `burst`, the number of bytes that can be transmitted between two timer expirations is a simulation parameter.

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, as allowed by the standard, and as we did in our simulation experiments, does not guarantee that rate control remains the only mechanism that regulates transmission speed. In fact, 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 can be lost, and the output buffer becomes full because ACKs 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 kind of flow control mechanism.

XTP retransmits unacknowledged frames based on either a *Go-Back-N* or a *Selective Retransmission* scheme. We will refer to these two versions of the protocol as XTP GBN and XTP SR in the remainder of the paper.

The XTP specification does not state when ACKs and NAKs are to be sent by the receiver. ACKs (NAKs) are requested by the transmitter, and the receiver sends back ACKs (NAKs) as soon as it gets a request. Moreover, the transmitter can request the receiver 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 initialized to a value based on an estimation of the round trip time similar to the one implemented in TCP.

If the ACK does not reach the transmitter before the expiration of `WTIMER`, the transmission of data is suspended and a *synchronizing handshake* is started: the transmitter sends a request for an ACK that contains the status of the transmitter and demands an ACK containing the actual status of the receiver. If the reply is not received before the timer expires, a new request 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 Sandia 1.3 C++ implementation adopts this option. If an ACK is requested before the previous one has been received (and before `WTIMER` has expired), the transmitter re-initializes `WTIMER`; thus, previous ACK requests cannot trigger a synchronizing handshake any more.

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 behaviours 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, for instance, from a very long file

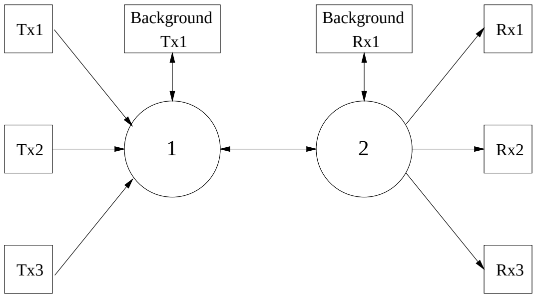

Figure 1 The bottleneck topology.

transfer or from an application requiring TCP or XTP to work in streaming mode. We assume that sources always have data to send; data are transmitted as fast as allowed by the control mechanism of the transport protocol. Each TCP and XTP connection† is mapped onto an ATM virtual circuit.

The topology we concentrate on in this paper is reported in Fig. 1 and we refer to this topology as the "bottleneck" topology in the remainder of the paper. 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 connections under study.

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) connections is assumed to be unidirectional: transmitters send data segments, and TCP (XTP) receivers return only ACKs; TCP ACKs fit in one cell, while XTP ACKs are two cells long (ACKs) or more (NAKs). Losses are avoided both in the user transmission buffer and at the receiver: they can occur only within ATM switches.

We take as simulation parameters the maximum TCP window size W and 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 IP over ATM. With the 32 bytes of the XTP header and the AAL5 overheads, XTP sources generate AAL5 CS PDUs of 9180 bytes. We will use the term PDU to identify either TCP segments or XTP frames and the acronym MPS to identify the Maximum PDU Size in our simulation runs. The TCP code was slightly modified so that only PDUs of size MPS are transmitted. As additional MPS size we use the value 1142 which is 1/8 of the reference MPS measured in ATM cells when headers are included. Note that actual implementations of TCP use a very small MPS for inter–LAN traffic, typically 512 bytes; this

† More precisely we should use the term *association* for XTP; we will refer to XTP associations as XTP connections for simplicity, adopting the TCP terminology

limitation is mainly due to the available IP routers, and it will probably disappear in future implementations.

XTP sources are rate controlled and generate traffic at an average rate of 50 Mbit/s. The burst length `burst` of the rate controller is set to one frame because high burstiness is usually critical in ATM networks. Thus, our XTP sources transmit one PDU and then wait for `RTIMER` to expire.

The XTP flow control is disabled, but the size of the transmission buffer is set to the same value of the TCP window. Our XTP transmitter implementation requests ACKs at regular intervals, and in the simulation presented in this paper an ACK is requested for each transmitted PDU, to be consistent with the TCP behaviour. 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.

Simulations were run both with and without a traffic shaper operating at the ATM level at 50 Mbit/s. The shaping device operates according to a modification of the Generic Cell Rate Algorithm described in ITU-TSS (1995).

4 NUMERICAL RESULTS

The main performance figures we present in this study are the *goodput*, i.e., the average effective amount of user data that are transferred from the transmitter to the receiver, discarding all overheads, faulty PDUs, 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. If not otherwise stated, the performance figures are averaged over the 3 connections: the actual performances of individual connections lie within 1–2% of the average.

We consider two types of background traffic: Poisson and ON-OFF. The Poisson 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. The ON-OFF background traffic is generated by a single ON-OFF source, sending a CBR cell flow during ON periods, with geometrically distributed ON and OFF period lengths. We consider two possible values for the average duration of the ON and OFF periods, namely 20000 and 10000 cells in the first case (we will refer to this background as fast ON-OFF) and 200000 and 100000 cells in the second one (slow ON-OFF). On average the ON-OFF background traffic source transmits for 2/3 of the simulation time; the transmission rate during the ON period is set to the value that produces the required average background traffic load. This type of traffic, which could be considered quite unrealistic, allows us to discuss phenomena and behaviours typical of transient phases using our time-averaged performance indices.

The following subsections present first the results obtained using 3 TCP or 3 XTP connections,

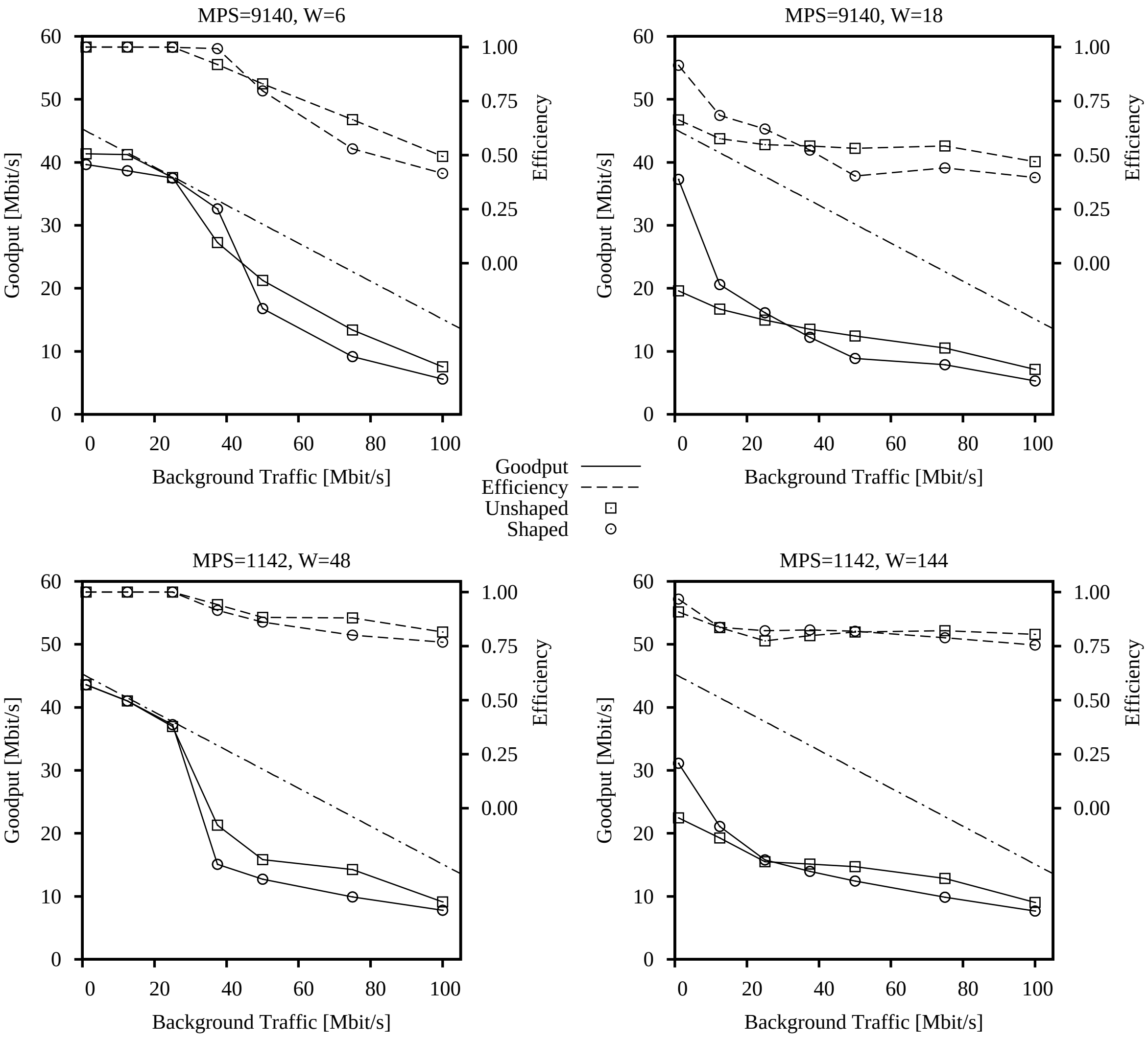

Figure 2 Aggregate average goodput and efficiency for TCP connections in the 1000 km bottleneck topology, as a function of the Poisson background traffic load, the maximum window W and MPS.

when a Poisson background traffic is considered. Then we discuss the situation when an ON-OFF background traffic is adopted. Later, variable length connections are studied, and finally a mixed XTP-TCP scenario is considered.

4.1 TCP connections with Poisson background traffic

Although a number of analyses of the performance of TCP connections on ATM networks already appeared [see Romanow and Floyd (1994), Bianco (1994), Ajmone Marsan et al. (1996), Lakshman et al. (1996), Li et al. (1996)], it can be useful to discuss the impact of such basic parameters as the maximum window and segment size. Fig. 2 reports the aggregate performance for 3 TCP connections with length 1000 km, as a function of the background traffic load. Each plot reports the aggregate goodput (solid lines) and efficiency (dashed lines), 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 (round markers). The plots in the left column refer to the case of "optimal" window size, when the maximum TCP window size in bytes $W_b = MPS \cdot W$ is set so as to allow each connection to exploit roughly 1/3 of the link bandwidth when no background traffic is present, disregarding queueing delays in the ATM switch buffers. Plots in the right column consider a 3 times larger value for W_b, either accounting for the possibility for each connection to exploit the whole link capacity, or to maintain a high throughput even if the queueing delays increase the RTT estimate. The two rows refer to results obtained with different maximum PDU size in bytes. The upper row refers to the case of MPS = 9140 which is the recommended maximum segment size for IP over ATM networks, while the lower row refers to the case MPS = 1142; 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 optimized, the network resources are very well exploited. The TCP 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 goodput and 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. A larger window implies, in fact, the possibility of transmitting at higher speed, but also a more aggressive policy in the window size increase algorithm: 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.

The effect of the PDU size on goodput is negligible; however, if we consider efficiency, it is quite clear that the smaller the PDU the better the efficiency, specially when the network is overloaded. Clearly, the PDU size cannot be reduced too much, otherwise the overheads would become too large. It may be worthwhile to stress the importance of efficiency; it must be taken into account that a sharp drop in efficiency 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. As a consequence, the efficiency is an indirect but significant measure of the effect of transport protocol connections on background traffic. In fact the link load ρ can be obtained summing the background load to the TCP or XTP goodpud divided by the efficiency, thus for a give goodput a low efficiency of TCP or XTP implies that the network is more havily loaded and hence the background traffic suffers in terms of higher cell loss and delay jitter.

As a final consideration, we observe that, in this scenario, shaping the TCP traffic does not

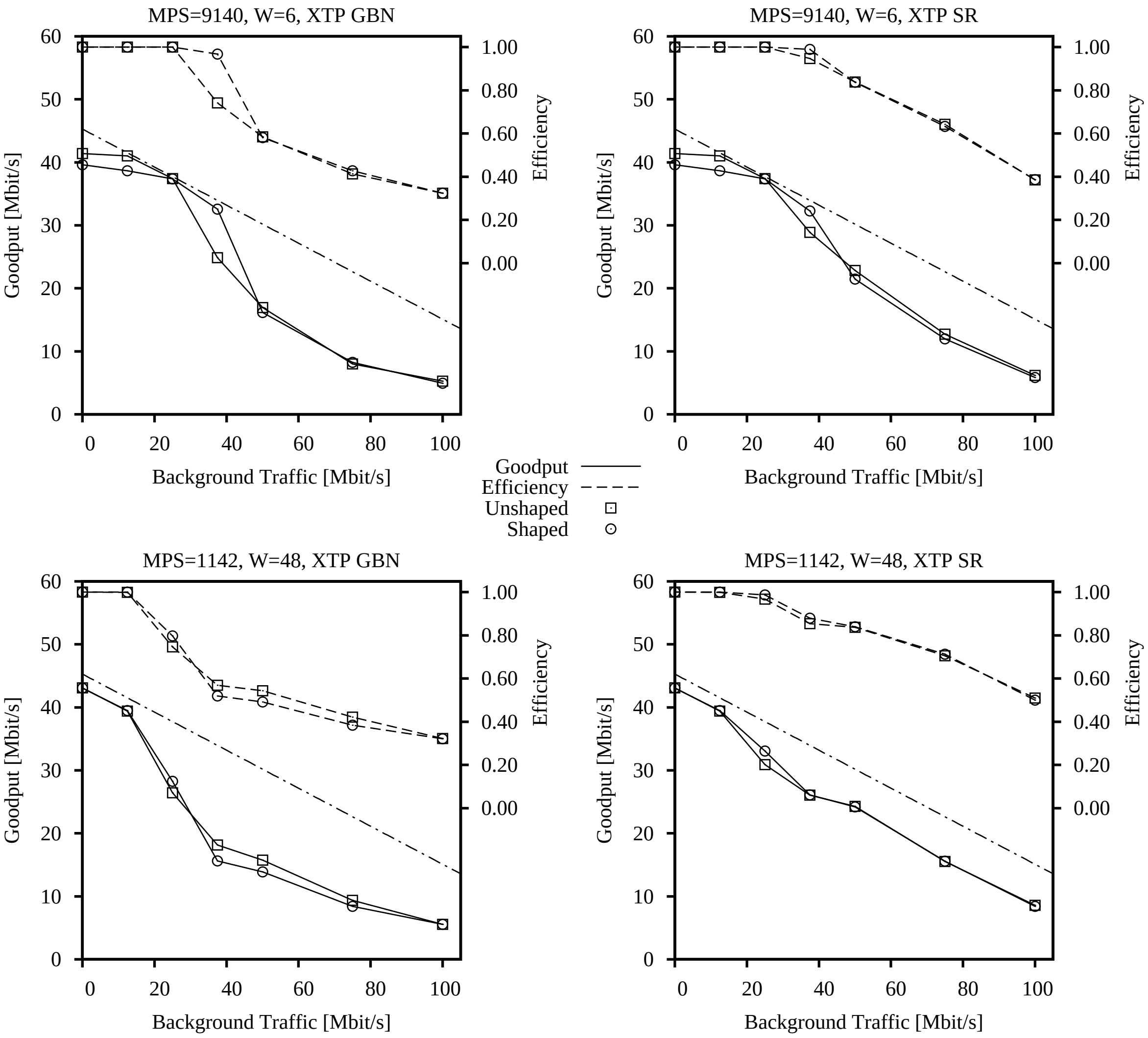

Figure 3 Aggregate average goodput and efficiency for XTP connections in the 1000 km bottleneck topology, as a function of the Poisson background traffic load, the maximum window W and MPS.

significantly alter performance, possibly even having a negative effect. 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, shaping makes it more likely that one of the cells is lost due to the background traffic fluctuations.

4.2 XTP connections with Poisson background traffic

In Fig. 3 we consider a scenario quite similar to the one that was just discussed, but now XTP is the transport protocol. Results refer to a MPS equal to 9140 bytes in the upper row, and 1142 in the lower row. Both the Go-Back-N (left column) and the Selective Retransmission (right column) schemes are studied. Each plot shows curves for both the aggregate average goodput and efficiency as functions of the background traffic load, in the cases of shaped (round marker) and unshaped (square marker) XTP traffic.

First of all, we observe that 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 PDU level; it is thus reasonable that a further rate control at the cell level has a minor impact on performance.

When the background traffic is low, XTP connections succeed in exploiting most of the available capacity (indicated by the dot-dashed line) without any PDU 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, i.e. the synchronizing handshake, cannot work properly with the given values of `RTIMER` and RTT. Indeed, XTP sources transmit regularly-spaced PDUs at the average rate of 50 Mbit/s; when a synchronizing handshake starts, the transmitter stops sending data PDU, thus reducing the load on the network. In our scenario, the RTT is very large compared to `RTIMER`, i.e. the time between two consecutive ACK requests. 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 PDU 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, ACKs start reaching the transmitter.

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 used 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 PDUs are lost), but RTT increases, and so does 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 ACK reaches the transmitter before the `WTIMER` expires, and thus synchronizing handshakes can be avoided.

As the background traffic increases, the buffer in the ATM switch cannot store a full output buffer of data from each XTP source, and some PDUs are lost. As soon as one PDU 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 XTP GBN is considered. The more congested the network, the larger the amount of lost data; thus the receiver does not receive PDUs and cannot send back ACKs or NAKs. When the sender buffer is full, it stops until `WTIMER` eventually expires and the transmitter at last stops overloading the network for a RTT (needed to complete the synchronizing handshake). Then,

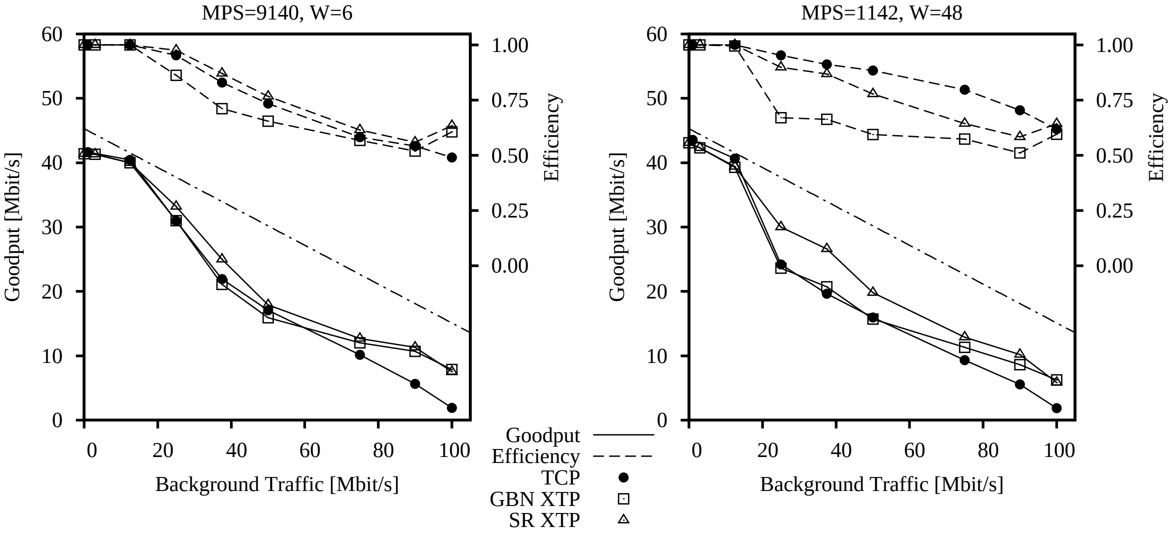

Figure 4 Aggregate average goodput and efficiency for XTP and TCP connections in the 1000 km bottleneck topology, as a function of the fast ON-OFF background traffic load, the maximum window W and PDU MPS size.

the same sequence takes place again; this phenomenon generates the sharp drop of goodput and efficiency that appears in the left column plots of Fig. 3. When XTP SR is analyzed, the decrease of goodput and efficiency is smoother, as shown by the plots in the right column of Fig. 3.

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.

4.3 Comparison between TCP and XTP with Poisson background traffic

The differences between the TCP and the XTP connections behaviour is not striking in this scenario. The TCP performance is significantly influenced by the choice of the window size; when an almost optimal window is used, the goodput obtained by TCP is slightly better than the corresponding figures for XTP GBN. The efficiency of TCP connections is always much better than the one of XTP, due to the adaptive congestion control algorithm. Increasing the MPS significantly worsen the efficiency for TCP, while the goodput behaviour is not dramatically influenced; only a slight increase for some values of background load can be noticed for bigger MPS. Increasing the MPS has a beneficial effect for the XTP goodput. As expected, XTP SR behaves better than both TCP and XTP GBN. Finally, shaping at the ATM layer has a marginal effect on both protocols.

Simulations were run also in the case of a 10 km network. We do not discuss the results in this paper, but we outline here the main conclusions that can be drawn from them.

The first observation is that, when the network is very short, in order to obtain good perfor-

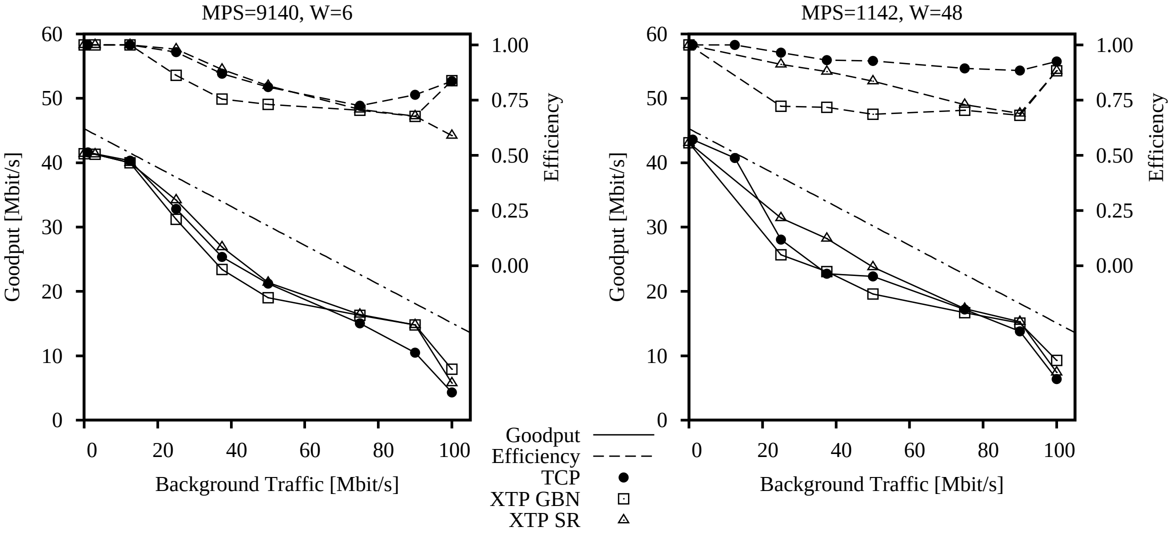

Figure 5 Aggregate average goodput and efficiency for XTP and TCP connections in the 1000 km bottleneck topology, as a function of the slow ON-OFF background traffic load, the maximum window W and MPS.

mances, both TCP and XTP must behave similarly to a stop-and-wait protocol, i.e. a very small window is required to obtain a significant goodput with acceptable efficiency. XTP connections can obtain a very high goodput and efficiency thanks to a particular mechanism of synchronization between the `RTIMER` and `WTIMER` due to the very short network span. The phenomenon is the following: `WTIMER` expires much earlier that `RTIMER`, hence if a frame is lost or delayed too much, a synchronization handshake starts and finishes before (or soon after) `RTIMER` expires, so that the transmission is basically not affected by the frame loss. This peculiar behaviour is very critical and can be easily disrupted, so that we can conclude that also in this scenario the two protocols behave quite similarly.

4.4 TCP and XTP connections with ON-OFF background traffic

We concentrate now on the bottleneck topology with either 3 TCP or 3 XTP connections sharing the bottleneck link with ON-OFF background traffic. In Fig. 4 (5) we present results when the background traffic is generated by a fast (slow) ON-OFF source. Goodput and efficiency averaged over the three connections are reported for TCP, XTP GBN and XTP SR, using round, square and triangle markers, respectively.

We first observe that, as expected, XTP SR behaves slightly better than both TCP and XTP GBN, even if, when MPS=9140, the difference is minimal.

TCP always exhibits better efficiency than XTP, and comparable goodput performance, except when the background load is very high with fast ON-OFF background; in this case, the slow increase control algorithm penalizes TCP connections with respect to XTP.

When comparing the slow and fast ON-OFF scenarios, it can be observed that the slow ON-OFF scenario always allows connections to obtain better performances, since the connections have longer periods in which the background traffic is regular. When the background is switched on and off more rapidly, the transport protocol control mechanisms must cope more frequently with the variable network load; during these transient phases, connections are not able to fully exploit the available bandwidth and the obtained goodput is reduced. TCP benefits of this effect at high load, when the slow ON-OFF background generators are considered, since its congestion control algorithm has enough time to increase the transmission window to a value that allows connections to exploit the available bandwidth; performance figures of TCP in this case are almost identical to those of XTP.

In general, when the ON-OFF background traffic is considered, it can be observed that the goodput is higher than in the Poisson background traffic case, since during the OFF period connections are able to grab the full link bandwidth; XTP connection efficiency also benefits of this phenomenon.

Also in this scenario the differences among TCP and XTP are not astonishing, and TCP still provides better efficiency; this implies that its more controlled behaviour has a beneficial impact on the background traffic.

4.5 Variable length connections

We consider here either 3 XTP connections or 3 TCP connections with variable lengths sharing the bottleneck link with Poisson background traffic; connections lengths are 10, 100 and 1000 km (square, round and diamond markers), and the MPS is set to 1142 bytes.

We first discuss the results reported in Fig. 6, referring to 3 XTP connections; left pictures are obtained without shaping at the ATM layer, while a 50Mbit/s shaping is enforced in the pictures in the right column. The upper row refers to XTP GBN, the lower row to XTP SR. The behavior of the curves in the left plots is rather puzzling, with the shortest connection performing very poorly at low loads.

This behavior is due to the `RTIMER` and `WTIMER` synchronization effect observed in very short networks, that in this case slows down the short connection, allowing the longer two to grab more bandwidth. However, if shaping is applied to the connections, the particular synchronization between `RTIMER` and `WTIMER` is destroyed, and the performance is much more predictable, with the longest connection performing poorly due to long waits and more retransmissions.

The bias against longer connections is roughly linear with the RTT estimation, so that the difference between the 100 km and the 1000 km connections is much more evident than the one between the 10 km and the 100 km connections. As always XTP SR performs better than XTP GBN.

Fig. 7 refers to TCP connections. In the top left plot we report results for unshaped TCP connections; all the connections have the same maximum window size equal to 48 segments. We can clearly observe the well known biased behaviour of TCP towards longer connections, due to their higher RTT that forces longer connections to increase their window size at a slower speed. Increasing the background traffic reduces this effect, since queueing delays tend to reduce the differences among the RTT experienced by the connections. In the top right corner this biased effect is smoothed by applying shaping at the ATM layer, mainly because now all the connections

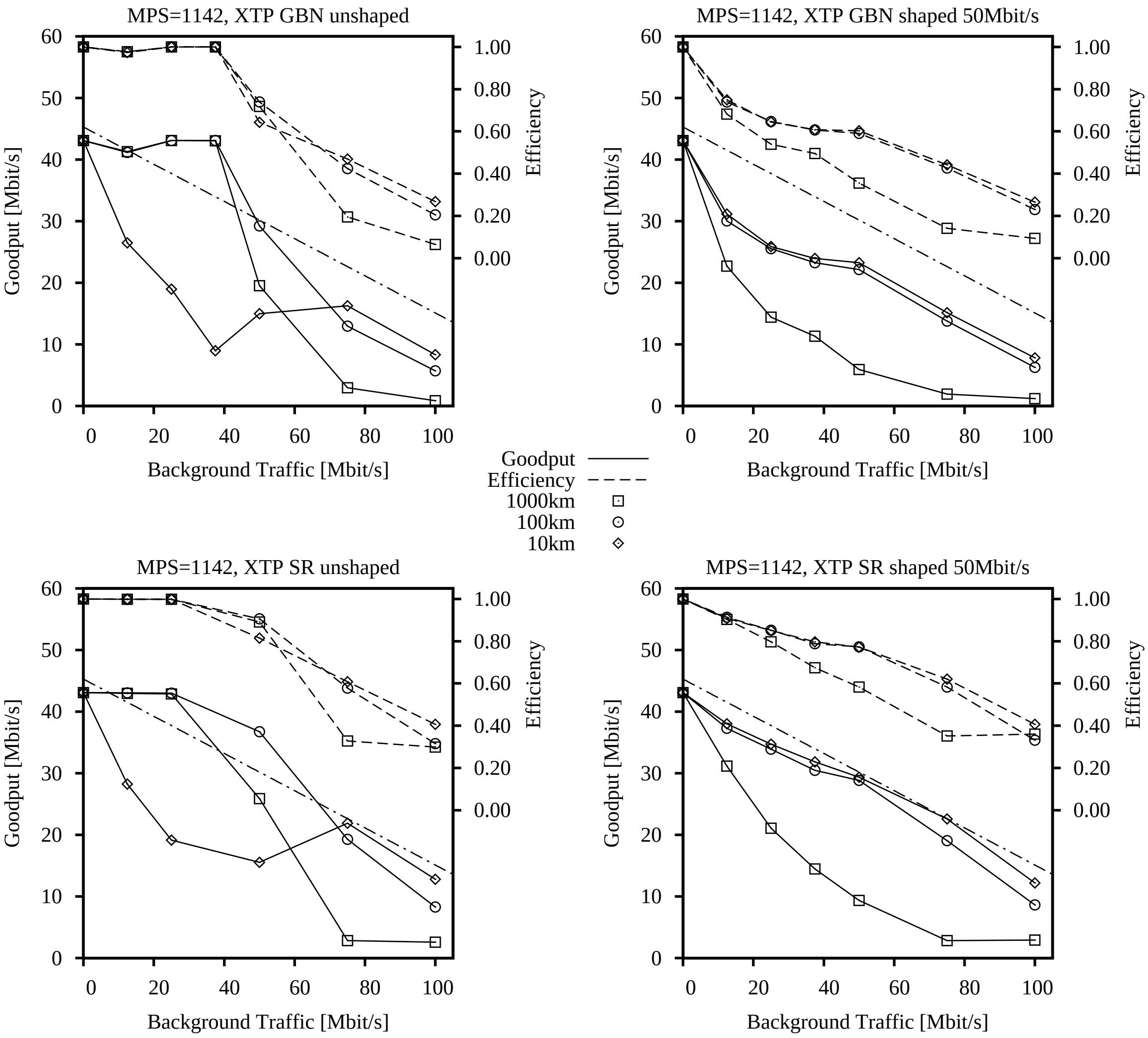

Figure 6 Aggregate average goodput and efficiency for XTP variable length connections, as a function of the Poisson background traffic load.

offer on average 50Mbit/s. Also, shaping requires buffering at the UNI so that the 10 km and 100 km connections experience a similar round trip time delay.

In the remaining four plots in Fig. 7 we try to regulate the maximum window size W so that all connections are close to offer a nominal load of 50Mbit/s. We first set, in the center left picture, the maximum window size to 2 segments for the 10 km connection, to 5 segments for the 100 km connection and to 48 segments for the 1000 km connection. These window sizes limit the capability of shorter connections to grab the available bandwidth; as a result, the longer connection obtains a higher throughput, and the situation is reversed with respect to the previous case. The center right frame refers to the same window sizes, but shaping is applied; again, shaping helps in obtaining better fairness.

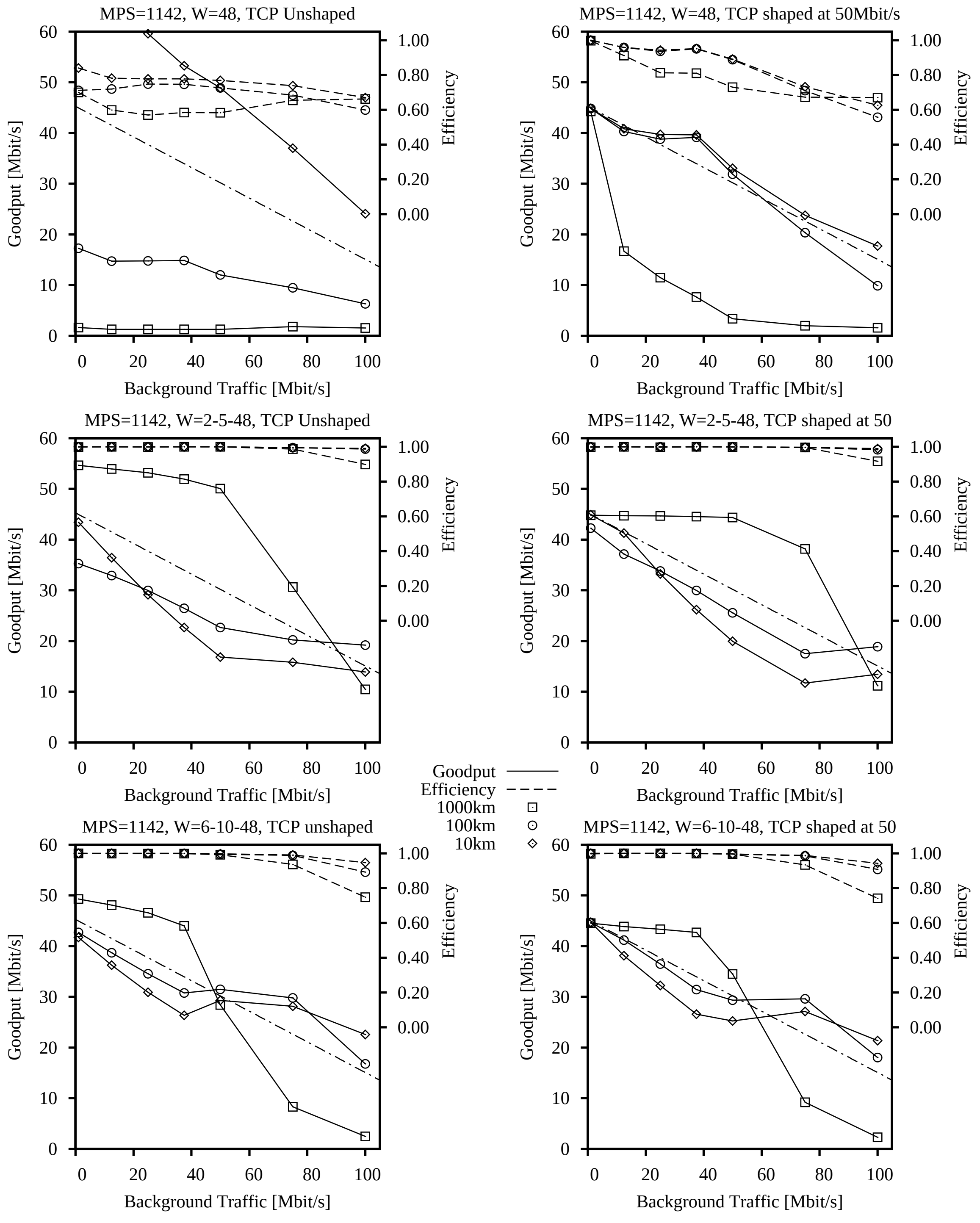

Figure 7 Aggregate average goodput and efficiency for TCP variable length connections, as a function of the Poisson background traffic load and the maximum window size W.

Finally, in the left and right bottom plots, we try to fine tune the window size of shorter connections, that were the limiting factor in the previous scenario. The results suggest that a careful window setting could help in obtaining fairness among connection not controlled at the ATM level, although this fine tuning of the window is probably very difficult to perform in real time, and, in any case, will be useful only for a limited range of network loads. Moreover, any window tuning is useful only for long connections, i.e. not in a LAN environment, in which we are limited by the fact that the window size should at least be equal to two segments; to further reduce the offered load we need to reduce the MPS, Maximum PDU Size.

As a general remark, both TCP and XTP are naturally prone to unfairness when comparing performances obtained by connections with different lengths. Shaping at the ATM level could help in reducing this unfair behaviour. For TCP connections, a careful setting of the window size together with an appropriate shaping at the ATM layer, allow a good fairness to be obtained. It is also significant that while the efficiency is reasonably high for TCP connections, it drastically drops for XTP connections as soon as the background load is increased.

4.6 Mixing TCP and XTP connections

In this last scenario, we simultaneously consider 3 XTP and 3 TCP connections in the 1000 km bottleneck topology with Poisson background traffic. In order to maintain the same network load, we reduced by half most of the connections parameters (like the window size, the XTP transmission rate and so forth) with respect to the previously examined scenario. As a results, also the average aggregate goodput ranges within roughly halved boundaries.

We report in Fig. 8 the aggregate XTP (white markers) and TCP (black markers) efficiency and goodput, averaged over the three connections. Regardless of the MPS and of the chosen XTP version (being it GBN in the first row, and SR in the second row) the main phenomenon that can be observed is that TCP connections obtain a much lower goodput than XTP connections. This is due to the fact that TCP tries to regulate its offered load via the adaptive window mechanism. As soon as TCP detects congestion in the network, it reduces the offered traffic by shrinking its transmission window. XTP connections can easily grab the bandwidth left by the TCP connections since their transmission speed is reduced only via the synchronizing handshake procedure. At the end of the handshake, XTP immediately resumes transmitting at its fixed rate, while TCP connections are forced to maintain their window small, since the network remains congested. TCP connections, which are trying to cooperate in order to relief network congestion obtain a low goodput and even a worse efficiency than XTP.

It must be observed that if traffic control mechanisms at the ATM layer are adopted, this biased behaviour should be easier to control; but, if the transport layer traffic control mechanisms come into play, TCP obtains worse performance due to its embedded window control mechanism. This very simple example, however, shows that some very unpleasant phenomena could arise if different protocols are mixed together without proper control.

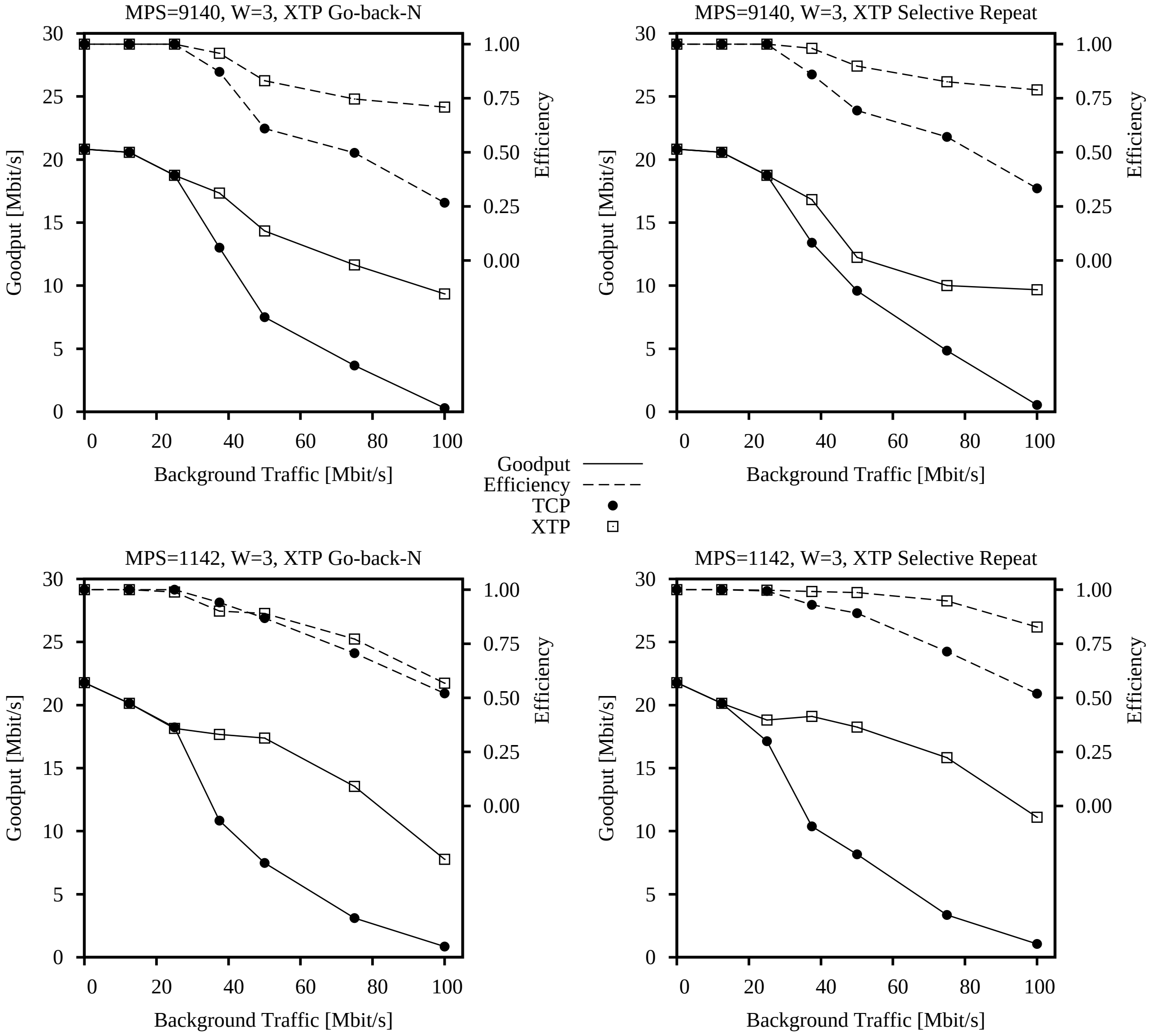

Figure 8 Aggregate average goodput and efficiency for 3 XTP and 3 TCP connections sharing the bottleneck link with Poisson background traffic.

5 CONCLUSIONS

The performance of two transport protocols, TCP and XTP, when used to support file 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.

Several numerical results were derived 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.

Both protocols tend to have unfair behaviours when several sources with different RTT share network resources. Shaping at the ATM layer reduces this phenomenon, but it is rather ineffective in the other considered scenarios.

Finally, users running TCP in a UBR-like context similar to the one presented in this paper suffer lower performance when compared with users relying on XTP. The bias against TCP users is quite clearly visible in a UBR context, but the picture could be significantly modified considering more elaborate (ABR-like) traffic control techniques.

REFERENCES

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

M. Ajmone Marsan, A. Bianco, R. Lo Cigno, M. Munafò, "Some Simulation Results about Shaped TCP Connections in ATM Networks", in: D.Kouvatsos (editor), *Performance Modelling and Evaluation of ATM Networks – Vol.2*, Chapman and Hall, London, 1996

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

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

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

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

T. Lakshman, A. Neidhart, T. Ott, *The Drop from Front Strategy in TCP and in TCP over ATM*, IEEE INFOCOM'96, San Francisco, CA, March 1996

H. Li, K. Siu, H. Tzeng, *A Simulation Study of TCP Performance in ATM Networks with ABR and UBR Services*, IEEE INFOCOM'96, San Francisco, CA, March 1996

R. Lo Cigno, M. Munafò, *CLASS User Manual*, (1996) available via WWW through `http://hp0tlc.polito.it/class.html`

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

W. Stevens, *TCP/IP Illustrated, Vol. 1,* Addison-Wesley, 1994

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

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