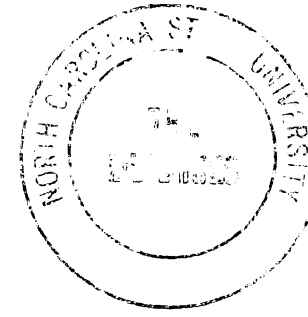
A Performance Study of the XTP Error Control

Arne A. Nilsson

Meejeong Lee



Center for Communications and Signal Processing
Department of Electrical and Computer Engineering
North Carolina State University

TR-93/6
March 1993

A Performance Study of the XTP Error Control

Arne A. Nilsson and Meejeong Lee

Center for Communications and Signal Processing
Department of Electrical and Computer Engineerin
North Carolina State University
Raleigh, North Carolina 27695-7914

Abstract

The performance of different implementation choices of the XTP Error Control in the presence of transmission errors and finite buffer overflows is studied by means of simulation. The performance measure investigated is the end-to-end delay achieved by different implementations in transferring variable length messages, arriving according to a Bernoulli process, between two stations over a network. The results are compared to those obtained for an Interrupted Bernoulli arrivals. The performance of different implementations under varying window sizes is also studied to investigate how the flow and the error control strategies impact each other in XTP.

1. INTRODUCTION

The Xpress Transfer Protocol (XTP) [1] is a high performance protocol motivated by the needs of contemporary and future distributed, real-time, transactional, and multi-media systems. One of the XTP features provided to meet the requirements of those systems is the error control mechanism.

Error control in XTP incorporates positive and negative acknowledgement to effect retransmission of missing or damaged data packets. Retransmission may be either go-back-N (GBN) or selective repeat (SR). The conservative error control algorithm reports an error only in response to the transmitter's request. The error control algorithm, while basically conservative, can also be aggressive: a quick-acting error notification is provided.

Since the XTP design allows various kinds of XTP implementations to interoperate, we have several different implementation choices both at the sender and the receiver. Moreover, since the sender has no knowledge about the strategy applied by the receiver and vice versa, all possible combinations of the sender and receiver implementations can occur in a single connection. It is important to study their performance and to understand which implementations are more robust, i.e., less dependent on the implementation chosen by the partner station. A similar study for ISO Transport Protocol was done by Meister [2].

The goal of this paper is to investigate the performance of the different implementation choices for the retransmissions of corrupted or lost data packets. Six different implementation combinations are studied by means of a simulation. The performance measure investigated is the end-to-end delay required to deliver variable length messages between two stations over a network.

Main features of XTP error control strategies are given in Section 2. The simulation and traffic models used for examination of those strategies are described in detail in Section 3. Six implementation combinations of the XTP protocol are analyzed in Section 4 with respect to their performance under two different arrival process assumptions: Bernoulli process, and Interrupted Bernoulli Process (IBP). The results of the experiment also shows that varying window size has different impact on the performance of different implementation choices. Finally Section 5 contains a summary and conclusions.

2. XTP ERROR CONTROL

In XTP, control (CNTL) packets exchange acknowledgement and state information between end systems. The sender of a data stream uses the SREQ and DREQ bits in a outgoing DATA packet or in a CNTL packet to affect the acknowledgement strategy of the receiver. Whenever the receiver sees DREQ, it transfers all the data packets which arrived in the input queue before that DREQ and then generates a CNTL packet. Whenever the receiver sees SREQ, it responds immediately with a CNTL packet containing its current state information.

If the sender does not receive a CNTL packet until a certain amount of time (WTIMER value) passes after it sends out SREQ, "synchronizing handshake" procedure starts. The sender sends out CNTL packet with SREQ bit set until a synchronization handshake completes. The term "synchronization handshake" refers to an exchange of control packets between sender and receiver after which

the sender is certain of the receiver's state. Synchronization handshake can happen only after the WTIMER expiration.

The XTP protocol also uses window flow control. A sender is allowed to transmit sequence numbers up to the value of ALLOC -1. The receiver is responsible for adjusting the sender's window by increasing the ALLOC value appropriately. Other state variables for flow and error control that we need to specify are the following: on the receiving side, HSEQ is one greater than the highest sequenced byte ever received; RSEQ is one greater than the highest monotonic sequenced byte ever received; and DSEQ is one greater than the highest sequenced byte delivered to the application. If there are no errors or dropped packets, then HSEQ and RSEQ will have the same value. Otherwise there will be "gaps" in the sequence space at the receiver. If the receiver exercises SR error control, it stores the out-of-sequence packets and maintains extra variables indicating the spans of sequentially received bytes. Figure 1 shows these state variables.

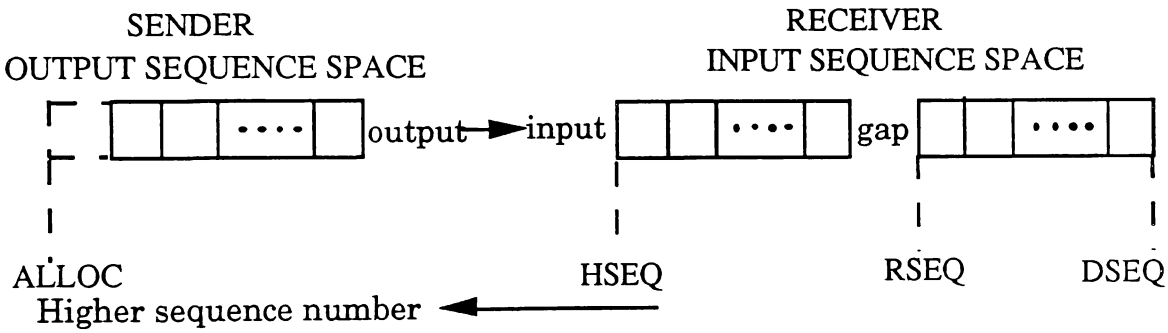


Figure 1: XTP flow control variables

In contrast to the ISO Transport Protocol Class 4, XTP error control is based on the exchange of information in CNTL packets regarding lost or damaged data. Thus outstanding packets do not require retransmission timers as in ISO TP4. CNTL packets contain the information necessary for both flow and error control: ALLOC, DSEQ, RSEQ, and the list of pairs of numerals defining spans of sequentially received bytes.

The error detection procedure is split between the sender and the receiver. A receiver detects missing packets by checking its incoming packet stream for sequence number gaps, and acknowledges the packets received in sequence or requests retransmissions by reporting its input queue status in CNTL packets. The sender is responsible for setting SREQ/DREQ to invoke the CNTL packet from the receiver and interpreting the CNTL packets.

The Protocol Definition permits various retransmission strategies of which the basic ones are presented in Table 1.

Table 1
Action of the sender upon receiving the CNTL indicating the gaps at the receiver input queue.

1:	Only retransmit after a synchronizing handshake:
	1.1: The sender retransmits only the gaps indicated by the CNTL packet (SR)
	1.2: The sender retransmits all the DATA packets containing byte sequence greater than or equal to received RSEQ (GBN)
2:	Initiate retransmission after receiving certain CNTL packets:
	2.1: SR
	2.2: GBN

There are additional variants of the sender strategies which can possibly affect the performance. Among those are the varying DREQ/SREQ setting policies to affect the receiver's acknowledgement. Only the case of setting SREQ in the outgoing DATA packet whenever the packet contains the byte with sequence number equal to ALLOC - 1 is considered in the following. Setting SREQ in every outgoing CNTL packet is required by the protocol definition. Case 1 is the most conservative form of error correction. For our simulation study, we are interested in more aggressive form of error control allowed by the second case.

When an XTP DATA packet arrives at the receiver side, it may either be the DATA packet expected next, i.e., it is in-sequence, or it carries a sequence number different from the number anticipated next, i.e., it is out-of-sequence. If it is an out-of-sequence packet and the receiver stores the out-of-sequence packets (SR error control), the receiver may apply different strategies: transmitting CNTL for the first gap in its current input queue, or taking no action. In a GBN receiver, whenever HSEQ differs from RSEQ, CNTL packet should be transmitted to the sender to request the retransmission immediately. These strategies are summarized in Table 2.

Table 2
Action of the receiver upon receipt of an out-of-sequence packet

1:	The receiver stores the out of sequence packet, and takes no further action.
2:	The receiver stores the out of sequence packets. If the number of gaps is one, send out CNTL to inform the sender of the gap.
3:	The receiver discard the out of sequence packets, and send out CNTL to request retransmission starting from RSEQ.

This leaves us with two different implementation choices at the sender sides and three different implementation choices at the receiver sides. Since the CNTL packet design contains all the state information for both SR and GBN error control and retransmissions, and the sender and the receiver have no knowledge about the strategy applied by their peer, all possible combinations of the sender and receiver implementations are completely legitimate and possible to occur in a single connection. Table 3 shows the six different combinations considered in our study.

Table 3
Protocol versions studied

1:	Sender practices Go-Back-N (GNB) retransmission.
1.1:	Receiver discards the out-of-sequence packets, and sends out retransmission request whenever it finds a gap.
1.2:	Receiver stores the out-of-sequence packets, and sends out retransmission request only when it is asked by the sender.
1.3:	Receiver stores the out-of-sequence packets, and sends out retransmission request for the first gap of the current input queue.
2:	Sender practices Selective Repeat (SR) retransmission.
2.1:	Receiver discards the out-of-sequence packets, and sends out retransmission request whenever it finds a gap.
2.2:	Receiver stores the out-of-sequence packets, and sends out retransmission request only when it is asked by the sender.
2.3:	Receiver stores the out-of-sequence packets, and sends out retransmission request for the first gap of the current input queue.

Obviously, there are other options for error control strategies to serve various application requirements. Such strategies are beyond the scope of this paper.

Subsequently, the performance resulting from the six possible combinations of sender and receiver protocol implementations is investigated. The simulation model used for examination of these strategies is described in the next section.

3. MODELS, PARAMETERS AND ASSUMPTIONS

3.1 Simulation model

Figure 2 shows the simulation model employed to compare the performance of the different implementation choices of the XTP error control. Our model encompasses two stations interconnected through a network. There are two modules explicitly modeled for our station model: the XTP layer and the link transport protocol layer. These two modules plus the network connecting two stations are embodied in a network of queues as displayed in Figure 2.

Both the XTP module and the link transport module have their own servers for their layer appropriate processing service, instead of sharing one processor together. This is realistic in the XTP environment since XTP is planned to be implemented with a very large scale integration (VLSI) chip set dedicated to protocol processing.

Operation of the simulation model is as follows. Since the time for the connection setup was not modeled, it starts with the first message arrival from the application. The variable length message arrived at the sender's XTP module (XTP1) is segmented into a fixed length XTP DATA packet. The DATA packet is then transferred to sender's link transport module (LINK1) and processed there to be transmitted to the receiver's link transport module (LINK2) via the network. While the DATA packet is transmitted though the network, transmission bit error can affect it. When the packet arrives at LINK2 module, it is processed to be

transferred to the receiver's XTP module (XTP2). If the packet was found to be corrupted due to a transmission bit error, LINK2 module discards it, instead of transfering it to XTP2 module. Each arriving DATA packet is processed and reassembled into the original message at XTP2 module before it is delivered to the application. The XTP2 module generates a CNTL packet as required by the protocol. This CNTL packet is then received and processed by the XTP1 module.

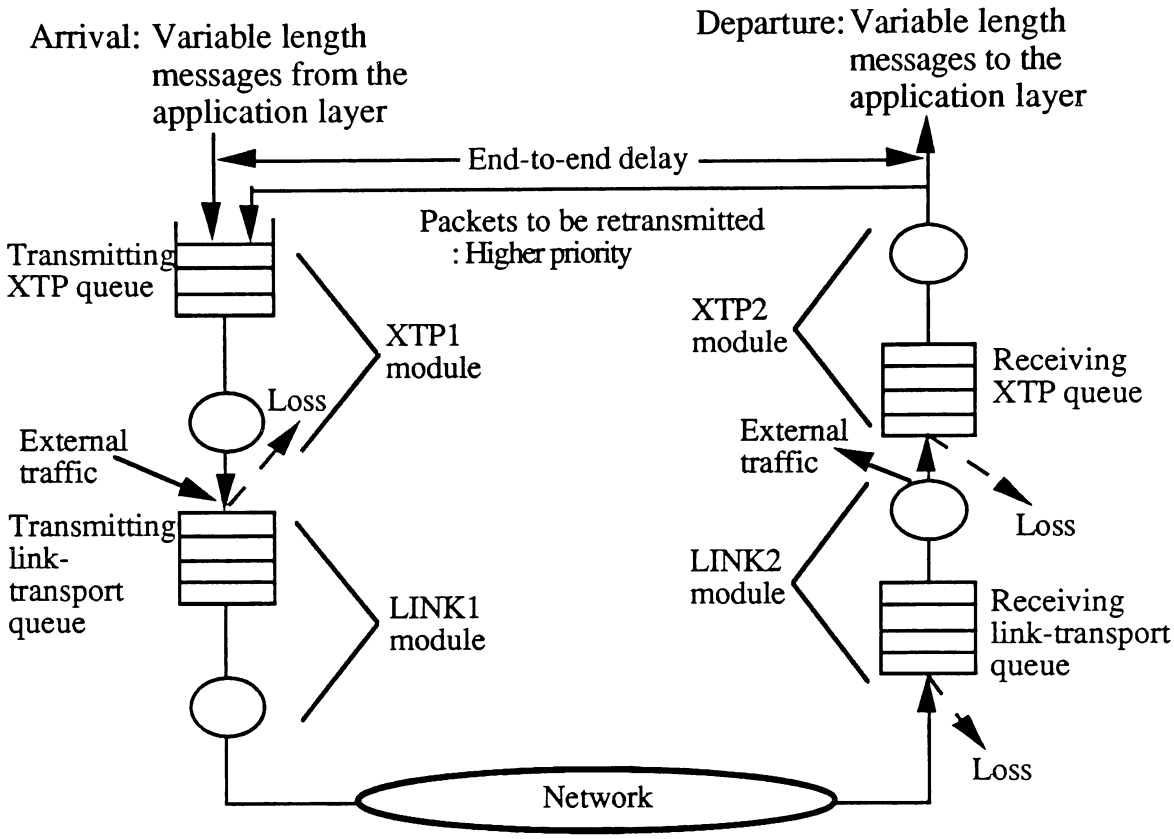


Figure 2: Simulation model

3.2 Assumptions and parameters

The application on top of XTP is assumed to generate variable length messages and not to allow the out-of-order delivery.

Except for the XTP1 queue, all the queues are assumed to have finite capacity, and a buffer size of 10 packets was assumed in our experiment. The service discipline at all queues is first come/first served, but packets to be retransmitted have priority over new packets arriving from the application in XTP1 queue.

While DATA packets can be lost or corrupted due to buffer overflow and transmission bit errors, CNTL packets are assumed not to be lost or corrupted. The amount of the processing time and the propagation delay for a CNTL packet is assumed to be the same as those for a DATA packet, and the transmission time for a CNTL packet is assumed to be negligible.

The network connecting two stations in our model is assumed to be just a simple transmission link, with 100 Mbps capacity. Thus the transmission time for a 1024-byte packet is 0.08 ms. We defined this 1024 byte packet transmission time as the slot time in the simulation experiment. The assumed propagation delay of the link is 0.5 ms (6 slots), corresponding to a length of 100 km.

A DATA packet may be corrupted during transmission by bit errors. The bit error rate considered for our study is 10^{-9}, and the corresponding packet error rate for a 1024-byte packet is then 8×10^{-6}.

Except for the experiment where the window size was varied, a window size of sixty packets was chosen.

For the processing requirements, we assumed 0.16 ms (2 slots) for both the XTP and link transport modules. It was further assumed that both stations are symmetric. In the simulation experiments, the implementation complexity is assumed to be about the same for all the six cases, i.e., they require the same processing times.

The traffic on the end-to-end XTP connection is called the 'user traffic'. The user traffic interferes with the traffic from the other connection(s), and we represent this traffic as the 'external traffic' in figure 2. The external traffic was removed after being served at LINK2 module.

3.3 Traffic models

We studied the performance of XTP error control under two different arrival processes, Bernoulli and IBP, to investigate how the burstiness of an arrival process can affect the performance of different implementation choices in XTP. The impact of the burstiness of an arrival process on the end-to-end performance of ATM and Broad Band networks was studied by Nilsson and Huterer [3]. The "burstiness" is quantified by the squared coefficient of variations of inter-arrival times (C^2) which is equal to the variance over the square of the mean interarrival times.

The Bernoulli process is defined over a slotted (discrete) time axis. The number of arrivals at each discrete time slot is 1 with the probability α and 0 with the probability $1 - \alpha$, that is, the interarrivals are independent with the same geometric distribution [4].

The IBP is also defined over a discrete time axis and it comprises two states, and active state and an idle state, which alternate. The transitions between states are governed by the two state Markov chain as shown in Figure 3. Given that the process is in the active state (or the idle state) at slot i, it will remain in the same state in the next slot $i + 1$ with probability p (or q), or will change to the idle state (or the active state) with probability $1 - p$ (or $1 - q$). During the active state, arrivals occur in a Bernoulli fashion with the probability that a slot contains an arrival to be α. No arrival occurs if the process is in the idle state [5].

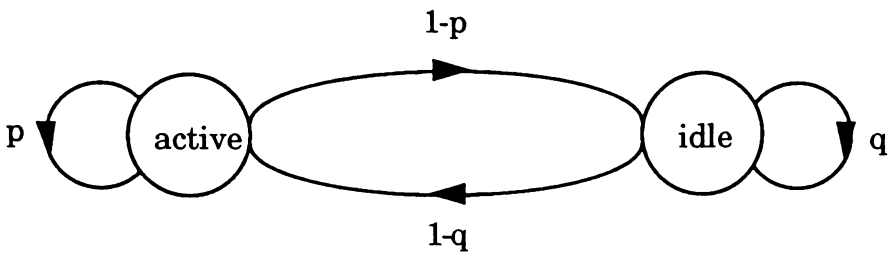


Figure 3: The Markov chain for an IBP

The size of the XTP DATA packet was set to be 1024 bytes. The length of a message in packets arriving from the XTP application is geometrically distributed with parameter 0.5. Thus the average message length is two packets.

The arrival process for the external traffic arriving at LINK1 module was assumed to be a Bernoulli process with $\alpha = 0.1$.

4. PERFORMANCE OF THE XTP ERROR CONTROL SCHEMES

This section presents the results on the performance comparison of several versions of the XTP error control. The six cases studied in detail have already been given in Table 3. Those six cases are investigated with two different arrival processes:

Bernoulli process
IBP

They are also studied under different window size assumptions to investigate how the window size affects the performance of those strategies. In [6], it is shown that the window size impacts the performance of SR and GBN error control and retransmission strategies in a different way.

In an abbreviated form, we shall call the six strategies as follows:

Scheme 1.1: GBN - discard,
Scheme 1.2: GBN - store and conservative,
Scheme 1.3: GBN - store and aggressive,
Scheme 2.1: SR - discard,
Scheme 2.2: SR - store and conservative,
Scheme 2.3: SR - store and aggressive,

where the first part describes the behavior of the sender upon CNTL receiving, and the second part describes the action of the receiver when an out-of-sequence XTP DATA packet was received.

The mean end-to-end delay for a message, given in number of slots, is the performance measure in which we are interested. The end-to-end delay for a message is the time from the arrival of a message at XTP1 queue until the departure of the message at XTP2 module.

Figure 4 displays the end-to-end delay of a message as the function of user throughput for the six different cases with Bernoulli arrivals and window size of 60 packets. Since the transmission bit error rate assumed is very low, packet corruption due to transmission error is rare and most of the packet losses are caused by the buffer overflow. When the throughput is very low, buffer overflow is rather unlikely, and thus the number of retransmissions required is small. Owing to the above fact and our assumption of equal processing times for all the six implementations, the end-to-end delay is almost the same when the throughput is very low.

The performance difference between the different schemes increases as the throughput gets larger. For increasing throughput scheme 2.3, "SR - store and aggressive" leads to the smallest delay, and schemes 1.1 and 2.1, "GBN - discard" and "SR - discard" to the largest delay under the given assumptions.

Note that when the receiver discards the out-of-sequence packets, the sender behavior does not affect the performance. Since, when the receiver discards the out-of-sequence packets, it requests the sender to retransmit everything in the current window starting from RSEQ, that is, it forces the sender to exercise GBN retransmission. For this reason the end-to-end delay for the schemes 1.1 and 2.1 turned out to be the same.

When the sender exercises SR and the receiver stores the out-of-sequence packets, aggressive error control reduces the end-to-end-delay in our experiment. This is due to the fact that the quick acting retransmission request reduces the retransmission delay.

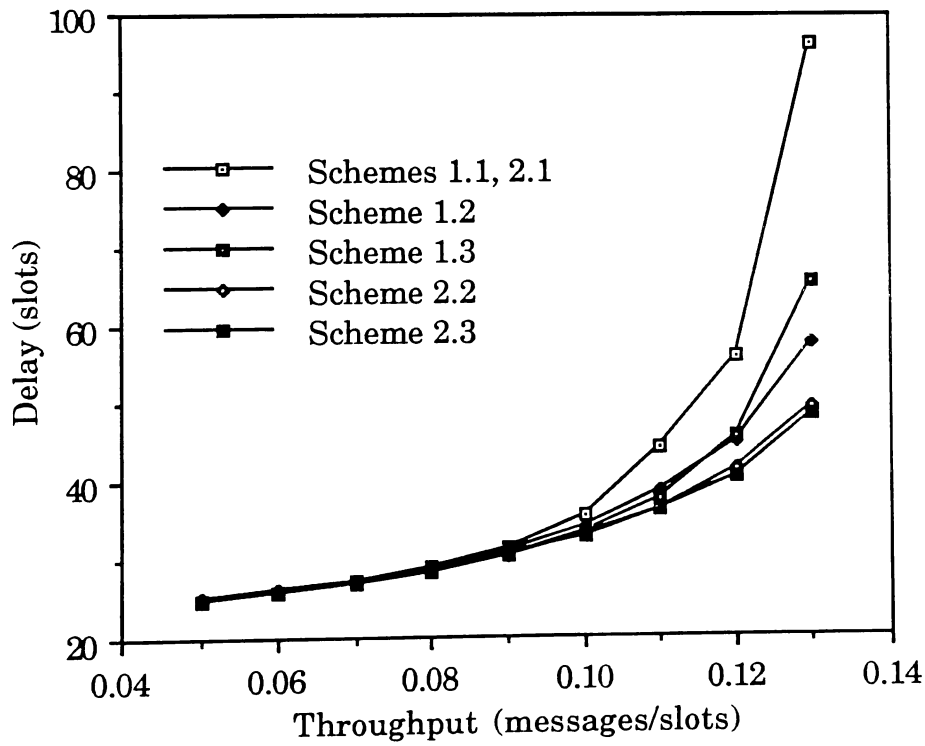


Figure 4: Mean end-to-end delay as a function of user throughput: Bernoulli arrival

On the other hand when the sender exercised GBN, aggressive error control helped decrease the end-to-end delay when the throughput was low, but worsened the performance after the amount of throughput passes a certain point as we can see from the crosspoint in Figure 4. This behavior can be explained as follows. As mentioned earlier, most of the packet losses are due to the buffer overflow under our very low transmission bit error rate assumption. Therefore if the quick acting error report from the receiver arrives too early, i.e, before the buffer overflow situation is relieved completely, it aggravates the overflow situation and causes another packet losses. This circumstances happen easier if the buffer overflow lasts longer, and in turn the buffer overflow is more likely to last longer under heavy traffic, that is, when the throughput is high. Since the SR sender retransmits only the out-of-sequence packets, the amount of traffic issued for a retransmission is rather small and immediate retransmission requests do not harm significantly. For GBN sender, however, too early retransmission requests can issue a large amount of traffic at once and degrade the performance.

Since the "GBN" senders retransmit more packets than the "SR" when the receiver stores the out-of-sequence packets, the schemes 2.2 and 2.3 show the better performance than the schemes 1.2 and 1.3 respectively.

Actually, the same number of packets is retransmitted in schemes 1.1 and 2.1 where the receiver discards out of sequence packets, and in scheme 1.3 where the receiver stores the out of sequence packets. But schemes 1.1 and 2.1 perform remarkably worse than scheme 1.3. This can be explained as follows. In scheme 1.3 the out-of-sequence packets which had already been received correctly and stored are also retransmitted by the sender. However, in scheme 1.3, the XTP module at the receiver side only has to wait for the arrival of the lost packets, and

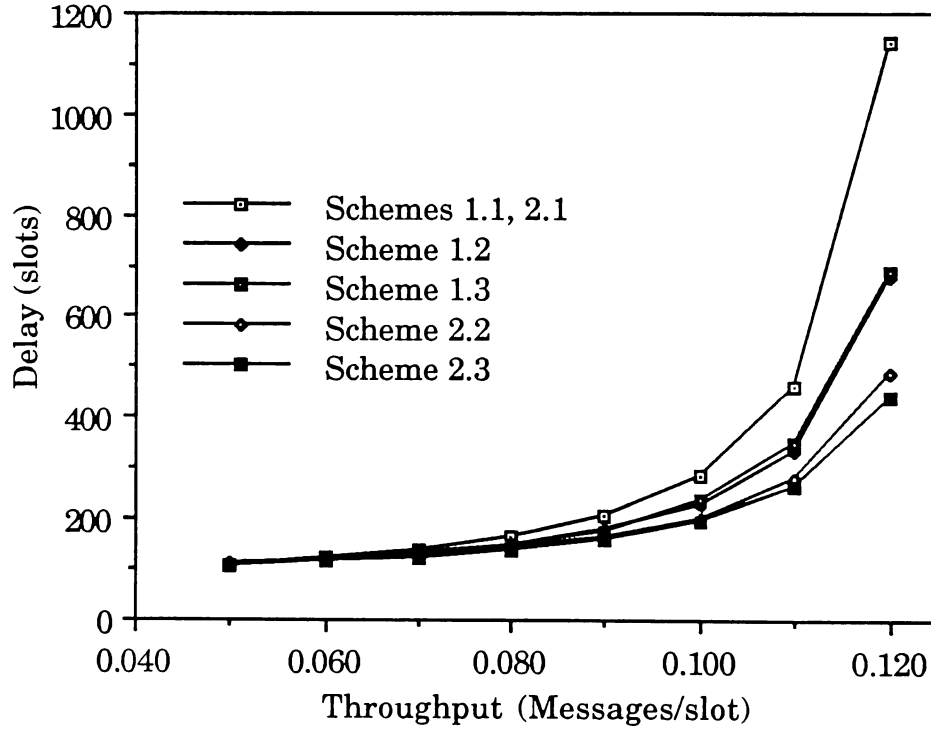


Figure 5: Mean end-to-end delay as a function of user throughput: IBP arrival

can start reassembling new messages with all the packets waiting in the queue as soon as it receives those lost packets. On the other hand, in Schemes 1.1 and 2.1, the XTP module at the receiver has to wait for all the discarded out-of-sequence packets.

The result of the same experiment for an IBP arrival is given in Figure 5. The burstiness of the IBP (C^2) was set to 10 in our experiment. The graphs for the IBP arrival show very similar characteristics to those for the Bernoulli arrival, and they can be explained in the same way.

Figure 6 compares the end-to-end delay for the Bernoulli and the IBP arrivals. The end-to-end delay for the IBP arrival is larger than that for the Bernoulli arrival in the order of tens at high throughput. The difference is bigger in the region of high utilizations, and the performance of scheme 1.1(2.1) degrades most significantly for the IBP arrival among the six schemes.

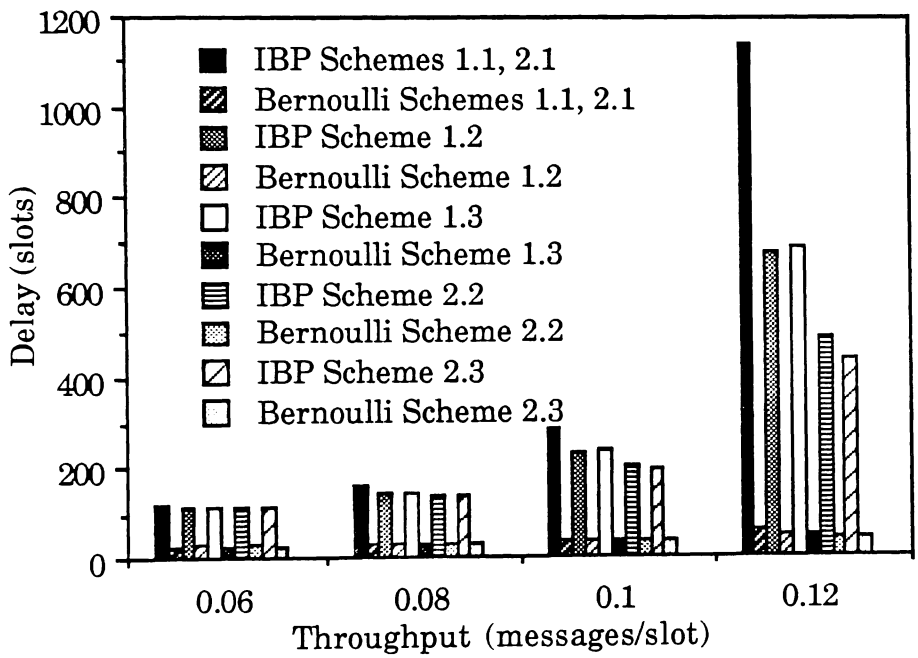


Figure 6: Comparison of the mean end-to-end delay: Bernoulli and IBP arrivals

Different schemes have different optimum window sizes. Table 4 shows the optimal window size for our six schemes when the throughput is set to 0.12 messages/slot. 10 different window sizes (from 30 to 120 packets) were tried in our simulation, and the figures shown in the table is the window size that gave the smallest end-to-end delay in this experiment.

Table 4
Optimal Window Size

Scheme	Window size
Scheme 1.1, 2.1	50
Scheme 1.2	70
Scheme 1.3	60
Scheme 2.2	90
Scheme 2.3	90

Rather big window sizes are expected to be the optimums for all the schemes according to this result. This is due to the fact that the sender invokes the acknowledgement once per window by setting the SREQ bit in the last DATA packet of a window. The sender has to stop and wait for the ALLOC value to increase at the end of every window, and the percentage of this overhead waiting time gets larger as the window size gets smaller.

On the other hand, there are several factors that make larger window size unprofitable. First thing is the increased retransmission delay: since the sender asks the receiver status once, at the end of a window, retransmission delay increases as the window size increases for the conservative receiver. Since, when there are lost or corrupted packets in the middle of a window, the subsequent packets arriving out-of-sequence at the receiver side should wait in the receiving XTP queue until the missing ones arrive, retransmission time impacts the end-to-end delay consequently.

Secondly, for the GBN sender, the average number of packets to be retransmitted at once increases as the window size increases. Since a sudden traffic increase may cause or lengthen the buffer overflow situation, it worsens the performance.

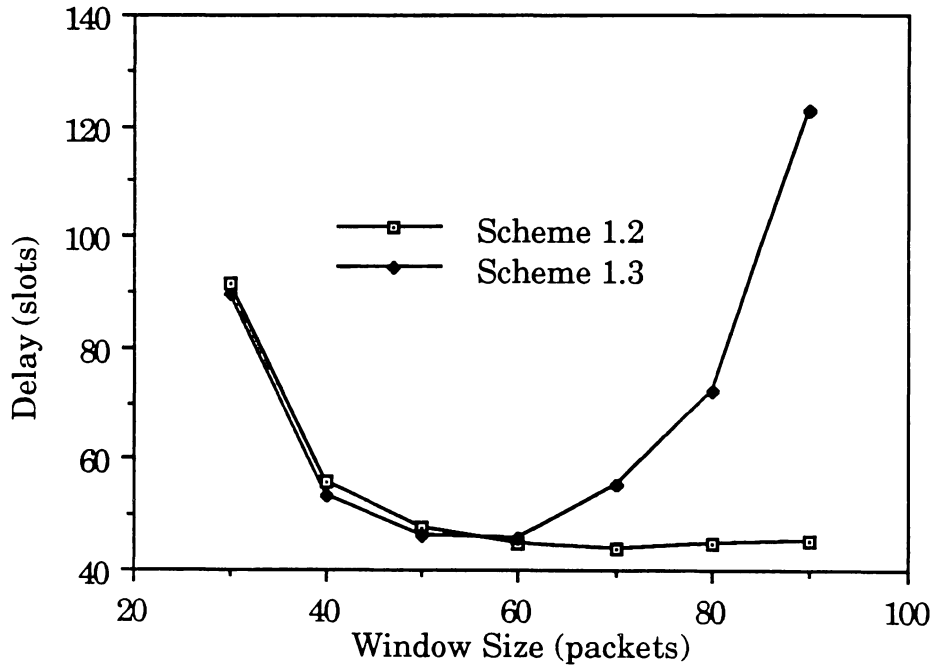


Figure 7: Mean end-to-end delay as a function of window size, schemes 2.1 and 1.3

Finally the aggressiveness of the receiver can degrade the performance as the window size gets larger. Figures 7 and 8 show this impact: the schemes 1.3 and 2.3 outperform the schemes 1.2 and 2.2 respectively for up to a certain window size, but the opposite becomes true for the window sizes larger than that. Since the receiver always set ALLOC value in the outgoing CNTL packet to be equal to DSEQ + constant (window size), the flow control is affected by the error control strategy very closely. Thus the too early error report explained above can

lengthen or aggrevagate the buffer overflow situation by sliding the window at the sender as well as by incurring traffic for the retransmission at an inappropriate time. The performance degradation for large window size due to the aggressiveness of the receiver is worse when the sender is GBN as we can see by comparing Figures 7 and 8.

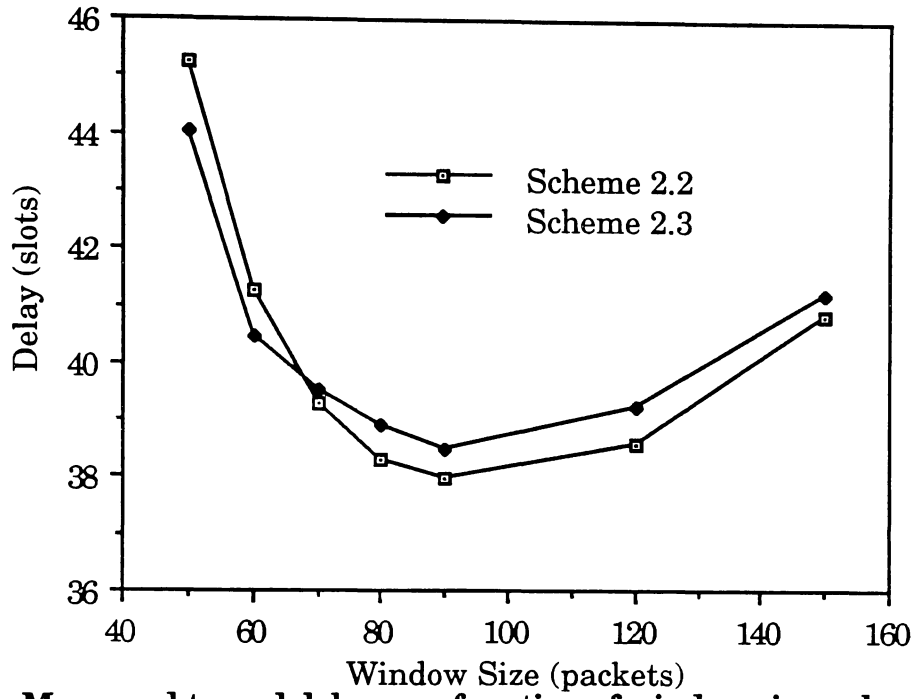


Figure 8: Mean end-to-end delay as a function of window size, schemes 2.2 and 2.3

Note that the results given in this section are drawn under certain limiting assumptions. We may have different results with different setting of the window size, and with different throughput assumption. There are also other aspects of which we are aware that are of importance in evaluating the performance of SR and GBN error control strategies. These aspects include processing overhead, intermediate buffer requirements, quality of the service provided by he lower layers, etc. [6]. Those factors are to be considered in our future studies.

5. SUMMARY AND CONCLUSION

The XTP permits several possibilities in the implementation of error control mechanism at both ends of a connection. Since the sender has no knowledge of the implementation chosen by the receiver, and vice versa, all possible combinations of sender and receiver strategies can occur in practice, and it is important to understand the behavior of the various implementation choices.

An application generating variable length messages and requiring orderly delivery was chosen and modeled on top of XTP. The network connection was a simple link, and the DATA packets were assumed such that they could be lost or corrupted. A simulation model was implemented including all the important functions of XTP error control.

XTP employing the following combinations of sender and receiver strategy were compared:

i) the sender behaves as SR/GBN;
ii) the receiver stores/discards out-of-sequence DATA packets, and aggressively/conservatively reports errors.

The results show that, at the sender side, "SR" yields higher performance than "GBN" except for the case when the receiver discards the out-of-sequence packets, the sender behavior does not matter in that case. At the receiver side, "store" results in smaller end-to-end delay than "discard", and the aggressiveness is helpful only when the system load and the window size are within a certain limit.

The performance of the XTP error control under two different arrival processes, Bernoulli process and IBP, was studied to investigate the impact of the burstiness of an arrival process. The end-to-end delay increases substantially as the burstiness increases. The performance degrades most significantly for an increase in burstiness when the receiver "discards" the out-of-sequence packets.

Different implementation takes different optimum window sizes, and the optimum window sizes are rather big. "SR - discard" and "GBN - discard" pairs take the smallest optimum window size. The aggressive receiver may perform poorly compared to the conservative receiver when the window size is big.

6. REFERENCES

1. XTP Protocol Definition Revision 3.6, PEI, Santa Barbara, CA, 1992.
2. Bernd W. Meister, A Performance Study of the ISO Transport Protocol, IEEE Trans. Comput., vol. 40, no 3, pp. 253-262, 1991.
3. A. Nilsson and M. Huterer, End-to-End Performance of ATM and Private Broadband Networks, in Proc. The Twenty-Fourth Southeastern Symposium on System Theory, Greensboro, NC, 1992.
4. J.-R. Louvion, P. Boyer, and A. Gravey, A Discrete-Time Single Server Queue with Bernoulli Arrivals and Constant Service Time, in Proc. 12th International Teletraffic Congress, Torino, 1988.
5. A. Nilsson, F. Lai and H. Perros, An approximate analysis of a bufferless N x N synchronous clos ATM switch, in Proc. Canadian Conference on Electrical and Computer Engineering, Ottawa, Ontario, Canada, pp. 39.1.1-39.1.4, 1990.
6. W. Stallings, Data and Computer Communications, Macmillan Publishing Company, Macmillan, Inc., N.J., pp. 145-147, pp 480-503, 1991.