Comparison of TP4, TCP and XTP - Part 2: Data Transfer Mechanisms (*)

Yves Baguette(**), André Danthine
Université de Liège, Service de Systèmes et Automatique,
Institut d'Electricité Montefiore B28, B-4000 Liège (Sart Tilman), Belgium

Abstract. This paper is the second part of a comparative study of the transport mechanisms of three protocols: ISO TP4, XTP and, to a lesser extent, DARPA TCP. Whereas the first part is entirely dedicated to the connection management mechanisms, the mechanisms compared in this second part are those used during the phase of data transmission over a connection. The comparison aims at highlighting the limitations and drawbacks of the data transfer mechanisms of the existing TP4 and TCP protocols as regards their applicability in HSLANs environments characterized by a high bandwidth and a low error rate. It also aims at clearly identifying the area where the emerging XTP contributes to the improvement of the past situation. Like its companion paper, this paper takes account of the last Revision 3.6 of XTP and of the upgraded version of the ISO Transport Protocol which has been circulated for ballot as Draft International Standard (DIS) 8073.

Keywords. Transport Protocols, High Performance, High-Speed LANs, TP4, TCP, XTP, OSI 95, CIO.

1 . Introduction

The subject of this paper and of the companion paper [1] is a detailed comparison of the transport mechanisms of the ISO Transport Protocol class 4 (TP4) [2], the Xpress Transfer Protocol (XTP) [3] and, to a lesser extend, the DARPA Transmission Control Protocol (TCP) [4]. The comparative analysis of these three protocols will focus on their mechanisms exclusively, and especially from a qualitative rather than quantitative point of view, so that all the aspects regarding the syntax or the implementation will simply be left apart.

The mechanisms of TP4 will be examined in detail to highlight all their limitations and drawbacks with respect to their applicability in the high-bandwidth low-error-rate area. An in-depth analysis of the transport mechanisms of XTP will be carried out in parallel in order to clearly identify how it contributes to the improvement of the past situation. In other words, every major transport mechanism of XTP, in addition to being described, will be compared with the equivalent one of TP4 (if it exists!). It means that the mechanisms of TP4 will be utilized as points of reference for the study of the transport mechanisms of XTP. Whenever it turns out that they bring further information into the discussion, the mechanisms supplied in TCP will also be described and compared with those of TP4 and/or XTP.

This paper is entirely dedicated to the data transfer mechanisms whereas its companion paper [1] is dedicated to the connection set-up and release mechanisms.

Like in [1], our work in this paper will be based on the very last Revision 3.6 [3], issued on January 1992.

Moreover, our comparative study will take account of the upgraded version of the ISO Transport Protocol which has been circulated for ballot as DIS 8073 [5].

Section 2 of this paper reminds us of the basis for the comparative study. The data transfer mechanisms which are to be examined in this paper are then classified in section 3. Each of the sections 4 to 8 tackles a category of mechanisms resulting from this classification. Conclusions in section 9 summarize the main lessons that can be drawn from the previous sections.

2 . Basis for the comparative study

The basis for the comparative study of the data transfer mechanisms is exactly the same one as that adopted in section 2 of the companion paper [1] to compare the connection set-up and release mechanisms.

(*) This paper results from work that was carried out within the framework of the ESPRIT II Project "OSI 95" (Project n°5341) and more recently of the RACE Project "CIO" (Project n°2060).
(**) Research assistant of the Belgian National Fund for Scientific Research (F.N.R.S.).

2.1. Initial assumptions regarding the comparison

We also make the same assumptions as in [1] as regards the context of the comparison:

- We will assume first an isolated LAN or a set of LANs interconnected at the MAC level by bridges. Such an initial assumption allows us to pay no attention to the routing part of XTP for the moment.
- The protocols TP4 and XTP will be assumed to have access to the LLC Type 1 Service [6] through the non-OSI SubNetwork Access Protocol (SNAP) for XTP and through the Inactive Network Layer Protocol Subset of the IS0 ConnectionLess Network Protocol (CLNP) [7] plus the SubNetwork Dependent Convergence Functions (SNDCFs) [8] for TP4.

2.2. Conventions

Again, we use the following conventions that have already been used in the companion paper [1].

When no confusion is possible, the generic term ‘transport connection’ will be used to designate any of the following specific connections: an OSI Transport Connection (TC), an XTP connection or a TCP connection. In the same manner, the generic term ‘transport protocol entity’ will be used to designate any of the following specific protocol entities: an OSI transport entity, an XTP entity or a TCP entity.

Additionally, the term ‘TPDU’ is an OSI naming which must be used imperatively with the standard ISO Transport Protocol. Throughout this paper, it will also be used with TCP. The designers of XTP preferred the term ‘packet’ to designate a PDU that is exchanged between peer XTP entities. We will use ‘TPDU’ as a generic term but, in the next sections, the specific term ‘XTP packet’ will replace it whenever XTP is discussed.

3. Classification of the main transport mechanisms

All three TP4, XTP and TCP are connection-oriented protocols. The main transport mechanisms that are used in LANs environments during the phase of data transfer over a connection may be classified into a few categories.

3.1. Data units mapping mechanisms

Peer transport protocol entities communicate by exchanging TPDUs. These TPDUs must be mapped into Network Service Data Units (NSDUs) on the sending transport protocol entity's side, and from NSDUs on the receiving transport protocol entity's side. Different mapping mechanisms between the TPDUs and the NSDUs at the network service interface are possible.

The TPDUs that carry user data must be mapped from Transport Service Data Units (TSDUs) on the sending transport protocol entity's side, and into TSDUs on the receiving transport protocol entity's side. Different mapping mechanisms between the TPDUs and the TSDUs at the transport service interface are possible too.

3.2. Error control mechanisms

Considering that a connection-oriented transport protocol is meant to ensure reliable data transmissions with an end-to-end control, the transport protocol entities must implement error control mechanisms to detect and recover from non-signalled errors which can occur during the TPDUs exchanges.

3.3. End-to-end flow control mechanism

An end-to-end flow control mechanism at the transport level is a mechanism that allows a receiving transport protocol entity to control the flow of arriving data on a per-transport-connection basis by acting on the remote sending transport protocol entities.

3.4. Congestion control mechanism

If two LANs are interconnected by a router and if this router is the sole joining element between the two LANs, the network packets which convey the TPDUs relating to the multiple transport connections established between end systems connected to the first LAN and end systems connected to the second LAN will all have to travel through the router. It is therefore possible to have a bottleneck in the router while each receiving transport protocol entity is able to follow the transmission rate imposed by the peer sending transport protocol entity.

Unlike the end-to-end flow control in the transport layer which is a function to prevent the arrival of data that have no place to go in the receiving transport protocol entity, the congestion control is a function to fight against the congestions inside the network service. Let us stress that the congestion control function, which concerns both network and transport layers, is provided in either of these two layers or is not provided at all, depending upon the family of protocols that is used.

3.5. Multicast mechanism

In LANs, broadcast and multicast facilities are offered by the various connectionless MAC services and by the unacknowledged connectionless LLC Type 1 Service. Such facilities are also being developed at the internet level [9]. That suggests and can only encourage their extension up to the transport level.

4. Data units mapping mechanisms

As stated in section 2, the transfer protocol XTP is assumed to have access directly to the LLC Type 1 Service through SNAP. The concepts of network service and more particularly of NSDU are therefore meaningless within the framework we have adopted for XTP. The underlying SDUs into/from which the XTP packets (i.e. the XTP-PDUs) have to be mapped are directly LSDUs.

With regard to TP4, the concept of NSDU is always meaningful since conceptually TP4 is always assumed to rely on the standard ISO ConnectionLess Network Service (CLNS) [10] within the OSI framework we have chosen for the LANs environments.

In this section, we investigate the mapping mechanisms between the TPDUs and the TSDUs on the one hand and between the TPDUs and the underlying SDUs, i.e. the NSDUs for TP4 and directly the LSDUs for XTP, on the other hand.

The normal mapping procedure between SDUs and PDUs, as defined in the OSI Reference Model (RM) of ISO, is one-to-one. Three special mapping procedures are also defined in the OSI RM, that will be analyzed hereafter: the segmentation/reassembling, blocking/deblocking and concatenation/separation procedures.

4.1. Segmentation/reassembling mechanism - Message delimitation

In LANs environments, there is usually a maximum size for the LSDUs. On the contrary, there is generally no upper bound on the maximum size of the TSDUs.

Thus, in LANs environments, either or both of the transport protocol and the internet protocol must have a mechanism to avoid submitting LSDUs whose size is larger than the maximum size authorized by the LLC service provider.

TP4 implements a segmentation/reassembling mechanism. This mechanism is mandatory because there is no limit on the maximum size of the TSDUs and because a fragmentation/reassembling mechanism is not systematically present in the internet sublayer. In fact, the two proper subsets of CLNP cannot fragment/reassemble NSDUs.

The situation is a little more complex with XTP. No segmentation/reassembling mechanism is defined explicitly for the transport part of XTP even though this transport part of XTP has a mechanism to preserve the boundaries of a user message.

4.1.1. Segmentation/reassembling mechanism in TP4

In the specification of the standard ISO connection-mode Transport Service (TS), there is no limit on the maximum size of the normal TSDUs. On the contrary, the maximum size of the TPDUs relating to a given TC is negotiated during the TC set-up phase. Even if the size of the TPDUs is not limited a priori by the Transport Protocol, it can be limited because the network service provider imposes a maximum size for the NSDUs.

So, whenever a TSDU is too large to fit into a single DaTa TPDU (DT TPDU), it has to be segmented into smaller data units.

In the transport header of the last DT TPDU of a sequence DT TPDUs stemming from a same TSDU, the EOT (End of Transmission) bit is set to mark the end of the TSDU. The receiving transport entity can thus reassemble the TSDU before delivering it to the receiving session entity.

4.1.2. Segmentation/reassembling mechanism in TCP

A TCP entity may (but this is optional!) ask a peer TCP entity to send only TPDUs which have a size less than or equal to a fixed size. Otherwise, the default maximum size for the TPDUs relating to a given TCP connection is 64 Kbytes. Segmentation in a TCP entity occurs only for user messages that are too large to fit into a single TPDU of maximum size.

The fragmentation required to fit user data in network packets which have a size less than or equal to the maximum size authorized for the packets is performed by the underlying IP [11].

4.1.3. Segmentation/reassembling mechanism in XTP

There exists no actual service definition for XTP today. Consequently, the data units that have to be submitted to the XTP service provider have not been well defined up to now. Nonetheless, according to the current specification of XTP, XTP can deal with user messages which are bounded blocks of user data bytes and which could perhaps be seen as TSDUs.

User data, i.e. data submitted by a user of the XTP service to be transmitted to a peer user, are sent in data subsegments. Only the FIRST packet and then the DATA packets may have a data subsegment in their information segment (i.e. in their middle segment between the XTP header and the XTP trailer). The mapping procedure between the user messages and the data subsegments of the FIRST packet and then of the DATA packets has not been clearly specified yet owing to the absence of a clear service definition. It seems however that a submitted user message could either be placed by the sending XTP entity into the data subsegment of a single packet or be split up into the data subsegments of several consecutive packets, which corresponds to a segmentation procedure.

Anyway, XTP provides a mechanism to safeguard the boundaries of a user message. The EOM (End of Message) flag, which may be set in the XTP header of any packet, signifies an end-of-message mark in a stream. This flag is similar to the EOT bit of TP4. Thanks to the EOM flag, a user message sent in several consecutive packets can be reassembled at the receiving endpoint.

Up to the Revision 3.5 [12], XTP also provided for the fragmentation of one XTP packet (i.e. one XTP-PDU) into several XTP fragments (i.e. several smaller XTP-PDUs) in a way which did not imply any new calculation of an integrity checksum over the data of the original XTP packet. This fragmentation mechanism of XTP was similar to the fragmentation/reassembling mechanisms of the classical internet protocols (e.g. CLNP and IP) in the routers. The fragmentation mechanism of XTP has been abandoned in the Revision 3.6 to simplify the design.

4.2. Blocking/deblocking and concatenation/separation mechanisms

Both blocking/deblocking and concatenation/separation mechanisms are used to save bandwidth.

Indeed, by blocking several (N)-SDUs into one (N)-PDU, a (N)-entity reduces the number of encapsulations required in the (N)-layer. Moreover, when blocking/deblocking is performed by the (N)-entities, the number of (N)-PDUs to process and thereby the number of (N-1)-SDUs to submit to the (N-1)-service provider is decreased, but the processing time per (N)-PDU increases.

By concatenating several (N)-PDUs into one (N-1)-SDU, a (N)-entity does not decrease the number of (N)-PDUs to process but decreases the number of (N-1)-SDUs to submit to the (N-1)-service provider. However, the processing time per (N-1)-SDU increases when concatenation/separation is performed by the (N)-entities.

4.2.1. Blocking/deblocking and concatenation/separation mechanisms in TP4

TP4 has no blocking/deblocking mechanism. Thus data stemming from several consecutive TSDUs that relates to a same TC may not be gathered into a single TPDU.

A transport entity may concatenate several TPDUs into a single NSDU. These concatenated TPDUs may relate to a same or different TCs, while maintaining the order of the TPDUs for a given TC compatible with the protocol operation. The specification of TP4 imposes some conditions to have a valid set of concatenated TPDUs. A receiving transport entity must accept any valid set of concatenated TPDUs. Then, it separates the TPDUs and processes them one after the other.

4.2.2. Blocking/deblocking and concatenation/separation mechanisms in TCP

Unlike TP4, TCP has no concatenation/separation mechanism. Thus a TCP entity may not group several TPDUs in order to send them in a single IP packet.

Nonetheless, a TCP entity may decide not to send a submitted user message to the peer TCP entity immediately. In fact, a TCP entity may consider that the amount of data waiting to be transmitted on a given TCP connection is not sufficient to justify the sending of a new data TPDU. Thus the data in a user message can be possibly combined with data in subsequent messages relating to the same TCP connection. This possibility presents similarities with the OSI blocking mechanism.

4.2.3. Blocking/deblocking and concatenation/separation mechanisms in XTP

XTP has neither a concatenation/separation mechanism nor a blocking/deblocking mechanism

A mechanism for grouping several XTP packets in a larger frame called a SUPER packet was introduced in Revision 3.5 of the specification. The promoters of XTP have decided eventually to remove the SUPER packet mechanism from the specification because the processing of the SUPER packets was deemed too complex.

The blocking/deblocking and concatenation/separation mechanisms go against the evolution towards high-speed and very-high-speed networking. Indeed, they aims at optimizing the use of the available bandwidth to the detriment of the delays in the layer where they are implemented, whereas the trend in the HSLANs for example is to recognize the existence of a large bandwidth and to try to decrease the delays. This explains why most of the new protocols, and XTP in particular, do not provide these two mechanisms any more.

5. Error control mechanisms

In general, the errors which can affect the exchanges of TPDUs between peer transport protocol entities that operate above a connectionless service are the following ones: corruption of TPDUs, loss of TPDUs, duplication of TPDUs and out-of-order delivery of TPDUs.

As regards the three protocols which are studied throughout this paper, the detection and recovery of losses, duplications and out-of-sequence deliveries apply only to the DT TPDUs in TP4, the FIRST and DATA packets in XTP and the TPDUs containing data in TCP.

5.1. Damaged TPDUs detection mechanism

To detect the corruption of the content of every TPDU in TP4 and in TCP or of every packet in XTP, all three protocols use one (or two) end-to-end checksum(s). The checksum at the transport level is intended to provide an end-to-end integrity check.

The TPDUs in TP4 and in TCP or the packets in XTP are simply discarded in case of an incorrect computed checksum at the receiving protocol entity. Thus a corruption of content is recovered as a loss. This is quite acceptable in the LANs environments considering their low bit error rate (typically 10^{-9} or better).

5.1.1. Damaged TPDUs detection mechanism in TP4

The use of the end-to-end transport checksum is optional in TP4. In fact, the non-use of the transport checksum on a given TC is negotiated during the TC establishment phase. The checksum is placed in the variable part of the transport header of every TPDU when used on a TC. It is calculated over the entire TPDU with the checksum field set to 0 while computing.

5.1.2. Damaged TPDUs detection mechanism in TCP

The use of the end-to-end checksum in TCP is mandatory. This checksum is placed in the TCP header of every TPDU. It is calculated over the entire TPDU, i.e. the TCP header and the user data (when the TPDU contains data), but also covers a pseudo-header which is conceptually prefixed to the TCP header. While computing the checksum, the checksum field is set to 0.

5.1.3. Damaged packets detection mechanism in XTP

XTP uses two complementary integrity checksums.

The Dcheck checksum covers the variable-length middle segment of the packet between the XTP header and the XTP trailer, i.e. the information segment or the control segment, including any leading (offset) padding bytes and any trailing (align) padding bytes. The <Dcheck> field is the sole field in the XTP trailer.

On the other hand, the HTcheck checksum is calculated over the fixed-length XTP header of the packet, excluding the <route> field, the <ttl> field and the

<HTcheck> field itself. The <HTcheck> field is indeed the last field of the XTP header in the packet.

In case of a wrong computed HTcheck checksum at the receiving XTP entity, the packet is discarded. In case of a correct computed HTcheck checksum but an incorrect computed Dcheck checksum, the receiving XTP entity can identify the remote sending XTP entity safely. A control packet (CNTL packet) could therefore be returned to this sending XTP entity immediately in order to report the error. However, as will be seen later on in this section, a receiving XTP entity normally sends a CNTL packet back only on demand from the remote sending XTP entity. So, normally, the damaged packet should simply be dropped without returning a CNTL packet immediately.

The use of the HTcheck checksum is mandatory whereas the use of the Dcheck checksum is optional. The NOCHECK flag is set in the <options> field of the XTP header of a packet to indicate that the calculation of Dcheck is disabled for this packet.

5.2. Lost TPDUs detection and recovery mechanisms

To detect the loss of a DT TPDU in TP4, of a DATA or a FIRST packet in XTP and of a TPDU containing data in TCP, all three protocols use a data numbering mechanism. TP4 uses a DT TPDU-based numbering whereas XTP and TCP use a byte-based numbering. This data numbering mechanism permits a sending transport protocol entity to provide to the peer receiving entity information about which TPDUs have been transmitted.

A receiving transport protocol entity must in turn communicate to the peer sending entity information about which TPDUs have been successfully received. All three TP4, XTP and TCP use a positive acknowledgement mechanism for this purpose, in association with the data numbering mechanism.

In all three protocols, the positive acknowledgements are cumulative in the sense that each positive acknowledgement carries the highest sequence number for which all the TPDUs with lower sequence numbers have been correctly received. An important advantage of the cumulative acknowledgements is that they give redundant error control information, thus reducing the problems that are caused by the loss of acknowledgements. On the other hand, a problem is that if TPDUi is lost, then each acknowledgement following this event contains the sequence number i until TPDUi arrives. These acknowledgements contain no information about the sequence number of the TPDU that triggered them.

Unlike what happens in TP4 today and in TCP, an XTP entity may in addition implement a selective positive acknowledgement mechanism. The selective positive acknowledgement mechanism in XTP is combined with the accumulative positive acknowledgement mechanism. Let us note however that, in the enhanced version of the ISO Transport Protocol which is currently being circulated for ballot as DIS 8073 [5], a selective positive acknowledgement mechanism is optionally provided for the class 4.

In TP4 today and in TCP, the policy for returning the positive acknowledgements is under the responsibility of the receiving protocol entity. Indeed, even though the transmission of positive acknowledgements by the receiving protocol entity is influenced by the arrival of TPDUs from the peer sending protocol entity, the positive acknowledgements in TP4 and in TCP are not solicited explicitly by this peer sending entity. Let us note however that, in the enhanced version of the ISO Transport Protocol [5], the possibility for a sending transport entity to request an immediate acknowledgement from the peer receiving transport entity is optionally provided.

In XTP, it is necessarily up to the sending XTP entity to choose the most appropriate policy for sending back the acknowledgements. The receiving XTP entity normally sends back an acknowledgement if and only if it has been demanded explicitly by the peer sending XTP entity. Under special condition however, the sending XTP entity may authorize the peer receiving XTP entity to return acknowledgements even when they are not requested. This will be explained in more detail below.

The loss of a DT TPDU in TP4, of a DATA or a FIRST packet in XTP and of a TPDU containing data in TCP is always recovered by means of a retransmission mechanism. Thus, on receipt of an acknowledgement, the sending protocol entity updates the pertinent state information and, in case of a positive acknowledgement, also discards the acknowledged user data that were retained for a possible retransmission.

Since the positive acknowledgements in TP4 and in TCP inform of the correct receipt of TPDUs but cannot report the loss of TPDUs explicitly, the retransmissions at the sending protocol entity are always based on timeouts.

5.2.1. Lost TPDUs detection and recovery mechanisms in TP4

The <TPDU-NR> field in the transport header of a DT TPDU contains the sequence number of the DT TPDU.

In general, TP4 assumes a network service provider which could deliver the TPDUs (and in particular the DT TPDUs) relating to a same TC out-of-sequence. That is why a receiving transport entity can send back only positive acknowledgements and not negative ones.

There is no “piggy-backed” control in TP4. This means that specific AK TPDUs are required to acknowledge the DT TPDUs.

The <YR-TU-NR> field in an AK TPDU contains the sequence number of the DT TPDU next expected in the reverse simplex stream relating to the TC. It is a cumulative positive acknowledgement.

Every DT TPDU is retained until it is acknowledged by an AK TPDU and is retransmitted on timeout.

The specification of TP4 does not impose any particular acknowledgement policy:

1) Considering that the positive acknowledgements are cumulative in TP4, the receiving transport entity may decide either to acknowledge each DT TPDU separately or to acknowledge several DT TPDUs with a single AK TPDU.
   Let us notice however that, at the time of TC set-up, an upper bound is negotiated for the maximum time

which can elapse between the receipt of an in-sequence DT TPDU by the transport entity and the transmission of the corresponding AK TPDU.

2) A sending transport entity may manage the timers on a per-TC basis or on a per-DT TPDU basis.
If the management of the timers is performed on a per-TC basis, two retransmission methods are possible when the timeout expires for a TC on which the highest sequence number received by the sending transport entity in a returned AK TPDU is i. The more conservative method is to retransmit the DT TPDU which has the sequence number i and then to wait for the next AK TPDU. The more aggressive method is to retransmit all the DT TPDUs between the DT TPDU with the sequence number i and the DT TPDU which was being transmitted when the timeout expires. The second method corresponds to a go-back-N retransmission.

Lastly, the timeout value can either be fixed once and for all by the implementors or be adjusted dynamically. In general, in order to maintain acceptable performance, the timeout value will have to be adjusted dynamically as a function of the current round-trip time. This could be done for example by means of the well-known algorithm that is implemented in TCP. Let us stress however that, over the years of running TCP/IP in the (D)ARPA Internet, many problems associated with the TCP retransmission timers have been encountered [13].

The upgraded version of the ISO Transport Protocol [5] provides for an optional selective positive acknowledgement mechanism in the class 4 only and for an optional request acknowledgement mechanism in the classes 1, 3 and 4. The use of these two options is negotiated on a per-TC basis at the time of TC set-up.

If the use of the selective positive acknowledgement mechanism is selected for a TC, the receiving transport entity at one end of this TC may put selective acknowledgements in addition to the cumulative acknowledgements in the returned AK TPDUs.

If the use of the request acknowledgement mechanism is selected for a TC, the sending transport entity at one end of this TC may request an immediate AK TPDU from the peer receiving transport entity by setting the ROA (Request Of Acknowledgement) flag in the transport header of an outgoing DT TPDU. Nevertheless, the usual policy for sending back AK TPDUs is not disabled.

The lost TPDUs detection and recovery mechanisms in TP4 have been designed with high-error-rate networks in mind. An AK TPDU must be sent back for every transmitted DT TPDU or at least for every group of a few transmitted DT TPDUs. The transmission of a lot of AK TPDUs degrades the performance because the transport entity which sends back AK TPDUs must interrupt its normal transmission of DT TPDUs to transmit each returned AK TPDU and because the transport entity which receives AK TPDUs must interrupt its normal transmission of DT TPDUs to process the received AK TPDUs.

Moreover, there is no selective retransmission mechanism in TP4. Therefore, when a timeout expires, either only the DT TPDU with the lowest sequence number which is not yet acknowledged or all the DT TPDUs which are not yet acknowledged are retransmitted. If only the DT TPDU with the lowest sequence number which is not yet acknowledged is retransmitted, one round-trip time is wasted while waiting for the next AK TPDU. On the other hand, retransmitting all the TPDUs which are not yet acknowledged is not necessarily a good solution. In fact, in the low-error-rate HSLANs on which we focus, the loss of a TPDU is often the indicator of a local congestion. The retransmission of all the DT TPDUs that are not yet acknowledged could thus worsen the problem instead of solving it.

5.2.2. Lost packets detection and recovery mechanisms in XTP

In XTP, the error control is split between the remote sending XTP entity and the local receiving XTP entity. As shown in Figure 5-1, the state of the input queue of a receiving XTP entity is defined by three state variables: the dseq, rseq and hseq variables. These three state variables belongs to the local connection context, i.e. to the local state information record for the connection.

The dseq value is the sequence number of the next data byte to be delivered to the receiving user. All data bytes which have a sequence number less than dseq have already been transferred to the user. The rseq value is one greater than the highest monotonic sequence data byte ever received. The hseq value is one greater than the highest sequence data byte ever received, possibly with some intermediate “holes” (or “gaps”) due to transmission errors and/or dropped packets. If all the data bytes received consecutively without errors have already been delivered to the user: dseq = rseq. If there is no gaps: rseq = hseq.

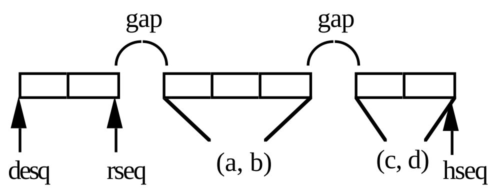

Fig. 5-1 Description of the state of the input queue of an XTP entity

The dseq value is placed in the <Dseq> field of the XTP header of all the packets, i.e. the DATA (“piggy-backed” control) and non-DATA packets, transmitted in the reverse simplex stream relating to the same XTP connection. The rseq value is placed in the <Rseq> field of the control segment of the CNTL packets only (no “piggy- backed control”).

The state of the input queue of the local receiving XTP entity is recorded by a CNTL packet which is sent back to the peer sending XTP entity in two cases:

1) on demand from the remote XTP entity, when the SREQ or the DREQ flag is received in the <flags> field of the XTP header of an incoming packet,
2) when the local receiving XTP entity detects a sequence gap in its input data stream, provided however that the fast (or aggressive) negative acknowledgement mode is selected by the remote sending XTP entity, i.e. provided that the FASTNACK flag is set in the <options> field of the XTP header of the packets that arrives from the remote sending XTP entity.

If the peer sending XTP entity does not enable the fast negative acknowledgement mode, the local receiving XTP entity is not allowed to send back a CNTL packet when it detects a sequence gap in its input data stream.

The peer sending XTP entity is responsible for setting the SREQ or the DREQ flag in the XTP header of an outgoing packet.

- Whenever the local receiving XTP entity sees the SREQ flag in an arriving packet, it immediately sends a CNTL packet in reply that records the current state of its input queue.
- Whenever the local receiving XTP entity sees the DREQ flag in an incoming packet, a DREQ indicator is placed in the input queue after the user data that stem from the packet. As the data in the input queue are moved to user buffers, the DREQ indicators in the queue will be discovered and a CNTL packet generated in reply for each one.

So the remote sending XTP entity may request the acknowledgement of data either when they are received (SREQ flag) or when they are delivered to the user (DREQ flag).

XTP has a synchronizing scheme called synchronizing handshake. Each connection context has a sync state variable and an echo state variable. The sync variable is a local counter which is incremented for every outgoing DATA packet. As an alternative, the sync variable could also be incremented whenever the transmission switches from control packets to data packets, creating a new epoch. The value of the echo variable is the last communicated value of the sync variable from the peer context. The current sync value is placed in the <sync> field of the XTP header in every outgoing packet. The echo value is placed in the <echo> field of the control segment in every outgoing CNTL packet. A synchronizing handshake consists in exchanging two CNTL packets. First, a (CNTL, SREQ) packet, that records the current values of the local sync and echo variables, is sent. When this CNTL packet arrives at the remote receiving XTP entity, the value of the remote echo variable is updated with the value in the <sync> field of the packet. Second, a CNTL packet is received in response to the SREQ request with a value in the <echo> field that matches the value of the local sync variable. The synchronizing handshake permits to measure the round-trip time thanks to the <time> and <techo> (time echo) fields in the control segment of the CNTL packets. The value in the <time> field of the (CNTL, SREQ) packet is simply replicated in the <techo> field of the returned CNTL packet.

The sending XTP entity uses a timer, WTIMER, on a per-XTP-connection basis in order to detect when a requested CNTL packet is not received within the expected period. WTIMER is started whenever a packet with the SREQ or the DREQ flag turned on is sent. The WTIMER timeout value should be adjusted as a function of to the current round-trip time.

When WTIMER is started, one of the following conditions will hold afterwards:

1) A CNTL packet is received in reply before the timeout expires and no errors are indicated. No special action is done. The timer is simply stopped.
2) A CNTL packet is received in return before the timeout expires but errors are indicated. In this case, the CNTL packet indicates which output sequence number spans need to be retransmitted. The timer is stopped.
3) No CNTL packet is received during the timeout interval. In this case, the sending XTP entity begins to send (CNTL, SREQ, sync = a given value) packets until a CNTL packet is received back with a value in the <echo> field that matches the local synchronizing value. When the protocol goes into a repetitive loop attempting to complete the synchronizing handshake, the time interval between attempts increases exponentially. Although this is more of an issue for protocols like TCP, that retransmit data with each attempt, the XTP designers deem this exponential increase important to minimize the retry traffic and give congested systems an opportunity to recover.

An essential characteristic of XTP is that the routing mechanism present in the routing part of XTP guarantees that the packets (and in particular the DATA packets) relating to a same XTP connection always arrive in-sequence at the receiving XTP entity. In reality, the packets relating to a same XTP connection can never be reordered in case of an isolated LAN or of a stable configuration of LANs interconnected by bridges. In case of LANs interconnected by routers, the packets can never be reordered provided that the routing part of XTP is in charge of the routing in all the routers. We will briefly explain this fundamental property of the routing function of XTP in § 7.3.1.

Making the transport part of XTP work with a classical internet protocol is an intricate problem because of the tight relation that exists between the error control mechanisms of the transport part of XTP on the one hand and this property of the routing function of XTP on the other hand. Indeed, the error control mechanisms in XTP have been designed and especially optimized on the premise that the packets relating to a given XTP connection are to arrive in the right order. Since the Revision 3.5 however, the possibility of using XTP when a classical datagram-oriented internet protocol is in charge of routing has really been taken into account. This has been done by introducing the possibility of disabling the fast acknowledgement policy (FASTNACK flag reset). The fast acknowledgement policy should be disabled when the packets can be reordered due to the fact that the routing is under the responsibility of a classical internet protocol. Error control cannot be optimal anyhow but performance degrade less severely when the fast negative acknowledgement policy is disabled.

In XTP, the retransmissions are based on the receipt and the interpretation of CNTL packets and not on the expiration of timeouts like in TP4. The three modes of operation for error control in XTP are the NOERR mode, the go-back-N retransmission and the selective retransmission.

The easiest mode to understand is surely the NOERR mode. Every retransmission is disabled by the NOERR flag in the XTP header. The receiving XTP entity always pretends that it has received everything and thus always sets rseq to hseq, masking the possible gaps in its input queue. The sending XTP entity will see no errors but will still observe the flow and rate controls.

Otherwise, the error control in XTP allows both go-back-N and selective retransmission strategies with the same basic protocol structure. The go-back-N retransmission mechanism uses only the <Rseq> field of the control segment of a CNTL packet to indicate where the retransmission must start in the data stream. The selective retransmission mechanism uses in addition the <nspan> and <spans> fields in the control segment of a CNTL packet. The <nspan> field gives the number of active pairs of sequence numbers, also called “spans”, that are present in the variable-length <spans> field. These pairs of sequence numbers are used to mark the packets that have been correctly received beyond rseq in the packet stream. A sending XTP entity always retains the transmitted data until they are acknowledged by the content of the <Dseq> field in a received CNTL packet.

The selective retransmission technique can improve the throughput in high-error-rate circumstances accompanied by low-bandwidth and/or long-delay characteristics. In the high-bandwidth low-error-rate HSLANs in which we are interested, there may be little or no performance difference between retransmitting an entire sequence of packets or simply retransmitting the dropped packets. Therefore, the simpler go-back-N procedure may be better. Since the circumstances dictate which policy should be used, XTP has been careful to define the go-back-N retransmission mechanism as a compatible subset of the selective retransmission mechanism. This ensures interoperability between end systems that may have implemented differing levels of error control.

The acknowledgement mechanism of XTP is better adapted than that of TP4 to the high-speed low-error-rate HSLANs. Indeed, in case of an error-free transmission, the rate at which a sending XTP entity receives CNTL packets back is equal to the rate at which this sending XTP entity requests CNTL packets by setting the SREQ or the DREQ flag in outgoing packets. If the SREQ or the DREQ requests in the outgoing packets are spaced out, the CNTL packets will also be spaced out and each CNTL packet will acknowledge a large group of DATA packets.

5.3. Duplicate or out-of-sequence TPDUs detection and recovery mechanisms

To detect the duplication or the out-of-sequence arrival of a DT TPDU in TP4, of a DATA or a FIRST packet in XTP and of a TPDU containing data in TCP, all three protocols use the data numbering mechanism. A duplicate is simply discarded.

Of course, when the in-sequence arrival of the XTP packets at the receiving XTP entity is guaranteed, duplicate or out-of-order packets are possible only in case of retransmissions.

6. End-to-end flow control mechanism

The end-to-end flow control mechanism in the transport layer allows a receiving transport protocol entity to control the flow of incoming data on a per-transport-connection basis by acting on the remote sending transport protocol entities.

All three TP4, XTP and TCP use a sliding window end-to-end flow control mechanism in association with the data numbering error control mechanism. As seen in § 5.2., TP4 uses a DT TPDU-based numbering whereas XTP and TCP use a byte-based numbering.

6.1. End-to-end flow control mechanism in TP4

The size of the window is expressed in DT TPDUs. The lower edge and the size of the window are communicated by the receiving transport entity to the peer sending transport entity respectively in the <YR-TU-NR> field and in the <CDT> field of the AK TPDUs (no “piggy-backed” control) which are sent in the reverse simplex stream relating to the same TC.

In general, TP4 assumes a network service provider which could deliver out-of-order the TPDUs (and in particular the AK TPDUs) relating to a same TC. In order to allow a sending transport entity to properly sequence a series of received AK TPDUs that all contain the same acknowledgement number in their <YR-TU-NR> field, and thereby to use the correct credit value, the AK TPDUs may contain a subsequence parameter. When the sending transport entity recognizes an out-of-sequence AK TPDU, it simply discards it.

As already said in § 5.2.1, the specification of TP4 does not impose any particular policy for returning the AK TPDUs. However, an AK TPDU which contains up-to-date window information must be sent back immediately when a DT TPDU is received outside the allocated window because of the loss of a previously sent AK TPDU. Besides, there is an upper bound on the maximum time a transport entity may wait before transmitting up-to-date window information. This bound is one of the tuning parameters in an implementation.

The policy for adjusting the size of the transmit window is also left unspecified in TP4. In practice, the size of the window is often allocated in a heuristic way by the receiving transport entity, depending upon the implementation.

During the establishment phase of a TC, the use of the expedited data option is negotiated between the two peer session entities. If this option is offered on the TC, a session entity may choose to send data in an expedited TSDU rather than in a normal TSDU. The size of an expedited TSDU cannot be greater than 16 bytes. The TS provider ensures that an expedited TSDU will not be delivered to the receiving session entity after any subsequently submitted normal TSDU or expedited TSDU on the same TC.

For this purpose, TP4 allows no more than one Expedited Data TPDU (ED TPDU) to remain unacknowledged at any time and for each simplex stream relating to a TC. TP4 also interrupts the flow of normal DT TPDUs until the ED TPDU is acknowledged by an Expedited Acknowledge TPDU (EA TPDU). When the EA TPDU is received, the flow of normal DT TPDUs may resume, unless one or more ED TPDUs are still waiting to be sent. An ED TPDU is retained until acknowledgement and retransmitted on timeout just as a DT TPDU. Let us stress that the ED TPDUs have their own numbering. Thus they are not constrained by the end-to-end flow control and can be sent even if the transmit window is closed.

6.2. End-to-end flow control mechanism in XTP

The size of the window is expressed in bytes. Its lower edge is communicated by the receiving XTP entity to the peer sending XTP entity in the <Dseq> field of the XTP header of all the packets ("piggy-backed" control) which are sent in the reverse simplex stream relating to the same XTP connection. The upper edge of the window is transmitted in the <alloc> field of the control segment of the CNTL packets only (no "piggy-backed" control).

As already stated in § 5.2.2, all the packets (and in particular the CNTL packets containing relevant window information) relating to a same XTP connection are always delivered in-sequence to the receiving XTP entity when there is no internet routing but also when there is an internet routing under the responsibility of the routing part of XTP. Thus the last received window information is necessarily the most recent one.

Since the upper edge of the window can be communicated by way of CNTL packets only, a sending XTP entity must request CNTL packets from the peer receiving XTP entity, by means of SREQ or DREQ, to obtain up-to-date information about the upper edge of the window. So, especially in the networks with a high T*RTD product, the size of the allocated window is a fundamental factor from the performance standpoint.

Usually, a receiving XTP entity will move alloc to the right of dseq based on a constant C or on a heuristic algorithm. Nonetheless, XTP also provides an explicit reservation mode for bulk transfers. In this mode, the RES flag is set in the XTP header of the packets to request from the receiving XTP entity the allocation of a transmit window whose size strictly represents the number of bytes available in the read buffers committed by the user for the XTP connection. In reservation mode, a CNTL packet with up-to-date window information should be sent back whenever the receiving XTP entity is given more buffer space for the XTP connection.

In the Revision 3.6, the XTP entity of the connection initiator may indicate, by setting the NOFLOW flag in the XTP header of the outgoing packets, that it does not observe the window-based flow control. This mode is intended for use on rate-controlled connections or in situations where unconstrained operation is desired. The XTP entity that receives the FIRST packet may decline to accept the NOFLOW mode by rejecting the FIRST packet and returning a DIAG packet in reply.

Normally, the access to the input and output queues of packets in an XTP entity obeys the FIFO discipline. Nevertheless, XTP may use a preemptive priority queue discipline for both output and input. In fact, each active context may have a current sort value supplied by the user. The XTP implementations are not required to support sorting/prioritizing but, in an XTP entity which supports sorting/prioritizing, all the packets that can be sent at the current highest priority must be sent before the packets at a lower priority. The priority queue mechanism is not completely described yet in the current Revision 3.6 of the specification of XTP. Therefore, it will not be explained in more details in the present paper.

A major feature of XTP however is that there is no way to bypass the flow control, even when the priority queue mechanism is used. This situation in XTP is quite different from those in TP4 and in TCP, where the sending of expedited data and urgent data is not constrained by the flow control.

In today's HSLANs that are generally characterized by high T*RTD products, an expedited data mechanism which blocks the sending of the normal data during at least one round-trip time to transmit expedited data can have an important negative impact on the global performance of an exchange of data over a transport connection. That is why such a mechanism does not attract any more a lot of interest.

7. Congestion control mechanism

The rate at which the data may be transmitted on a given transport connection should be driven by the rate at which the receiving transport protocol entity can receive and process the TPDUs containing the data and then can forward these data to the target user. But the rate at which the data may be transmitted on a given transport connection should also be driven by the sustainable rate through the network service.

The end-to-end flow control is not able to solve the problems of congestion which can occur inside the network service itself, e.g. local congestions in routers. Fighting against the congestions inside the network service is the function of the congestion control.

Up to now, we have assumed an isolated LAN or a set of LANs interconnected by bridges only. From now onwards, we will also include in our study the configurations where LANs are interconnected by routers.

In case of a set of LANs interconnected by routers, TP4 is assumed to have access to the LLC Type 1 Service through the full CLNP (or at least the Non-Segmenting Protocol Subset of CLNP) plus the SNDCFs. The routing is then under the responsibility of the standard ISO CLNP.

XTP is still assumed to attack the LLC Type 1 Service through SNAP. When XTP is used in LANs interconnected by routers, it must be installed in all the end systems but also in all the routers.

7.1. Congestion control mechanism in TP4

The specification of TP4 does not provide a congestion control mechanism. This limitation will have to be wiped out to ensure the viability of the OSI standards in the HSLANs.

7.2. Congestion control mechanism in TCP

At the start, TCP/IP had no congestion control mechanism and many problems of congestion soon occurred while running TCP/IP in the (D)ARPA Internet. The problems of congestion were largely due to the inability of IP to perform an appropriate congestion control. In fact, the algorithm for adjusting the timeout value of the retransmission timers in TCP relies on the assumption that the losses of TPDUs containing data are random and rare, let us say a 1≈2% loss ratio [13]. Unfortunately, this assumption is invalidated when IP tries to solve a congestion by systematically dropping packets in the congested router. TCP cannot help IP to solve the problems of congestion inside the network while still keeping good performance, unless TCP includes its own congestion control mechanism.

Several timeout-based congestion control mechanisms which can apply to TCP/IP have been developed [14-16]. These timeout-based congestion control mechanisms rely on the basic idea that, in present low-error-rate networks, a packet loss is generally a good indicator of a congestion. Therefore, whenever a timeout associated with a TPDU expires, the load on the network should be reduced. Later on, if there is no further loss, the load should be increased slowly. The available mechanisms differ mainly in the policies that are chosen to decrease and then to increase again the traffic load. The timeout-based congestion control mechanisms are characterized by a "slow start" and by "slow restarts" after congestions. So they are not really suitable for the present and future networks which are very promising from the throughput point of view, since they lead to a blatant underuse of the available bandwidth at the start and at each restart. Nevertheless, they have at least a major advantage: the feedback to signal a potential congestion is implicit and thus does not increase the traffic during a congestion.

The congestion control mechanism that is implemented in XTP belongs to the routing part of XTP and not to the transport part of XTP. Considering this paper pertains to a comparative study which is directed towards transport mechanisms, we will only give a short description of this routing part of XTP.

7.3. Brief description of the routing part of XTP

The two basic mechanisms that are implemented in the routing part of XTP are the routing proper mechanism and the rate control mechanism.

Up to the Revision 3.5, the routing part of XTP had also a fragmentation mechanism. The fragmentation mechanism allowed the sending XTP entity or an intermediate XTP router to fragment one XTP packet (i.e. one XTP-PDU) into several XTP fragments (i.e. several smaller XTP-PDUs) in a way which did not imply any new calculation of an integrity checksum over the data of the original XTP packet. This fragmentation mechanism of XTP has been removed from the Revision 3.6 to simplify the design.

7.3.1. Routing proper mechanism in XTP

Let us now examine what happens when an XTP connection must be established between two end systems A and B that are on different LANs interconnected by routers. In this section, it is assumed that routing is under the responsability of the routing part of XTP.

The FIRST packet sent to open the XTP connection (see [1] for more details), which has to travel through one or more XTP routers to reach its destination, marks out a route in the network. All the packets that will circulate afterwards on both simplex streams relating to this XTP connection will necessarily travel along the route followed previously by the FIRST packet. That is why the routing function of XTP always guarantees the in-sequence arrival of the packets pertaining to a same XTP connection. In fact, in the network, a packet cannot go past another packet transmitted on the same XTP connection previously, unless the discipline to access the input and output queues of packets is not FIFO.

The relation between the concept of 'route', which is a concept of the routing part of XTP, and the concept of 'connection', which is a concept of the transport part of XTP, is not one-to-one. Indeed, an XTP connection is always linked to a unique route in the network but a same route can be shared by several XTP connections established between the same two end systems. Thus distinct XTP connections that are set up between the same two end systems may use distinct routes in the network or may share a same route.

The selection of a route at the time of connection establishment is based on the route variables used in the <route> field of the XTP header in the packets.

7.3.2. Rate control mechanism in XTP

XTP has a congestion control mechanism based on rate control. The rate control mechanism has to be included in the routing part of XTP seeing that the rate control is carried out not only by the XTP entities in the end systems but also by the intermediate XTP routers.

A clear-cut distinction between the concept of 'XTP connection' and the concept of 'route' is necessary to study the rate control mechanism in XTP because the rate control in XTP is performed on a per-route basis rather than on a per-XTP-connection basis. This means that route sharing between several XTP connections also implies rate sharing. The output rates may be controlled separately on both directions of a route.

A receiving XTP entity can exercise rate control on the incoming direction of a route by providing rate control information in the <rate> and <burst> fields of the control segment of CNTL packets sent back to the peer sending XTP entity. The rate value specifies the maximum data rate in bytes per second. The burst value specifies the maximum number of bytes to be transmitted in a burst of packets. Thus the rate value divided by the burst value gives the number of authorized burst-size data transmissions per second.

Any possible intermediate XTP router along the route is allowed to decrease the values in the <rate> and <burst> fields of a CNTL packet as it passes through. Besides, an XTP router does not have to wait for a passing CNTL packet to send rate control information to a sending XTP entity in an end system. Indeed, it can decide to send a RCNTL packet (Router-generated CNTL packet) with suitable rate and burst values.

In theory, the rate control mechanism in XTP permits to prevent the congestion phenomena in the XTP routers since a router can record but also modify the values of the rate and burst variables relating to all the routes that pass through it. In practice, the rate control mechanism brings about some problems like the accurate measure of the resources that are put at the disposal of an XTP router. Moreover, even though the rate control mechanism in XTP permits to fight against local congestions in the XTP routers, the bridges that interconnect LANs at the MAC level cannot take part in the rate control. Thus the rate control mechanism of XTP is powerless to solve the problems of congestion in a network where several LANs are connected by bridges.

8 . Multicast mechanism

8.1. Multicast mechanism in TP4

TP4 developed when the communication infrastructure was still dominated by switched lines, leased lines and packet switching WANs. The network service provider did not offer broadcast and/or multicast facilities. Therefore, including a multicast mechanism at the transport level was really not a prerequisite.

A TC is opened between two TSAPs and allows the transfer of TSDU only between the two session entities that are associated by this TC.

8.2. Multicast mechanism in XTP

XTP provides a multicast mechanism. The underlying service must support at least a broadcast capability. An address filtering within XTP can be used to layer a multicast address architecture on top of a broadcast transmission.

Thanks to this multicast mechanism in XTP, the multicast facility in LANs can be extended up to the interface of the service provided by XTP. On an isolated LAN or on LANs interconnected by bridges, this extension certainly seems interesting. On LANs interconnected by XTP routers, offering a transport-level service with the multicast facility is still attractive as far as this does not translate into killing performance. Most of the time, the XTP routers are the critical elements from the performance standpoint.

The multicast mechanism in XTP is a mechanism for sending multicast packets from a single multicast sending XTP entity to a set of multicast receiving XTP entities. The error control may remain enabled or be disabled by setting the NOERR flag in the XTP header of all the outgoing packets.

The multicast packets obey the same syntax rules as the non-multicast packets: the XTP header, the XTP trailer and the information or control segment are identical.

The multicast packets differ from the unicast ones in the following ways:

- all the multicast packets have the MULTI flag turned on in their <options> field in the XTP header while the unicast packets have this flag reset,
- the multicast FIRST packets utilize group addresses rather than individual addresses in their address subsegment,
- the multicast packets always use SREQ, not DREQ,
- SREQ may be set only in CNTL packets, not in DATA packets.

When a user gives output commands to the underlying XTP entity with the multicast mode selected via configuration parameters, the MULTI flag is set by this underlying XTP entity in the XTP header of all the outgoing packets. The other aspects of the transmission remain unchanged: the initial output packet will be of type FIRST whereas subsequent ones will be of type DATA and, furthermore, the command options bitflags in the XTP header will operate in the usual way.

A user that wishes to receive multicast transmissions sets up a listening context that will match any multicast address of interest and will then become active. When active, this context accepts the multicast transmission and delivers the data to the user.

As said above, there are two error control methods available in the XTP multicast mode :

1) No error control :
   The NOERR flag is set in all packets. The multicast receiving XTP entities discard the damaged input data and deliver only the correct data to their users. No further actions are taken. That is the standard procedure for the NOERR processing.
2) Go-back-N retransmission of corrupted packets :
   The multicast with go-back-N retransmission method is used whenever it is not inhibited by the NOERR flag. The multicast receiving XTP entities monitor the incoming sequence numbers using the same mechanism as described in § 5.2.2.
   In multicast mode however, a sending XTP entity must positively associate the returned CNTL packets with past events if a continuous output streaming is desired. This can only be accomplished by matching the echo values that are returned in the CNTL packets with local sync values. This in turn requires that a multicast sending XTP entity set the SREQ flag in CNTL packets only, not in FIRST or DATA packets. The CNTL packets that are sent by the multicast receiving XTP entities in response to received SREQ are multicast to the group of multicast receiving XTP entities as well as to the multicast sending XTP entity. This is necessary for the damping and slotting algorithms. With the damping, a multicast receiving XTP entity does not send a CNTL packet if it would duplicate a CNTL packet already observed on the network. This algorithm reduces the total number of CNTL packets that are transmitted by the number of duplicates that are cancelled. This is done at the expense of having the multicast receiving XTP entities read all the CNTL packets. The slotting technique impose a random delay before sending SREQ-induced CNTL packets. The results of slotting are twofold. First, the number of CNTL packets is distributed in time rather concentrated in time. Second, the distribution of CNTL packets in time increases the probability that an implementation will be able to exercise the damping.
   The generation and the interpretation of the CNTL packets require special considerations in the multicast mode. In fact, in the unicast mode, a dseq value that is received in the <Dseq> field of a CNTL packet sent back to the sending XTP entity indicates the correct delivery of data to the user by the receiving XTP entity. But in multicast mode, a returned dseq value only indicates that one of the multicast receiving XTP entities has correctly received the data. The

bucket algorithm provides a strategy for sending SREQ requests, interpreting the returned CNTL packets and releasing the output buffers.

Such a multicast scheme does not work efficiently in all circumstances and in all environments. If there are many multicast XTP recipients on an error-prone network, there may be many multicast error-CNTL packets and consequently little forward progress. In some error-prone circumstances, it may be advisable to use the NOERR policy.

9. Conclusions

The design of both TCP and TP4 had as background the characteristics of the communication infrastructures and of the applications that existed in the late '70s and in the early '80s.

The design of the contemporary transport protocols should be strongly influenced by the tremendous technological evolution in the field of network communications as well as by the widening of the application needs that have occurred since then.

The emerging XTP is a novel approach to protocol design. The will of designing XTP as a protocol adapted to the new communication infrastructures and to the new applications is quite perceptible, through its data transfer mechanisms notably.

In order to make XTP a protocol better adapted than TP4 and TCP to new applications, the designers have concentrated significant efforts on a multicast mechanism and on a selectable error control mechanism.

XTP has an interesting multicast mechanism even though multicasting over a network is not necessarily compatible with maintaining acceptable performance in all circumstances and in all environments.

Also, the goal to make XTP a protocol better adapted than TP4 and TCP to the high-bandwidth low-error-rate requirements has influenced most of the data transfer mechanisms.

The transport part of XTP does not include any mechanism similar to the concatenation/separation mechanism in TP4 which many people considered to be completely obsolete today.

Simulations have put forward that error and flow controls are usually the most two harmful transport functions from the performance standpoint. As far as these two functions are concerned, XTP differs from the classical protocols TP4 and TCP less in the mechanisms used (in fact, XTP uses well-known mechanisms to implement error and flow controls) than in the way these mechanisms are used. The error and flow control mechanisms in XTP are better adapted to the properties of the high-speed low-error-rate networks in general, and of the HSLANs in particular.

In the routing part of XTP, the major improvement is the introduction of a rate control mechanism to avoid congestion in the routers. An important feature of the routing mechanism of XTP is that all the XTP packets relating to a same XTP connection necessarily follow the same route in the network, which ensures the in-sequence delivery of the packets to the receiving XTP entity.

10. References

[1] Y. Baguette, A. Danthine: *Comparison of TP4, TCP and XTP - Part 1: Connection Management Mechanisms*. «ETT», Vol. 3, No. 5, Sept./Oct. 1992, pp. ###-###.

[2] ISO/IEC JTC1: *IPS - OSI - Connection-Oriented Transport Protocol Specification*. ISO 8073, 1988.

[3] Protocol Engines, Inc.: *XTP Protocol Definition - Revision 3.6*. Ref.: PEI 92-10, Jan. 11, 1992.

[4] Network Information Center: *Transmission Control Protocol*. J.B. Postel, ed., NIC Standard RFC-793, Sept. 1981.

[5] ISO/IEC JTC1: *IT - TIES - Connection-Oriented Transport Protocol Specification*. Forwarded as DIS 8073 to ITTF (voting terminated on June 5, 1992).

[6] ISO/IEC JTC1: *IPS - LAN - Part 2: Logical Link Control*. ISO 8802-2, 1989.

[7] ISO/IEC JTC1: *IPS - DC - Protocol for Providing the Connectionless-Mode Network Service*. ISO 8473, 1988.

[8] ISO/IEC JTC1: *IPS - DC - Protocol for Providing the Connectionless-Mode Network Service, Addendum 1: Provision of the Underlying Service Assumed by ISO 8473*. ISO 8473 DAD1, 1988.

[9] D.R. Cheriton, S.E. Deering: *Host Groups: a Multicast Extension for Datagram Internetworks*. 9th Data Communication Symposium, IEEE Computer Society and ACM SIGCOMM, Sept. 1985.

[10] ISO/IEC JTC1: *IPS - DC - Network Service Definition, Addendum 1: Connectionless-Mode Transmission*. ISO 8348 ADD1, 1987.

[11] Network Information Center: *Internet Protocol*. J.B. Postel, ed., NIC Standard RFC-791, Sept. 1981.

[12] Protocol Engines, Inc.: *XTP Protocol Definition - Revision 3.5*. Ref.: PEI 90-120, Sept. 10, 1990.

[13] L. Zhang: *Why TCP Timers Don't Work Well*. ACM SIGCOMM '86 Symposium on Communications Architectures and Protocols, Stowe, VT, Aug. 5-7, 1986, pp. 397-405.

[14] V. Jacobson: *Congestion Avoidance and Control*. ACM SIGCOMM '88 Symposium, Stanford, CA, Aug. 16-19, 1988, pp. 314-329.

[15] R. Jain: *A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks*. «IEEE Journal on Selected Areas in Communications», Vol. SAC-4, No. 7, Oct. 1986, pp. 1162-1167.

[16] R. Jain: *A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks*. «ACM Computer Communication Review», Vol. 19, No. 5, Oct. 1989, pp. 56-71.

[17] Y. Baguette, A. Danthine: *Comparison of the Transport Mechanisms of TP4, TCP and XTP*. RACE technical report, Ref.: 60/ULg/CIO/PI/C/001/b1, March 13, 1992.