INTERNATIONAL STANDARD

ISO/IEC 14476-3

First edition
2008-04-15

Information technology — Enhanced communications transport protocol: Specification of duplex multicast transport

Technologies de l'information — Protocole de transport de communications amélioré: Spécification pour le transport duplex en multidiffusion

Reference number
ISO/IEC 14476-3:2008(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

CONTENTS

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14476-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*, in collaboration with ITU-T. The identical text is published as ITU-T Rec. X.607 (02/2007).

ISO/IEC 14476 consists of the following parts, under the general title *Information technology — Enhanced communications transport protocol*:

— *Part 1: Specification of simplex multicast transport*

— *Part 2: Specification of QoS management for simplex multicast transport*

— *Part 3: Specification of duplex multicast transport*

— *Part 5: Specification of N-plex multicast transport*

Introduction

This Recommendation | International Standard specifies the Enhanced Communications Transport Protocol (ECTP), which is a transport protocol designed to support Internet multicast applications running over multicast-capable networks. ECTP operates over IPv4/IPv6 networks that have the IP multicast forwarding capability with the help of IGMP and IP multicast routing protocols. ECTP could possibly be provisioned over UDP.

ECTP is designed to support tightly controlled multicast connections in simplex, duplex and N-plex applications. This third part of ECTP (ITU-T Rec. X.607 | ISO/IEC 14476-3) specifies the protocol mechanisms for reliability control in the duplex case. ECTP also provides QoS management functions for stable management of the QoS of the connection users. The procedures for QoS management of the duplex case will be defined in the duplex QoS management specification (ITU-T Rec. X.607.1 | ISO/IEC 14476-4).

In the duplex multicast connection, the participants are classified into one TC-Owner and many TS-users. TC-Owner will be designated among the TS-users before the connection begins. In the duplex multicast connection, the two types of data transports are supported: multicast data transport from TC-Owner to all the other TS-users and unicast data transport from TS-users to TC-Owner. After the connection is created, TC-Owner can transmit multicast data to the group, whereas each TS-user is allowed to send unicast data to TC-Owner just after it gets a token from the TC-Owner.

In ECTP, TC-Owner is at the heart of multicast group communications. It is responsible for overall connection management by governing the connection creation and termination, connection pause and resumption and the late join and leave operations.

The duplex multicast connection specified in ECTP-3 is targeted to the multicast applications in which the TC-Owner (a single multicast sender) transmits the data information to all the other TS-users, and some of the TS-users respond to the multicast sender with the unicast feedback data. Basically, the duplex multicast transport will be well suited to the one-to-many multicast applications that need the unicast feedback channels from some TS-users (e.g., remote education, Internet broadcasting, etc). For example, in a remote education application, the multicast sender (lecturer) transmits the data such as voice, text and image to the student group, whereas some of the students may respond to the lecturer with the unicast data like questions for confirmation.

It is noted that this duplex multicast connection can also be used for the 'some-to-many' multicast applications (e.g., a panel conferencing) in which a few of TS-users want to send multicast data to the group. In this scenario, the multicast data from the TS-users may first be delivered to the TC-Owner by unicast, and then TC-Owner will transmit the received unicast data to the group by multicast. For example, in the panel conferencing, some of the TS-users may act as a panel and transmit multicast data via TC-Owner (the conference convener) to the listener group. The detailed use of the duplex multicast connection depends on the applications of this duplex multicast transport protocol.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – Enhanced communications transport protocol: Specification of duplex multicast transport

1 Scope

This Recommendation | International Standard specifies the Enhanced Communications Transport Protocol (ECTP), which is a transport protocol to support Internet multicast applications over the multicast-capable IP networks.

This Recommendation | International Standard specifies the ECTP part 3 (ECTP-3) for the duplex multicast transport connection in which the participants are classified into one TC-Owner and many TS-users. The duplex multicast transport connection supports two kinds of data transport: the multicast data transport from TC-owner to all the other TS-users and the unicast data transport from TS-users to TC-Owner. A TS-user is allowed to send unicast data to TC-Owner, only if it gets a token from TC-Owner.

This Specification describes the protocol for supporting the duplex multicast transport, which includes the connection management (establishment, termination, pause, resumption, user join and leave) and the reliability control mechanisms for the multicast and unicast data transport. In particular, the protocol operations for the multicast data transport from TC-Owner to the TS-users will be designed with the congruency of the simplex multicast transport protocol (ECTP-1), as specified in ITU-T Rec. X.606 | ISO/IEC 14476-1.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

- ITU-T Recommendation X.601 (2000), *Multi-peer communications framework.*
- ITU-T Recomendation X.602 (2004) | ISO/IEC 16513:2005, *Information technology – Group management protocol.*
- ITU-T Recommendation X.605 (1998) | ISO/IEC 13252:1999, *Information technology – Enhanced communications transport service definition.*
- ITU-T Recommendation X.606 (2001) | ISO/IEC 14476-1:2002, *Information technology – Enhanced communications transport protocol: Specification of simplex multicast transport.*
- ITU-T Recommendation X.606.1 (2003) | ISO/IEC 14476-2:2003, *Information technology – Enhanced communications transport protocol: Specification of QoS management for simplex multicast transport.*

3 Definitions

This Recommendation | International Standard is based on the following definitions, which were specified in Enhanced Communications Transport Service (ITU-T Rec. X.605 | ISO/IEC 13252).

a) Transport connection: Simplex, duplex and N-plex;

b) TC-Owner and TS-users.

This Recommendation | International Standard uses the following terminologies specified in Enhanced Communications Transport Protocol: part 1 (ITU-T Rec. X.606 | ISO/IEC 14476-1).

a) control tree;

b) parent and children;

c) TO (Top Owner):

TO is a single sender of multicast data packets, which can transmit multicast data to the other TS-users, and it manages overall operations of ECTP-3. The TO will be designated among the TS-users before the connection begins, and the TO will do the functions of TC-Owner;

d) LO (Local Owner):

An LO is located on the control tree of ECTP-3. One or more LOs could be designated for scalable error recovery and status monitoring in ECTP-3. An LO is also a TS-user, which can also receive the multicast data from TO. LOs will be configured as a parent of the local groups through the control tree configuration in ECTP-3; and

e) LE (Leaf Entity):

An LE is a leaf node on the ECTP-3 control tree. It is a TS-user in the ECTP-3 connection, and it can receive multicast data sent by TO.

This Recommendation | International Standard also applies the following definitions:

a) SU (Sending TS-user):

Some of the ECTP-3 TS-users can send unicast data to the TO. A sending TS-user (SU) is a TS-user who gets a token from TO. Only the SU is allowed to send unicast data to TO. In other words, before sending unicast data, each user must request a token to TO.

b) Token:

It represents the right for a TS-user to transmit data. The TS-user who has a token is called a Sending TS-user (SU). The tokens are managed by TO.

c) Forward data channel:

It represents the multicast data channel from TO to the group members. TO sends multicast data to all the other group members over IP multicast address.

d) Backward data channel:

It represents the unicast data channel, in which the data packets flow from an SU to TO. An SU can send unicast data to TO over IP unicast address.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply, which includes the ECTP-3 packets:

ACK	Acknowledgment
CC	Connection Creation Confirm
CR	Connection Creation Request
CT	Connection Termination Request
DT	Data
HB	Heartbeat
HBACK	Heartbeat Acknowledgment
JC	Late Join Confirm
JR	Late Join Request
LE	Leaf Entity
LO	Local Owner
LR	User Leave Request
ND	Null Data
RD	Retransmission Data
SU	Sending TS-user
TC	Tree Join Confirm
TGC	Token Get Confirm
TGR	Token Get Request

TJ	Tree Join Request
TO	Top Owner
TRC	Token Return Confirm
TRR	Token Return Request
TS	Transport Services

5 Conventions

In this Recommendation | International Standard, the capital characters are used to represent a 'packet' of ECTP-3 (e.g., *CR* for Connection Creation Request packet), and the capital and italic characters are used for 'timers' or 'variables' used in ECTP-3 (e.g., *CCT* for Connection Creation Timer, and *AGN* for ACK Generation Number).

6 Overview

The Enhanced Communications Transport Protocol (ECTP) is a transport protocol designed to support Internet multicast applications. ECTP operates over IPv4/IPv6 networks that have the IP multicast forwarding capability with the help of IGMP and IP multicast routing protocols, as shown in Figure 1. ECTP could possibly be provisioned over UDP.

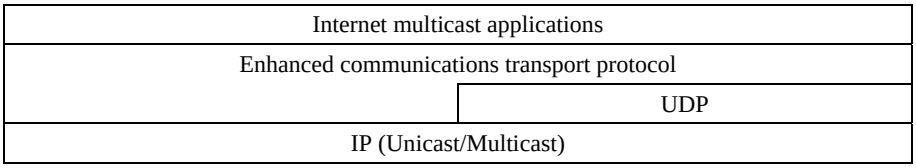

Figure 1 – ECTP model

This Recommendation | International Standard describes the protocol specification of the ECTP Part 3 (ECTP-3) for the duplex multicast connection. The duplex multicast connection is used for supporting multicast data transport between the participants that are classified into a single TC-Owner (TO) and the other TS-users. A duplex multicast connection supports the two types of data channels between the participants: *multicast data channel* (sent by TO toward all the other group members) and *unicast data channel* (sent by a TS-user to TO). Such a TS-user is called Sending TS-user (SU) in the ECTP-3.

Figure 2 illustrates these two types of data transport channels used in the duplex multicast connection. As shown in the figure, TO can transmit multicast data to the other TS-users over IP multicast (group) address. Some SUs may send unicast data to TO. The SU must first get a token from the TO before sending the unicast data.

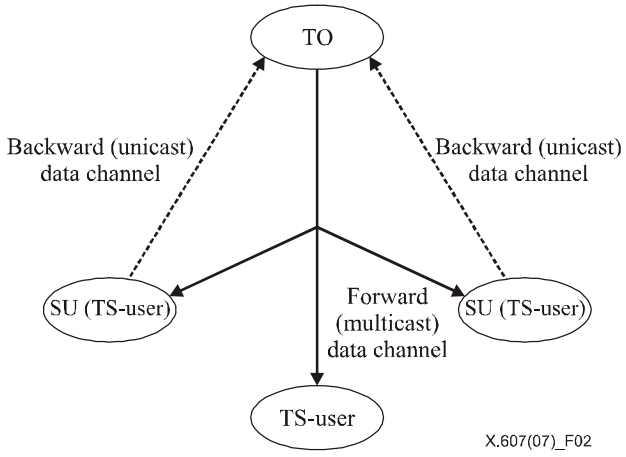

Figure 2 – Data transport in ECTP-3

To establish a duplex multicast connection, TO transmits a Connection Creation Request (CR) packet to the group. The CR packet contains the connection information including general characteristics of the connection. In particular, the CR packet must indicate that the connection type is the duplex multicast transport. Each TS-user who wants to participate in the connection will respond to the TO with a Connection Creation Confirm (CC) packet. The connection creation operation will be completed when a predetermined *CCT* timer expires.

During the connection creation phase, a logical control tree is configured between TO and TS-users, or between TS-users for providing the scalable reliability control. With the root of the TO, the control tree defines a parent-child relationship between any pair of two TS-users. The parent TS-user is called Local Owner (LO). Based on the control tree, the error recovery will be performed. To configure a control tree, each TS-user sends a Tree Join Request (TJ) message to a candidate parent node that has already been connected to the tree. The parent node will respond to the promising child TS-user with the Tree Join Confirm (TC) message. In this way, the control tree will gradually be expanded from the root toward the leaf nodes.

Some of the prospective TS-users may join the connection as late-joiners. The late-joining TS-user participates in the connection by sending a Late Join Request (JR) message to TO. In response to the JR message, TO sends a Late Join Confirm (JC) message to the TS-user. The late-joiner TS-user will also join the control tree by using the TJ and TC messages. For this purpose, the JC message of TO may include the information about the prospective parent LO node for the late-joiner. The late-joining TS-user may try to connect to the prospective LO node so as to configure the control tree.

After the connection is established, the data transmission phase starts. ECTP-3 protocol supports two types of data channels: the forward multicast channel from TO to the group and the backward unicast channel from the TS-user to TO. ECTP-3 provides the reliable data transport with error recovery, in which all the Data (DT) packets will be recovered by the parent on the tree.

In the forward multicast data transmission, TO can begin the multicast data transmission to the group by using the IP multicast address and group port number. The multicast data packets sent by TO will be sequentially segmented and transmitted by DT packets to the receiving TS-users. The TS-users will deliver the received DT packets to the upper-layer application in the order transmitted by TO.

For the forward multicast data channel of TO, the error control will be performed based on the local group defined by the ECTP control tree. If a child node detects a data loss, it sends a retransmission request to its parent via ACK packets. Each child generates an ACK packet every *ACK Generation Number* (*AGN*) data packets. That is, an ACK packet is generated for the *AGN* data packets of TO. An ACK message contains a 'bitmap' to indicate which data packets have been received or not. In response to an ACK packet, each parent LO may retransmit the RD packets to its children.

In the forward multicast data transport, the Heartbeat (HB) and Heartbeat ACK (HBACK) packets are used between a parent and children for the control tree maintenance. A parent transmits HB packets to the children every *HB Generation Time* (HGT). The HB contains which child must respond to this HB packet with the HBACK packet. The corresponding child will send a HBACK packet to the parent. The HB packet may also be used by the parent to calculate the local Round Trip Time (RTT) for the group. For this purpose, the HB and HBACK packets contain a timestamp.

For the backward unicast data transport, a certain TS-user in the connection may get a token from TO by sending a Token Get Request (TGR) message. The TO will then respond to the TS-user with the Token Get Confirm (TGC) message that contains a *Token ID*. Accordingly, the total number of tokens in the connection is controlled by TO. Token ID is used to identify the sender of the unicast DT packets at the TO side. The TS-user who has a token is called Sending TS-user (SU).

The SU can send unicast DT packets to TO. For the error recovery and congestion control, the HB and HBACK packets are exchanged between SU and TO. The SU sends an HB message to TO. The TO then responds with the HBACK packet that contains the acknowledgement information, as done in ACK packets in the forward multicast channel. It is noted that the HBACK is used for retransmission request in the backward channel.

After completing the unicast data transmission, the SU will return the token to the TO by sending a Token Return Request (TRR) message. TO will respond to the SU with a Token Return Confirm (TRC) message.

The connection management operations are taken in the connection: user leave, the connection pause and resumption, and connection termination. In the User Leave operation, a participating TS-user may leave the connection by sending a User Leave Request (LR) message to the parent. In a certain case, the parent may enforce a specific child node to leave the connection by sending the LR message, which is called the troublemaker ejection. The TO may temporarily pause and resume the connection. In the connection pause period, the TO will send Null Data (ND) packets to the group. After the TO has completed the data transport, it may terminate the duplex connection by sending a Connection Termination Request (CT) message to the group.

7 Design considerations

In this clause, some considerations for ECTP-3 are described.

7.1 Participants

All participants to a duplex multicast connection are TS-users and one of them functions as TC-Owner.

TC-Owner:

In a duplex multicast connection the TC-owner is responsible for connection management including connection creation/termination, late join, connection maintenance, and token management.

TS-user (Transport Service User):

A duplex multicast connection has one or more TS-users who can receive the multicast data from the TC-Owner. Some of the TS-users can send unicast data to the TC-Owner.

A TS-user can become TO, LO or LE, depending on its role.

TO (Top Owner):

A duplex multicast connection has a single TO, which corresponds to the TC-Owner. The TO is responsible for the overall operations required for connection management including connection creation and termination, control tree creation, late join, and connection maintenance. TO is also a single sender of the forward multicast data channel. Only the TO is allowed for sending the original multicast data to the other participants.

LO (Local Owner):

In the duplex multicast connection, an LO is a TS-user who is responsible for error recovery to the local group by retransmission of data. On the control tree hierarchy of ECTP-3, an LO is a parent node and has its children nodes. Note that an LO is also a TS-user. That is, an LO also receives multicast data from the TO. In ECTP-3, a TS-user may act as an LO in the connection, or some designated LOs may be used for the error recovery in the connection. It depends on the deployment of ECTP-3.

LE (Leaf Entity):

In the duplex multicast connection, an LE represents a leaf node on the control tree. Each LE is a TS-user that receives the multicast data from the TO.

A TS-user can become SU when it obtains a token from TC-Owner.

SU (Sending TS-user):

An SU is a TS-user who can send unicast data to TO. In the duplex multicast connection, a TS-user becomes an SU when it has a token and it can thus transmit unicast data to TO.

7.2 Control tree

A duplex multicast connection may configure a control tree for scalable reliability control as shown in Figure 3:

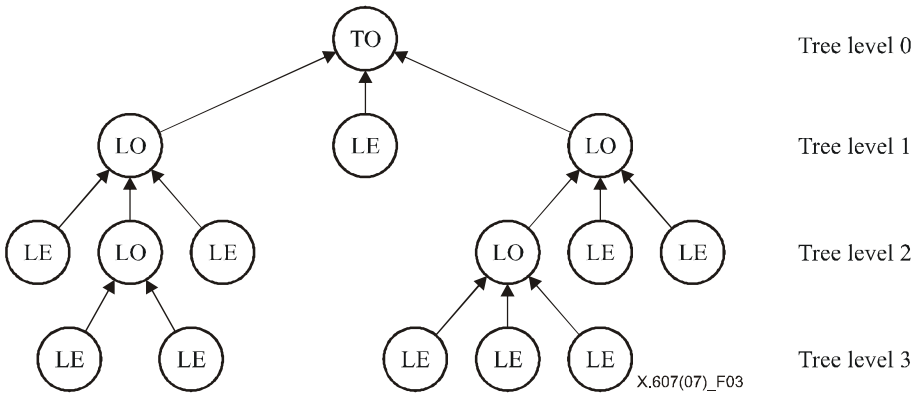

Figure 3 – Control tree in ECTP-3

In the ECTP-3 control tree, TO is on the top of the tree, which is in the Tree Level 0. An LO is a parent node on the tree and has one or more children. A TS-user, not designated as LO, is called a Leaf Entity (LE), which cannot have its children. Such a control tree will be configured in the connection creation phase.

Error recovery in ECTP-3 will be performed within each local group defined by the control tree. A child can request retransmission to its parent LO. In response to the request, the parent LO will retransmit the data packets to the children, if it has them in the buffer. An LO is also a TS-user, and it thus receives the multicast data from the TO. The control tree is applied only for forward multicast data channel. The control tree does not apply to the backward unicast data channel.

7.3 Data channels

In ECTP-3, the two types of data channels are used: forward and backward data channels.

7.3.1 Forward data channel

The forward data channel is used for TO to send multicast data to the other members. The forward multicast data channel can also be used for an LO to send Retransmission Data to its children users.

The forward data channel address consists of the group (multicast) IP address and the group port. TO sends multicast data via DT packets by using the forward data channel address. TO and LOs can also retransmit multicast data via RD packets by using the forward data channel address.

Figure 4 illustrates the use of the forward multicast data channels in ECTP-3.

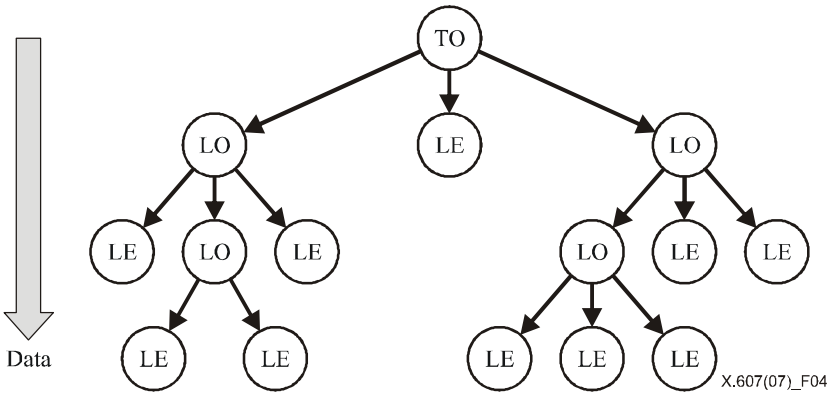

Figure 4 – Forward data channels and control tree in ECTP-3

7.3.2 Backward data channel

The backward data channel is used by a Sending TS-user (SU) to send unicast data to TO. The backward channel address consists of the IP address of TO and the 'group' port.

Figure 5 illustrates the use of the backward unicast data channels in ECTP-3.

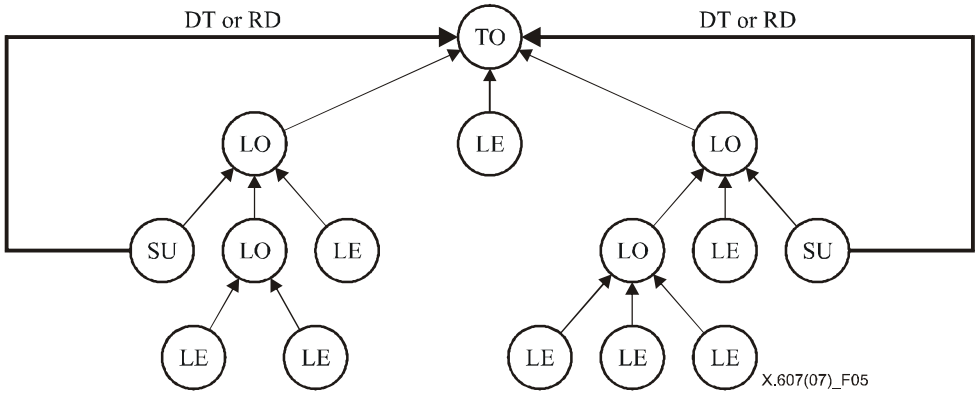

Figure 5 – Backward data channels in ECTP-3

Each SU must send unicast data via DT and RD packets to TO by using this backward data channel address as the destination address. On the other hand, TO must bind its backward data channel address to receive the unicast data from any SU in the connection.

7.4 Addressing

In ECTP-3, each packet uses the following types of IP addresses and port number for its source and destination address:

a) group IP address and local IP address;

b) group port and local port.

7.4.1 Group and local IP addresses

The group IP address is the IP multicast address (e.g., Class D address for IPv4), whereas a local IP address represents the unicast IP address for each of the ECTP participants: TO, LOs and LEs.

The group IP address is used as the destination address of the packets that need to be multicast by TO and LOs. For example, the CR and DT packets of TO will use the group IP address as the destination address of the associated IP packets. Each LO also uses the group IP address as the destination address of the RD and HB packets.

The local IP address of each participant is used as the source and destination IP address for the unicast packets, and also the source address for the multicast packets.

It is noted that the group IP address and the local IP address of TO must be announced to all the prospective participants via an out-of-band signalling such as Web announcement.

7.4.2 Group and local ports

The group port represents the port number that has been announced to all of the ECTP-3 participants before the connection. In ECTP-3, the group port will typically be used as the 'destination port' of the ECTP-3 multicast packets transmitted by TO or LOs, such as CR and DT. That is, each TS-user should bind to the group IP address and port so as to receive the relevant ECTP-3 multicast packets.

The group port number is also used by SU to send unicast data to TO. That is, TO will bind to the local port with its local IP address so as to receive the unicast data from any SU. In particular, the group port is also used as the destination port of the packet that requests a certain action, such as Late Join.

On the other hand, in the other cases that are not described above, the ECTP-3 packet will use the local port number as source and/or destination ports. For example, in response to the Late Join Request (JR) from a TS-user, the TO will respond with the Late Join Confirm (JC) message that use the local port of the TS-user as the destination port.

The detailed use of the local IP address and port is specified below for each of the ECTP-3 packets.

7.4.3 Addresses of data channels

In ECTP-3, all the data packets use the group port number as the destination port. Accordingly, before the connection creation, the following information must be announced to all of the ECTP-3 participants via an out-of-band signalling such as Web announcement.

a) group IP address and group port;

b) local IP address of TO.

Figure 6 describes the use of IP address and port for the forward and backward data channels. The forward multicast data packets use the group IP address and port number as the destination address of the data packets, whereas the backward data packets use the local IP address of TO and the group port number as the destination address.

The detailed use of the group and local addresses for the other packets will be specified later.

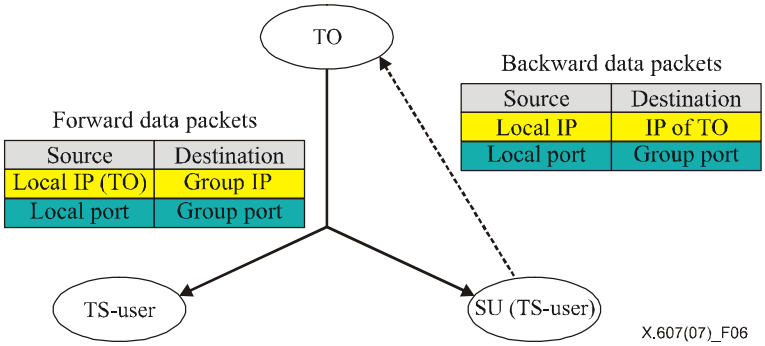

Figure 6 – Data channel addressing in ECTP-3

7.5 Tokens

In ECTP-3, a token represents the right for a TS-user to send a unicast data to TO. Before transmitting the data, each TS-user must get a token from the TO, as per the Token Control procedures of ECTP-3.

Each token is represented as a 1-byte non-negative integer in ECTP-3. Such a token number (or Token ID) will be assigned by TO when a TS-user requests a token in the connection. Token ID is ranged between 1 and 255. The Token ID of '0' is reserved for use of TO. At the TO side, the Token ID can be used to identify who is sending the unicast data.

8 Packets

An ECTP packet contains a 16-byte base header together with either extension elements or user data. It is noted that the data packets do not include any extension elements. The ECTP-3 packet format is illustrated in Figure 7:

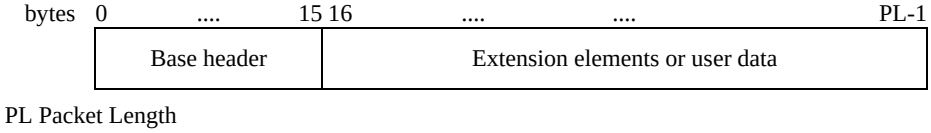

Figure 7 – ECTP-3 packet format

8.1 Base header

The 16-byte base header contains the information helpful to all the protocol operations, in particular for the data packets. Figure 8 shows the structure of the base header, when ECTP operates over IP.

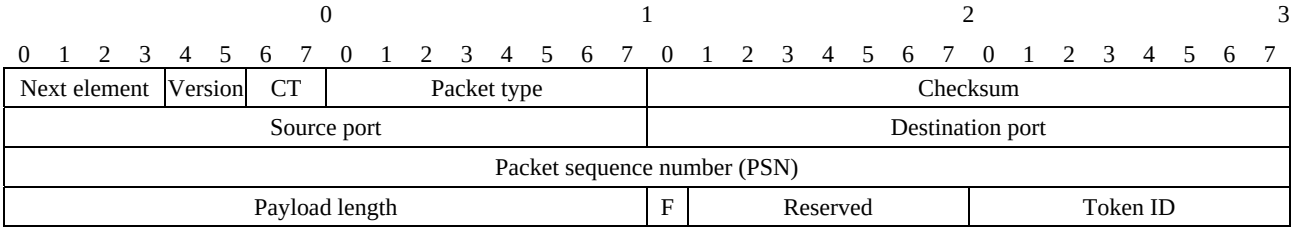

Figure 8 – Base header (ECTP over IP)

The base header contains the following information:

a) *Next element* (4 bits)

This specifies the type of the extension element immediately following the base header. The encoding values of the extension elements will be described later. The extension element value of '0000' means that the next part of this packet contains the user data, if any.

b) *Version* (2 bits)

This defines the version of the ECTP-3 protocol. Its current version is encoded as '00'.

c) *CT (Connection type)*: (2 bits)

This specifies the type of the ECTP connection. The encoding value for the connection type is as follows:

1) 01 – simplex multicast connection (for ECTP-1 and ECTP-2);
2) 10 – duplex multicast connection (for ECTP-3 and ECTP-4);
3) 11 – N-plex multicast connection (for ECTP-5 and ECTP-6).

The value '00' is reserved for future use. In this ECTP-3 specification, the *CT* must be set as '10'. It is noted that this definition is compatible with the specifications of ECTP-1 and ECTP-2.

d) *Packet type* (8 bits)

It indicates the type of this ECTP-3 packet. The encoding values of the ECTP-3 packets will be described later.

e) *Checksum* (16 bits)

This is used to check the validity of the ECTP-3 packet that includes the base header, extension header and/or user data. The ECTP-3 checksum is calculated by using the conventional one's complement arithmetic operation, as done in TCP and UDP.

f) *Source port* (16 bits) and *destination port* (16 bits)

These port numbers are used to identify the sending and receiving applications for the case of ECTP-3 over IP. When ECTP-3 operates over UDP, these fields are used to represent the connection identifier, as described later.

g) *PSN* (32 bits)

This value represents the sequence number of the data packet for the ECTP-3 DT or RD packets. For some control packets such as ND or HB packets, this value has a different semantic, which will be described later. For the other control packets, it is ignored. This sequence number is a 32-bit unsigned number that starts with the initial sequence number and increases by '1', and wraps back around to '1' after reaching '$2^{32} – 1$'.

h) *Payload length* (16 bits)

This value indicates the total length of the extension headers or user data in byte, following the base header.

i) *F* (1 bit)

It is a flag bit. The use of this flag depends on the packet types:

1) For the JC (Late Join Confirm), TJ (Tree Join Confirm), Token Get Confirm (TGC), Token Return Confirm (TRC) packets, the *F* = 1 indicates that each of the corresponding join request is accepted. *F* is set to 0, otherwise;
2) For the LR (User Leave Request) packet, *F* is set to '1' for the user-invoked leave, or set to '0' for the troublemaker ejection;
3) For the CT (Connection Termination Request) packet, *F* is set to '1' for an abnormal termination, or set to '0' for the normal termination, after all the data have been transmitted.

For the other packets, the detailed description is given in the protocol procedure clause. Otherwise, if any usage is not specified, this field will be ignored.

j) *Reserved* (7 bits)

This field is reserved for future use.

k) *Token ID* (8 bits)

The Token ID is valid only for data packets: DT and RD packets. This represents who is the source of the data packets. The Token ID value is ranged between 0 and 255. Each SU receives a Token ID from TO via the token get procedure and sets this field to be the number assigned by TO. The forward multicast data packets of TO will set this field to '0'.

On the other hand, when ECTP operates over UDP, the packet header does not need to specify the source and destination ports, which will be referred to from the UDP header. In this case, the 32-bit field for the source and destination ports will be filled with '*Connection ID*'. By default, it may be set to be the IPv4 group address.

The base header format for ECTP over UDP is as shown in Figure 9:

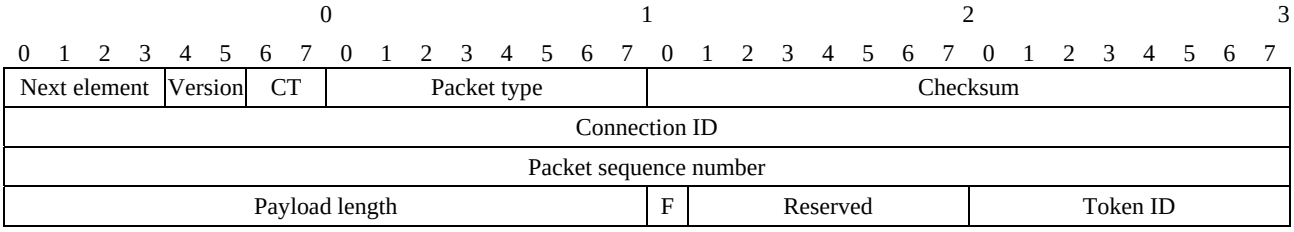

Figure 9 – Base header (ECTP over UDP)

The *Connection ID* is used to identify an ECTP connection by the ECTP host. It may also be used to verify the connection. In the connection setup phase, this information must first be informed by TO to the other participants via the CR or JC packets. All the other ECTP-3 packets must set this field to be the value announced by TO.

8.2 Extension elements

The ECTP packets used for control may contain one or more extension elements along with the base header. The based header and each extension element have the field of 'Next element' that points to the immediately succeeding extension element, if any.

The Next element field is encoded as shown in Table 1. It is noted that the '0000' means 'No element'. Accordingly, the last extension element of an ECTP packet must set its Next element field to '0000'.

Table 1 – Extension elements

Extension element	Encoding value in next element of the preceding element (4 bits)	Length of extension element (bytes)
No element	0000	0
Connection	0001	4
Acknowledgment	0010	Varied
Membership	0011	4
Timestamp	0100	12
Address	0110	8 or 20

It is noted that all the extension elements other than 'Address' element have already been defined in ECTP-1 and ECTP-2. Accordingly, the encoding values of those extension elements will be reused in ECTP-3. It is noted that the encoding value of '0101' is reserved for the QoS extension element, which is not used in ECTP-3, and may be defined for the QoS management in ECTP-4.

All the extension elements described in the table will be defined in this subclause by encompassing the requirements for the ECTP-3 protocol.

8.2.1 Connection element

The connection extension element contains overall information on the ECTP-3 transport connection. It is encoded as '0001' in the Next element field of the preceding element or based header. This extension element must be included in the CR, JC and TGR packets. The element structure is shown in Figure 10, which has the length of '4' bytes:

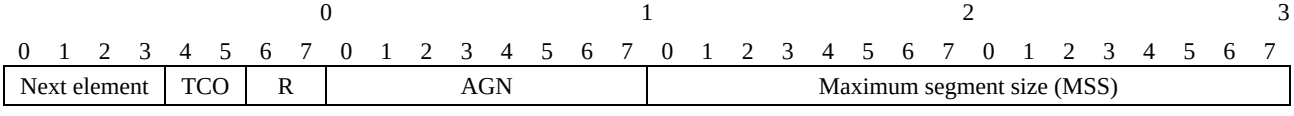

Figure 10 – Connection extension element

Each field is specified as follows:

a) *Next element* (4 bits)

This indicates the type of the next extension element, as indicated in Table 1.

b) *TCO (Tree Configuration Option)*: (2 bits)

This specifies the tree configuration option used in this ECTP-3 connection as follows:

1) 00 – Level 1 configuration:

No LO is used in the connection. Thus all TS-users must be connected to the TO as their parent on the control tree.

2) 01 – Level 2 configuration:

A control tree with the tree level of 2 is configured: TO ⇔ LO ⇔ LE. Each LE may be connected to an LO or directly to TO. The LEs are all connected to TO on the tree.

3) 10 – General configuration:

In this option, a control tree with more than two tree-levels may be configured.

4) 11 – Reserved for future use.

c) *R* (Reserved): (2 bits)

Reserved for future use.

d) *AGN (ACK Generation Number)*: (8 bits)

This is a positive integer ranged from 1 to 255 that represents the ACK Generation Number. The AGN is used to generate and transmit an ACK packet by a child user in the ECTP-3 protocol.

e) *MSS* (16 bits)

This specifies the maximum size of the user data segment that can be contained in an ECTP-3 DT packet in byte. An ECTP-3 DT packet can maximally contain '65535' bytes of user data. The default value is '1024'.

8.2.2 Acknowledgement element

This extension element provides information on the status of the packet reception at the child node, which will be referred to by the parent node for the error, flow and congestion control. This extension header is attached to the ACK packet. It is encoded as '0010' in the Next element field of the preceding element or based header.

This element consists of the fixed 4-bytes and the variable size of *ACK Bitmap*, as depicted in Figure 11.

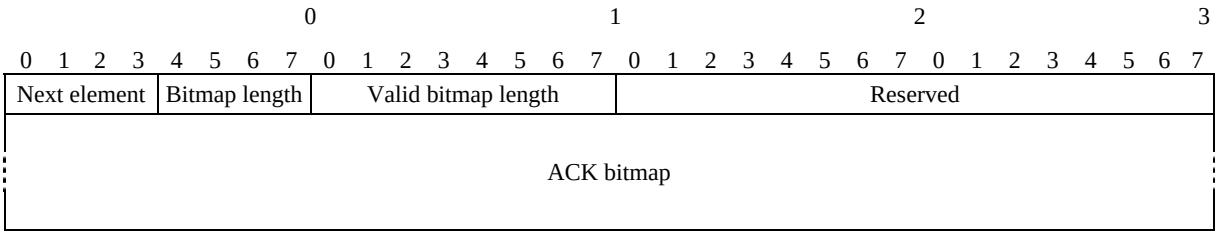

Figure 11 – Acknowledgement extension element

Each field is specified as follows:

a) *Next element* (4 bits)

This indicates the type of the next extension element, as indicated in Table 1.

b) *Bitmap length* (4 bits)

This specifies the total size of the variable ACK Bitmap in unit of word (4 bytes), which excludes the fixed 8 bytes. It is noted that the maximum value of Bitmap Length is '8' (words).

c) *Valid bitmap length* (8 bits)

This represents the length of the actually valid portion in 'bit' for the *ACK Bitmap*.

d) *Reserved* (16 bits)

This is reserved for future use. In particular, some information on QoS management may be specified in this field by ECTP-4.

e) *ACK bitmap* (variable)

This represents information by using '0' or '1' about which data packets have been received (1) or lost (0) at the receiver side. In the *ACK Bitmap*, the bitmap information starts with the *LSN* sequence number, which represents the sequence number of the lowest numbered DT packet that has not been received yet at the receiver side. The *LSN* will be specified in the PSN field of the base header. The *ACK Bitmap* contains the total bitmap information of the size of *Valid Bitmap Length*.

The total length of the *ACK Bitmap* part is represented by *Bitmap Length* in word. Accordingly, the padding of '0's may occur for the portion of *ACK Bitmap* other than the *Valid Bitmap Length* bits so as to ensure the alignment of Acknowledgement element into the unit of word.

As an example of the use of *ACK Bitmap*, let us assume that LSN = 100, *Valid Bitmap Length* = 8, and *ACK Bitmap* = 01110111. This *Bitmap* implies that the receiver has lost the data packets with the packet sequence number of 100 and 104, and it received all the data packets with the sequence number of smaller than 100. In this case, *Bitmap Length* is of '1' and all of the last 24 bits of ACK *Bitmap* will be padded with '0's. It is noted that the first bit of ACK *Bitmap* must always be set to '0'.

8.2.3 Membership element

This 4-byte extension element contains information on the tree membership, as illustrated in Figure 12. It is encoded as '0011' in the Next element field of the preceding element or based header.

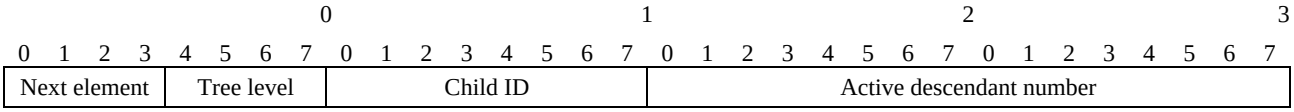

Figure 12 – Membership extension element

Each field is specified as follows:

a) *Next element* (4 bits)

b) *Tree level* (4 bits)

This specifies the tree level of the sender of this packet on the ECTP-3 control tree. TO will be in the tree level '0', and its children are in the tree level '1'. The *Tree level* value will be increased by '1' as the tree grows gradually.

c) *Child ID* (8 bits)

This is a non-negative integer that specifies the ID number of a child node. In the tree creation phase, this Child ID will be assigned to each of the children by the parent. Child ID is used to determine when to send an ACK packet to its parent, which will be described later. An ACK packet of a child node contains its Child ID, which can be used for the error recovery by the parent. The value of '0' is used to indicate that this field must be ignored.

d) *Active Descendant Number (ADN)*: (16 bits)

This represents the number of active descendants of the node along the tree. Each TS-user sets the *ADN* to '1', since it has no child. The parent LO will aggregate all of the *ADN* values for its children. In this fashion, TO can obtain the information on how many TS-users are participating in the connection.

8.2.4 Timestamp element

The Timestamp element is encoded as '0100' in the Next element field of the preceding element or based header. The ECTP-3 uses the 8-byte timestamp so as to calculate *Round Trip Time* (*RTT*).

The 12-byte timestamp extension element is formatted as shown in Figure 13:

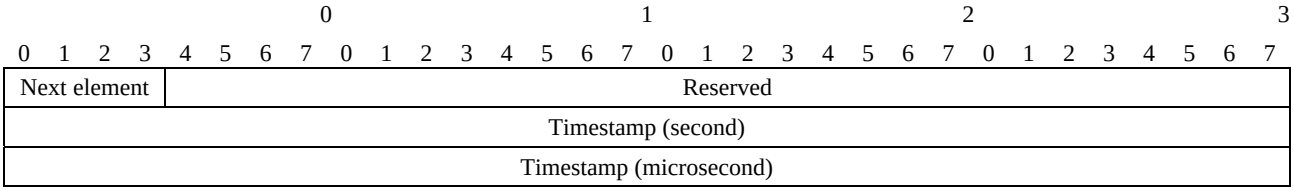

Figure 13 – Timestamp extension element

Each field is specified as follows:

a) *Next element* (4 bits);

b) *Reserved* (28 bits);

c) *Timestamp* (64 bits)

This contains the 8-byte timestamp value. The first 4 bytes represent the time value in second, and the second 4-byte does in microsecond, as done in the conventional *ping* program.

8.2.5 Address element

The Address extension element is encoded as '0110' in the Next Element field of the preceding element or based header. This element is attached to the packet when the protocol needs to specify the IPv4 or IPv6 address of the participant concerned. For example, the JC packet of TO may include this element so as to inform a late-joining user about the IP address of the LO that the late-joining user may connect.

The element length is 8 or 20 bytes, depending on the IP version. This element is formatted as shown in Figure 14.

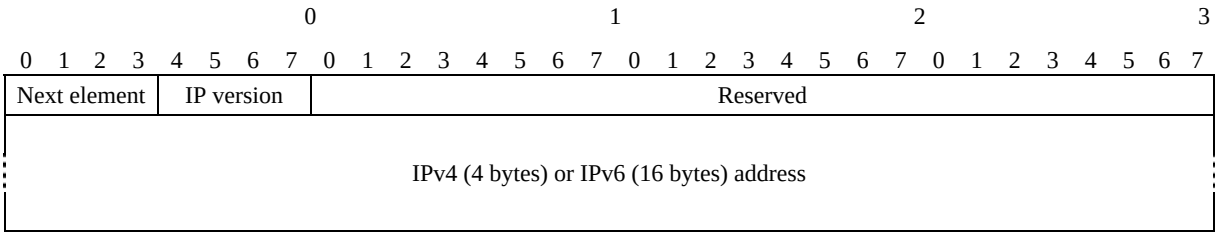

Figure 14 – Address extension element

Each field is specified as follows:

a) *Next element* (4 bits);

b) *IP version* (4 bits):

This field indicates the version of the IP address contained in this element. At present, the following values are available: IPv4 (0100) and IPv6 (0110).

c) *Reserved* (24 bits);

d) *IP address* (32 or 128 bits): This field indicates the IPv4 or IPv6 address of the participant concerned.

8.3 Packet format

ECTP-3 defines the total 18 packet types: 3 types of data packets and 15 types of control packets. The data packets are DT, ND and RD. Table 2 summarizes the packets used in ECTP-3. In the table, the shaded regions indicate the newly defined packet types in ECTP-3.

Table 2 – ECTP-3 packets

Full name	Acronym	Transport	From	To
Connection Creation Request	CR	Multicast	TO	TS-users
Connection Creation Confirm	CC	Unicast	TS-user	TO
Tree Join Request	TJ	Unicast	Child	Parent
Tree Join Confirm	TC	Unicast	Parent	Child
Data	DT	Multicast Unicast	TO SU	TS-user TO
Null Data	ND	Multicast	TO	TS-users
Retransmission Data	RD	Multicast Unicast	Parent SU	Children TO
Acknowledgement	ACK	Unicast	Child	Parent
Heartbeat	HB	Multicast Unicast	Parent SU	Children TO
Heartbeat Acknowledgement	HBACK	Unicast	Child/TO	Parent/SU
Late Join Request	JR	Unicast	TS-user	TO
Late Join Confirm	JC	Unicast	TO	TS-user
User Leave Request	LR	Unicast	Parent/Child	Child/Parent
Connection Termination Request	CT	Multicast	TO	TS-users
Token Get Request	TGR	Unicast	SU/TO	TO/SU
Token Get Confirm	TGC	Unicast	TO/SU	SU/TO
Token Return Request	TRR	Unicast	TS-user/TO	TO/TS-user
Token Return Confirm	TRC	Unicast	TO/TS-user	TS-user/TO

In the table, the parent node represents TO or LO, and the packets used for token control can be initiated by SU as well as TO. As also shown in the table, all of the ECTP-3 packets are exchanged between the participants in the request-and-confirm manner. For example, the CR request expects to receive the corresponding CC confirm message. The ACK messages will be used to confirm the DT and RD messages. On the other hand, LR and CT messages do not require any responding confirm messages.

On the other hand, the encoding value and packet structure are shown for each of the ECTP-3 packets in Table 3. The extension elements are attached to the base header in the order specified in the table.

Table 3 – Format of ECTP-3 packets

Packet type	Encoding value	Extension elements or user data (packet structure)	Length (bytes)	Operational protocol stage
CR	0000 0001	Connection	20	Connection Creation
CC	0000 0010		16	
TJ	0000 0011		16	Control Tree Creation
TC	0000 0100	Membership + *Address*	20+	
DT	0000 0101	User Data	16+	Data Transport (Forward and Backward)
ND	0000 0110		16	
RD	0000 0111	User Data	16+	
ACK	0000 1000	Membership + Acknowledgement	20+	
HB	0000 1001	Membership + *Timestamp*	20+	Control Tree Maintenance
HBACK	0000 1110	Membership + Acknowledgement + *Timestamp*	20+	
JR	0000 1010		16	Late Join
JC	0000 1011	Connection + *Address*	20+	
LR	0000 1100		16	User Leave
CT	0000 1101		16	Termination

Table 3 – Format of ECTP-3 packets

Packet type	Encoding value	Extension elements or user data (packet structure)	Length (bytes)	Operational protocol stage
TGR	0001 0001		16	Token Get
TGC	0001 0010		16	
TRR	0001 0011		16	Token Return
TRC	0001 0100		16	

In the table, the shaded regions indicate the newly defined packets in ECTP-3: HBACK, TGR, TGC, TRR and TRC. It is noted that the italic parts (of the extension elements) indicate that the use of the corresponding extension element is optional, rather than mandatory, in the implementation.

In the packet length of the table, '+' means that the packet size may get larger by adding the specified optional element or user data. It is noted in the table that the ND packet is used only for the forward multicast data transport. On the other hand, the following encoding values are reserved for future use: '0000 0000', '0000 1111' and '0001 0000'.

8.3.1 Connection creation request (CR)

The CR packet is used by TO so as to create a duplex multicast connection. TO sends the CR packet to the group with the following source and destination addresses:

- Source IP: IP address of TO.
- Source port: Local port number of TO.
- Destination IP: Multicast IP address of the group.
- Destination port: Group port number.

The length of the CR packet is 20 bytes (16-byte base header + 4-byte Connection element). The CR packet is formatted as shown in Figure 15:

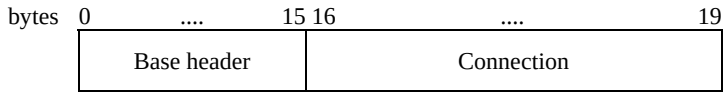

Figure 15 – CR packet

The 16-byte base header of CR packet must be encoded as follows:

a) *Next element*: '0001' (Connection element).

b) *Version*: '00' (current version of ECTP-3).

c) *CT*: '10' (duplex multicast connection).

d) *Packet type*: '0000 0001' (CR).

e) *Checksum*: To be calculated.

f) *Source port*: Local port number of TO (or Connection ID).

g) *Destination port*: Group port number (or Connection ID).

h) *PSN*: The initial *PSN* which is assigned by TO and used for the forward data channel.

i) *Payload length*: 4.

j) *F*: '0' (to be ignored).

k) *Token ID*: '0' (Token ID of TO is set to '0').

The 4-byte Connection element must be encoded as follows:

a) *Next element*: '0000' or '0100' (if Timestamp element is optionally added).

b) *TCO*: As configured by TO.

c) *RO*: As configured by TO.

d) *AGN*: As configured by TO.

e) *MSS*: As configured by TO (the default value is '1024').

8.3.2 Connection creation confirm (CC)

The CC packet is used by TS-user in response to the CR packet of TO. A TS-user sends the CC packet to TO over the following source and destination addresses:

- Source IP: IP address of TS-user.
- Source port: Local port number of TS-user.
- Destination IP: IP address of TO.
- Destination port: Group port number.

The CC packet contains the 16-byte base header only. The base header of CC packet must be encoded as follows:

a) *Next element*: '0000' or '0100' (if Timestamp element is optionally added).

b) *CT*: '10'.

c) *Packet type*: '0000 0010' (CC).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of TS-user (or Connection ID).

f) *Destination port*: Group port number (or Connection ID).

All the fields other than specified above will be set to '0' and ignored at the receiver side.

8.3.3 Tree join request (TJ)

The TJ packet is used by TS-user to the prospective parent node in order to join the control tree. The prospective parent node may be TO itself, or indicated by TO (via the JC packet). A TS-user sends the TJ packet to the parent (TO or an LO) over the following source and destination addresses:

- Source IP: IP address of TS-user.
- Source port: Local port number of TS-user.
- Destination IP: IP address of the parent node (TO or LO).
- Destination port: Group port number.

The TJ packet contains the 16-byte base header only. The base header of TJ packet must be encoded as follows:

a) *Next element*: '0000'.

b) *CT*: '10'.

c) *Packet type*: '0000 0011' (TJ).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of TS-user (or Connection ID).

f) *Destination port*: Group port number (or Connection ID).

All the fields other than specified above will be set to '0' and ignored at the receiver side.

8.3.4 Tree join confirm (TC)

The TC packet is used by the parent TO or LO node, in response to the TJ packet. The parent node sends the TC packet to the TS-user over the following source and destination addresses:

- Source IP: IP address of the parent node (TO or LO).
- Source port: Group port number.
- Destination IP: IP address of the TS-user.
- Destination port: Local port of the TS-user.

The TC packet contains the 16-byte base header. In the case of acceptance of the corresponding TJ request, the TC packet will also contain the 4-byte Membership element, and possibly with the Address element (optional).

The TC packet is formatted as shown in Figure 16:

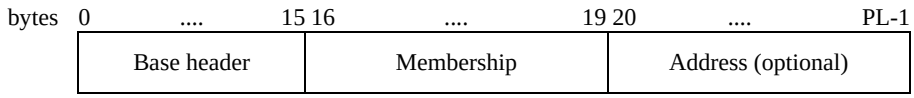

Figure 16 – TC packet

The 16-byte base header of TC packet must be encoded as follows:

a) *Next element*: '0011' (Membership element).

b) *CT*: '10'.

c) *Packet type*: '0000 0100' (TC).

d) *Checksum*: To be calculated.

e) *Source port*: Group port (or Connection ID).

f) *Destination port*: Local port of TS-user (or Connection ID).

g) *Payload length*: 4 or more (if the Address element is added).

h) *F*: '1' if the TJ request is accepted, '0' otherwise.

All the fields other than specified above will be set to '0' and ignored at the receiver side.

The 4-byte membership element must be encoded as follows:

a) *Next element*: '0000' or '0110' (if Address element is optionally added).

b) *Tree level*: Represents the current tree level of the parent node.

c) *Child ID*: Is assigned by the parent node.

d) *ADN*: Set to '0' and ignored.

The address element may optionally be added to the membership element. This address element will be added only if the TJ request is not accepted. In this case, the address information represents the IP address of another prospective parent node that the failed child node may retry to connect to.

In this case, the address element must be encoded as follows:

a) *Next element*: '0000'.

b) *IP version*: '0100' (IPv4) or '0110' (IPv6).

c) *IP address*: IP address of the prospective parent LO.

8.3.5 Data (DT)

The DT packet is used by TO to transmit the forward multicast data to the group members, or by SU to send the backward unicast data to TO.

The forward multicast DT packets are sent to the TS-users over the following source and destination addresses:

- Source IP: IP address of TO.
- Source port: Local port number of TO.
- Destination IP: Multicast IP address of the group.
- Destination port: Group port number.

The backward unicast DT packets are sent to the TO over the following source and destination addresses:

- Source IP: IP address of SU.
- Source port: Local port number of SU.
- Destination IP: IP address of TO.
- Destination port: Group port number.

The DT packet contains the 16-byte base header and the variable-length user data.

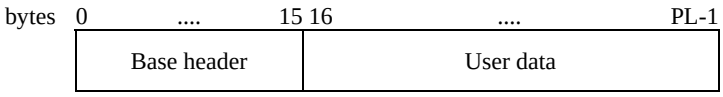

Figure 17 – DT packet

The 16-byte base header of DT packet must be encoded as follows:

a) *Next element*: '0000'.

b) CT: '10' (duplex multicast connection).

c) *Packet type*: '0000 0101' (DT).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of TO (or Connection ID).

f) *Destination port*: Group port number (or Connection ID).

g) *PSN*: The *PSN* of this DT packet, which will be increased by 1, starting from Initial PSN.

h) *Payload length*: Indicates the length (in byte) of the user data contained in this packet.

i) *F*: '0' (to be ignored).

j) *Token ID*: Token ID of the sender of this data packet ('0' for TO, or a positive number of SU).

8.3.6 Null data (ND)

The ND packet is used in the forward data channel by TO so as to indicate the ECTP connection is still alive. The receiving TS-users do not have to respond to this packet.

The TO sends the ND packet to the group over the following source and destination addresses:

- Source IP: IP address of TO.
- Source port: Local port number of TO.
- Destination IP: Multicast IP address of the group.
- Destination port: Group port number.

The ND packet contains the 16-byte base header only. The 16-byte base header of ND packet must be encoded as follows:

a) *CT*: '10' (duplex multicast connection).

b) *Packet type*: '0000 0110' (ND).

c) *Checksum*: To be calculated.

d) *Source port*: Local port number of TO (or Connection ID).

e) *Destination port*: Group port number (or Connection ID).

f) *PSN*: The *PSN* of the last DT packet that has been sent by TO.

All the fields other than specified above will be set to '0' and ignored at the receiver side.

8.3.7 Retransmission data (RD)

The RD packet is used in the forward data channel by the parent node (TO or LO) on the control tree to retransmit the DT packet for error recovery. It is also used in the backward data channel by the SU for error recovery.

The packet format is the same as that of the DT packet.

8.3.8 Acknowledgement (ACK)

The ACK packet is used in the forward data channel by a child node to the parent node on the control tree in order to acknowledge the DT packets received from TO. An ACK packet is generated by the ACK generation rule, which will be described later. It is not used in the backward data channel.

For the forward data channel, an ACK packet is transmitted by a child node to its parent over the following source and destination addresses:

- Source IP: IP address of TS-user.
- Source port: Local port number of TS-user.
- Destination IP: IP address of the parent node (TO or LO).
- Destination port: Group port number.

The ACK packet contains the 16-byte base header, 4-byte Membership element, and a variable length of Acknowledgement element, as shown in Figure 18.

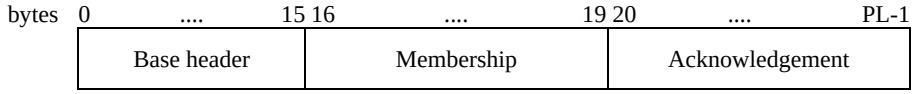

Figure 18 – ACK packet

The base header of ACK packet must be encoded as follows:

a) *Next element*: '0011' (Membership element).

b) *CT*: '10'.

c) *Packet type*: '0000 1000' (ACK).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of TS-user or group port number TO (or Connection ID).

f) *Destination port*: Group port number or local port number of SU (or Connection ID).

g) *PSN*: *LSN*, the *PSN* of the lowest numbered DT packet that has not been received yet.

h) *Payload length*: Length (in byte) of the extension elements attached to the base header.

i) *F*: '0' (to be ignored).

j) *Token ID*: Token ID of the corresponding sender ('0' for TO, or a positive number of SU).

The 4-byte membership element must be encoded as follows:

a) *Next element*: '0010' (Acknowledgement element).

b) *Tree level*: Represents the current tree level of the child node ('0' for TO).

c) *Child ID*: Child ID (an integer) that is assigned to the child; this is ignored for the backward channel.

d) *ADN*: The number of active descendants of the child for the forward data channel; this is ignored for the backward channel.

The acknowledgement element must be encoded as follows:

a) *Next element*: '0000'.

b) *Bitmap length*: Represents the total length of the ACK Bitmap in word (in 4-byte).

c) *Valid bitmap length*: The actually valid length of the ACK Bitmap in bit.

d) *ACK bitmap*: Represents the bitmap information about which DT packets are lost, starting from the LSN.

8.3.9 Heartbeat (HB)

The HB packet is used by the parent TO or LO node for tree maintenance. This can also be used to calculate the local *RTT* optionally between the parent and the child. This is also used in the backward data channel between SU and TO.

The parent node sends the periodic HB packet to its children over the following source and destination addresses:

- Source IP: IP address of the parent node (TO or LO).
- Source port: Local port number of the parent node.
- Destination IP: Multicast IP address of the group.
- Destination port: Group port number.

The HB packet contains the 16-byte base header and the 4-byte Membership element, and possibly with the Timestamp element (optional), as shown in Figure 19.

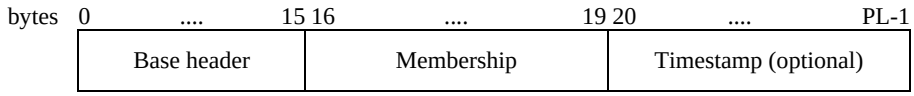

Figure 19 – HB packet

The base header of the HB packet is formatted as follows:

a) *Next element*: '0011' (Membership element).

b) *CT*: '10'.

c) *Packet type*: '0000 1001' (HB).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of the parent (or Connection ID).

f) *Destination port*: Group port number (or Connection ID).

g) *PSN*: *PSN* of the lowest numbered DT packet that the parent node contains in the buffer.

h) *Payload length*: Length (in byte) of the extension elements attached to the base header.

All the fields other than specified above will be set to '0' and ignored at the receiver side.

The 4-byte membership element must be encoded as follows:

a) *Next element*: '0000' or '0100' (if the Timestamp element is optionally added).

b) *Tree level*: Represents the current tree level of the parent node ('0' for TO).

c) *Child ID*: Child ID that should respond to this HB packet with the HBACK packet.

d) *ADN*: Is set to '0' and ignored.

The optional 12-byte Timestamp element must be encoded as follows:

a) *Next element*: '0000'.

b) *Timestamp*: 8-byte time value measured at the parent, which is used for calculation of local RTT.

8.3.10 Heartbeat acknowledgement (HBACK)

The HBACK packet is sent by the child node, in response to the HB packet of the parent. When a child receives the HB packet from the parent, if the Child ID of the HB packet is equal to its Child ID, then the child should respond with the HBACK packet. This packet is used to indicate that it is still alive. It can also be used to calculate the local RTT by the parent.

The HBACK packet is also used in the backward data channel between SU and TO.

The child node sends the HBACK packet to its parent over the following source and destination addresses:

- Source IP: IP address of the child.
- Source port: Local port number of the child.
- Destination IP: IP address of the parent.
- Destination port: Local port number of the parent.

The HBACK packet contains the 16-byte base header and the 4-byte Membership element, the Acknowledgement element, and possibly with the Timestamp element (optional), as shown in Figure 20.

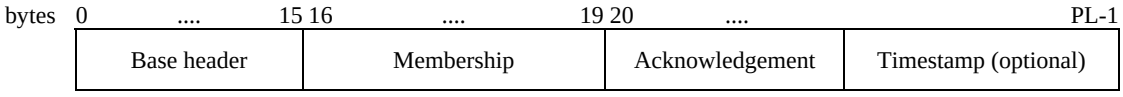

Figure 20 – HBACK packet

The base header of the HBACK packet is formatted as follows:

a) *Next element*: '0011' (Membership element).

b) *CT*: '10'.

c) *Packet type*: '0000 1110' (HBACK).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of the child (or Connection ID).

f) *Destination port*: Local port number of the parent (or Connection ID).

g) *Payload length*: Length (in byte) of the extension elements attached to the base header.

All the fields other than specified above will be set to '0' and ignored at the receiver side.

The 4-byte membership element must be encoded as follows:

a) *Next element*: '0000' or '0100' (if the Timestamp element is optionally added).

b) *Tree level*: Represents the current tree level of the child.

c) *Child ID*: Child ID of the child.

d) *ADN*: The number of active descendants of the child.

The acknowledgement element must be encoded as done in the ACK packet.

The optional 12-byte Timestamp element must be encoded as follows:

a) *Next element*: '0000'.

b) *Timestamp*: 8-byte time value that has been contained in the HB packet of the parent.

8.3.11 Late join request (JR)

The JR packet is used by a new joining TS-user in order to join the ECTP connection. The new joiner TS-user sends the JR packet to the TO over the following source and destination addresses:

- Source IP: IP address of TS-user.
- Source port: Local port number of TS-user.
- Destination IP: IP address of the TO.
- Destination port: Group port number.

The JR packet contains the 16-byte base header only. The base header of JR packet must be encoded as follows:

a) *Next element*: '0000'.

b) *CT*: '10'.

c) *Packet type*: '0000 1010' (JR).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of TS-user (or Connection ID).

f) *Destination port*: Group port number (or Connection ID).

All the fields other than specified above will be set to '0' and ignored at the receiver side.

8.3.12 Late join confirm (JC)

The JC packet is used by TO, in response to the JR packet. The TO sends the JC packet to the new joiner over the following source and destination addresses:

- Source IP: IP address of the TO.
- Source port: Local port number of TO.
- Destination IP: IP address of the TS-user.
- Destination port: Local port of the TS-user.

The JC packet contains the 16-byte base header. In the case of acceptance of the corresponding LJ request, the JC packet will also contain the 4-byte Connection element, and possibly with the Address element (optional).

The JC packet is formatted as shown in Figure 21:

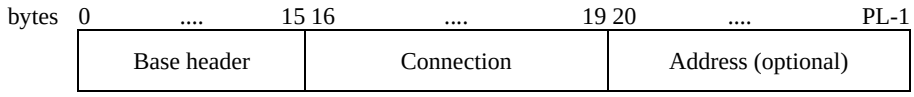

Figure 21 – JC packet

The 16-byte base header of JC packet must be encoded as follows:

a) *Next element*: '0001' (Connection element).

b) *CT*: '10'.

c) *Packet type*: '0000 1011' (JC).

d) *Checksum*: To be calculated.

e) *Source port*: Local port of TO (or Connection ID).

f) *Destination port*: Local port of TS-user (or Connection ID).

g) *Payload length*: 4 or more (if the Address element is added).

h) *F*: '1' if the LJR request is accepted, '0' otherwise.

All the fields other than specified above will be set to '0' and ignored at the receiver side.

The 4-byte Connection element must be encoded as follows:

a) *Next element*: '0000' or '0110' (if Address element is optionally added).

b) *TCO*: As configured by TO.

c) *AGN*: As configured by TO.

d) *MSS*: As configured by TO (the default value is '1024').

The address element may optionally be added to the Connection element. This address element will be added only if the TO wants to inform the new joiner TS-user about its promising parent LO node. In this case, the new joiner may try to connect the parent node that is indicated by the address element in the tree configuration.

In this case, the address element must be encoded as follows:

a) *Next element*: '0000'.

b) *IP version*: '0100' (IPv4) or '0110' (IPv6).

c) *IP address*: IP address of the prospective parent LO.

8.3.13 User leave request (LR)

The LR packet is used by the child to indicate that it will leave the connection, or by the parent node to eject a trouble-making child. The LR packet does not need to require the corresponding confirm packet. This packet is sent by the child or parent over the following source and destination addresses:

- Source IP: IP address of the child (user leave) or parent (troublemaker ejection).
- Source port: Local port number of the child or parent.
- Destination IP: IP address of the parent or child.
- Destination port: Group port number or local port number of the child.

The LR packet contains the 16-byte base header only. The base header is formatted as follows:

a) *Next element*: '0000'.

b) *CT*: '10'.

c) *Packet type*: '0000 1100' (LR).

d) *Checksum*: To be calculated.

e) *Source port*: Local port of the child or parent (or Connection ID).

f) *Destination port*: Group port or local port of child (or Connection ID).

g) *F*: '1' for the user-invoked leave, or '0' for the troublemaker ejection.

All the fields other than specified above will be set to '0' and ignored at the receiver side.

8.3.14 Connection termination request (CT)

The CT packet is used by the TO to terminate the connection. CT packet does not need to require the corresponding confirm packet. This packet is sent by TO over the following source and destination addresses:

- Source IP: IP address of the TO.
- Source port: Local port number of TO.
- Destination IP: Multicast IP address of the group.
- Destination port: Group port number.

The CT packet contains the 16-byte base header only. The base header is formatted as follows:

a) *Next element*: '0000'.

b) *CT*: '10'.

c) *Packet type*: '0000 1101' (CT).

d) *Checksum*: To be calculated.

e) *Source port*: Local port of the TO (or Connection ID).

f) *Destination port*: Group port (or Connection ID).

g) *F*: '1' for an abnormal termination, or '0' for the normal termination (after completing the multicast data transmission).

All the fields other than specified above will be set to '0' and ignored at the receiver side.

8.3.15 Token get request (TGR)

The TGR packet is used by a TS-user to get a token for the backward unicast data transport, which is called 'User-initiated Token Get'. In this case, the TS-user can request a token to TO by sending a TGR packet. The TS-user that has a token becomes an SU.

The TGR packet is sent over the following source and destination addresses:

- Source IP: IP address of TS-user.
- Source port: Local port number of TS-user.
- Destination IP: IP address of the TO.
- Destination port: Group port number of TO.

The TGR packet contains the 16-byte base header only, which must be encoded as follows:

a) *Next element*: '0000'.

b) *CT*: '10'.

c) *Packet type*: '0001 0001' (TGR).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of TS-user (or Connection ID).

f) *Destination port*: Group port number of TO (or Connection ID).

g) *PSN*: The initial *PSN* of the backward data channel.

h) *Payload length*: '0'.

i) *Token ID*: Is set to '0' and ignored.

8.3.16 Token get confirm (TGC)

The TGC packet is sent by TO to give a token to the associated TS-user. The TGC packet is sent over the following source and destination addresses:

- Source IP: IP address of TO.
- Source port: Group port number of TO.
- Destination IP: IP address of the TS-user.
- Destination port: Local port of TS-user.

The TGC packet contains the 16-byte base header only, which must be encoded as follows:

a) *Next element*: '0000'.

b) *CT*: '10'.

c) *Packet type*: '0001 0010' (TGC).

d) *Checksum*: To be calculated.

e) *Source port*: Group port number of TO (or Connection ID).

f) *Destination port*: Local port number of TS-user (or Connection ID).

g) *Payload length*: '0'.

h) *F*: '1' (for acceptance) or '0' (for rejection).

i) *Token ID*: Token ID allocated by TO (Token Get).

8.3.17 Token return request (TRR)

The TRR packet is used by a TS-user to return a token to TO, which is called 'User-initiated Token Return'. In this case, the TS-user sends a TRR packet.

The TRR packet is sent over the following source and destination addresses:

- Source IP: IP address of TS-user.
- Source port: Local port of TS-user.
- Destination IP: IP address of the TO.
- Destination port: Group port of TO.

The TRR packet contains the 16-byte base header only, which must be encoded as follows:

a) *Next element*: '0000'.

b) *CT*: '10'.

c) *Packet type*: '0001 0011' (TRR).

d) *Checksum*: To be calculated.

e) *Source port*: Local port number of TS-user (or Connection ID).

f) *Destination port*: Group port number of TO (or Connection ID).

g) *Token ID*: Token ID of SU.

All the fields other than specified above will be set to '0' and ignored at the receiver side.

8.3.18 Token return confirm (TRC)

The TRC packet is sent by TO to confirm the associated TRR request. The TRC packet is sent over the following source and destination addresses:

- Source IP: IP address of TO.
- Source port: Group port of TO.
- Destination IP: IP address of the SU.
- Destination port: Local port of SU.

The TRC packet contains the 16-byte base header only, which must be encoded as follows:

a) *Next element*: '0000'.

b) *CT*: '10'.

c) *Packet type*: '0001 0100' (TRC).

d) *Checksum*: To be calculated.

e) *Source port*: Group port number of TO (or Connection ID).

f) *Destination port*: Local port number of SU (or Connection ID).

g) *Token ID*: Token ID of SU.

All the fields other than specified above will be set to '0' and ignored at the receiver side.

9 Procedures

This clause describes the protocol procedures of ECTP-3. Before a duplex multicast connection is created, the following address information should be announced to the prospective participants (TS-user).

a) multicast IP address of the group;

b) group port number;

c) IP address of TO.

This information may be announced to the prospective participants via an out-of-band signalling mechanism such as Web announcement. Accordingly, the prospective TS-user should be able to bind the group IP address and port so as to receive the CR packet from the TO. A prospective late-joiner should also send a JR packet to the TO.

9.1 Connection creation

A duplex multicast connection will begin when TO sends the first ECTP packet, CR, to the group over the multicast group IP address and port. An ECTP-3 control tree is also configured in the connection creation phase.

The overall operations for connection creation and control tree creation are shown in Figure 22.

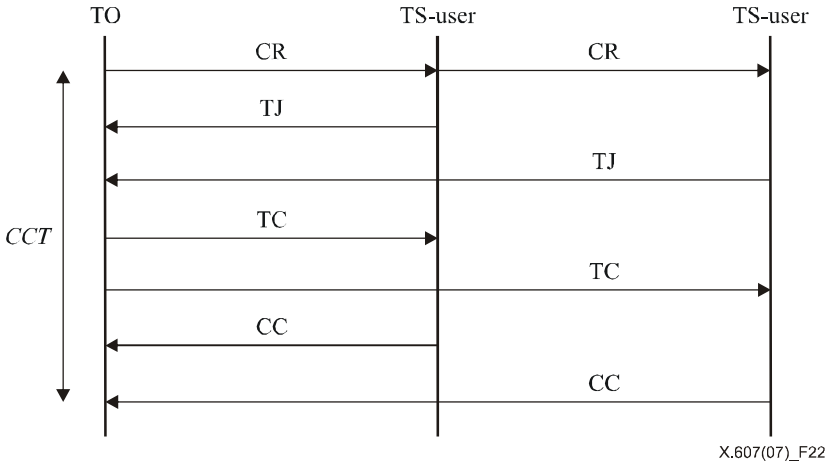

Figure 22 – Connection Creation procedures

9.1.1 Initiation of connection creation

TO begins the connection creation operation by sending the CR packet to the group. The CR packet contains the generic information on the connection element such as *TCO* (*Tree Configuration Option*), *RO* (*Reliability Option*), and *MSS* (*Maximum Segment Size*).

After sending the CR packet, TO starts the *CCT* (*Connection Creation Timer*). Only the CC packets will be allowed during the *CCT* timer.

Each TS-user should join the control tree before responding with a CC packet. In the connection creation phase, each user can join only the TO.

9.1.2 Control tree creation

To join the control tree, each TS-user sends a TJ packet to TO. TO then responds to the TS-user with the TC packet. The TC packet should indicate whether the tree join request is accepted or not by using the *F* flag of the base header.

The TC packet should also contain the *Child ID* and *Tree Level* in the membership element. The TC packet may optionally contain the address element to represent a promising parent LO for the TS-user. It needs to ensure that the parent LO has already been on the tree.

9.2 Late join

Some of the prospective participants may join the duplex multicast connection as a late joiner. In particular, any TS-user is allowed to join the connection as a late joiner, after the *CCT* timer expires.

The overall operations for a late joiner are shown in Figure 23.

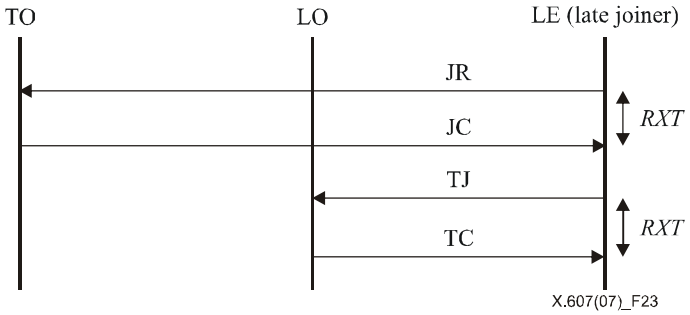

Figure 23 – Late Joining procedures

A promising TS user can join the connection via a normal join (by sending a CC packet in response to the CR packet) or a late-join (by sending a LJ packet to the TO). For this purpose, the promising TS-user may try to perform the late-joining operation after waiting for the CR packet from the TO for a pre-configured time, *CWT* (*CR Waiting Time*), which may be configured by each promising TS-user locally.

The late joining TS-user sends a JR packet to TO. In response to the JR packet, the TO sends a JC packet to the late joiner, which should indicate that the request is accepted or not by using the *F* flag of the base header.

The JC packet should contain the connection element. It may also contain the address element so as to recommend the promising parent LO for the TS-user (late-joiner). If no address element is indicated, the late-joiner may try to join the TO in the control tree configuration. If the JC packet does not arrive within the *RXT* (*Retransmission Timeout*), the late joiner may try to send the JR packet again.

When the TS-user receives the JC packet from the TO, it will join the control tree, as per the 'control tree creation' operations described in the previous subclause.

9.3 Forward data transport

In the ECTP-3 forward data channel, the TO sends multicast DT packets to the group. When a data packet loss is detected by the receiving user, the retransmission for error recovery will be performed within a local group that is defined by the control tree.

ECTP-3 provides the reliable data transport with the error recovery scheme, where all the DT packets will be recovered by the parent on the tree. Each child requests the retransmission via ACK packet, and the parent sends the corresponding RD packet over the multicast address.

9.3.1 Multicast data transmission

After the connection creation, the TO can send multicast DT packets to the group members.

TO will generate DT packets by the segmentation procedure. To do this, TO splits a multicast data stream of application into multiple DT packets. Each DT packet has its own sequence number. When TO has no data to transmit, TO may transmit the periodic ND packets in the connection pause period.

Each receiver delivers all the data packets received to the application in the order sent by TO. Each receiver reassembles the received packets. Corrupted and lost packets are detected by using a checksum and sequence number. A corrupted packet is also considered as a loss. The lost DT packets are recovered in the error control function.

ECTP-3 uses the flow control based on a fixed-size *window*. The *window size* represents the number of unacknowledged data packets in the sending buffer. TO can maximally transmit the *window size* of data packets at the configured data transmission rate. In ECTP-3, the transmission rate of multicast data is controlled by the rate-based congestion control mechanisms.

A new DT packet is sequentially numbered by TO. The sequence number of the DT packet starts from *Initial PSN* and increases by '1'. The sequence number is used to detect lost data packets by receivers. The *Initial PSN* is randomly generated other than '0'. The sequence number of '0' is reserved. The packet sequence number is increased for each new DT packet. Modulo 2^{32} arithmetic is used and the sequence number wraps back around to '1' after reaching "$2^{32} – 1$". The Initial PSN number will be informed to the group members by way of CR or JC packet.

9.3.2 Control tree maintenance

The ECTP-3 control tree is maintained using the HB and HBACK packets. Each parent LO advertises periodic HB packets using *Heartbeat Generation Time* (*HGT*), after it becomes an on-tree node. The default *HGT* is 3 seconds. The HB packet contains the information (*Child ID* of the membership element) about which child should respond to this HB packet. The corresponding child should respond with the HBACK packet.

Figure 24 shows the operation for tree maintenance using HB and HBACK packets.

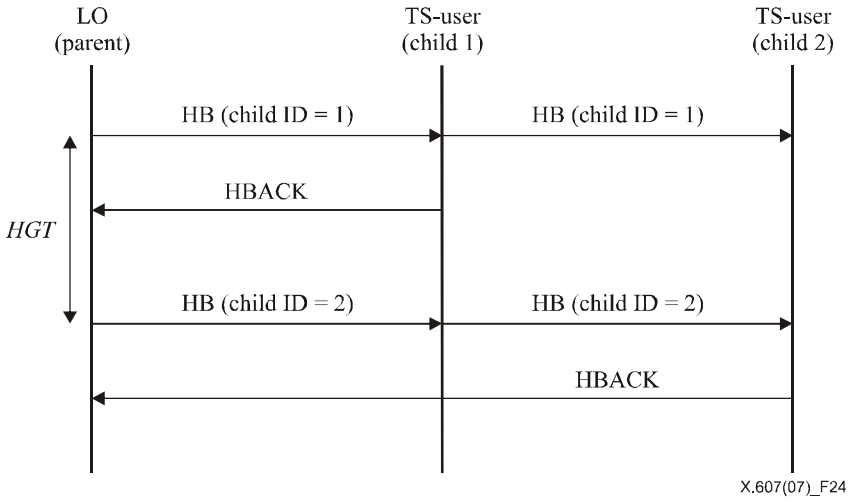

Figure 24 – Tree maintenance using HB and HBACK packets

In ECTP-3, the exchange of HB and HBACK packets between a parent and children is done with the following two purposes:

a) Check whether the tree node is still alive:

   A child detects the failure of its parent, if it cannot receive any packets such as HB and RD packets from the parent during the time interval of *FDN* (*Failure Detection Number*) × *Heartbeat Generation Time* (*HGT*). Then the child begins to find an alternate parent. The default *FDN* is 3.

   A parent detects the failure of a child, if it cannot hear any HBACK packets from the child for the *FDN* consecutive HB packets. If a child is detected as a failure, the parent sends an LR packet (for troublemaker ejection) to the failed child, and clears the child out of its children list.

b) Calculation of local RTT:

   On the other hand, the HB and HBACK can also be used to calculate the Round Trip Time (*RTT*) for a local group. A parent LO sends a HB packet containing a timestamp element to its children every *HGT* interval. The corresponding child will also contain the timestamp element as it is in the HBACK packet. Receiving an HBACK packet from a child, the parent LO calculates the RTT by subtracting *Timestamp* from the current time. The RTT is recorded into the children list. The parent determines the Local RTT by the maxim RTT value for its children.

It is noted that these HB and HBACK packets are exchanged 'periodically' between a parent and children, which will be implemented using the *HGT* timer. Accordingly, even if a HB or HBACK packet is lost, the subsequent HB or HBACK packets will be used for the control tree maintenance. As described above, if a child or parent cannot receive any HB or HBACK packets from the corresponding parent or child during the *FDN* × *HGT* time interval, then it will realize that the correspondent node failed in the control tree.

9.4 Token control

In ECTP-3, a token represents the right to send a data to the TO in the backward unicast data channel. Each TS-user who wants to transmit a data to TO must get a token from the TO. The TS-user will be an SU after getting a token from TO. In this way, TO can control the maximum number of tokens simultaneously active for the connection.

An SU returns the token after completing the unicast data transmission to TO.

9.4.1 Token get

The TS-user can get a token from TO. In this Token Get operation, the TS-user first requests a token to TO. Figure 25 shows the operations for Token Get.

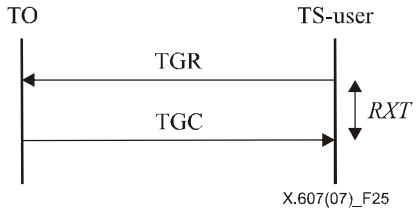

Figure 25 – Token Get procedure

To get a token in the Token Get operation, a TS-user sends a TGR message to TO, and then waits for the corresponding TGC message. In response to the TGR packet, the TO should send a TGC message to the TS-user. The TGC message should indicate whether the request is accepted or not by using the *F* flag of the base header. In case of acceptance, the message will also contain a valid *Token ID* and *Initial PSN* in the base header. If the responding TGC message has not arrived until the *RTO* timer expires, the TS-user may send the TGR message again.

9.4.2 Token return

When completing data transmission, the SU may return the token to TO. The SU can return its token to TO. In this Token Return operation, the SU sends the TRR packet to TO.

Figure 26 shows the operations for Token Return.

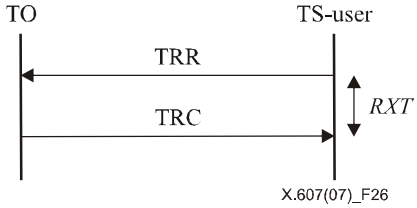

Figure 26 – Token Return procedures

In the Token Return case, the SU sends a TRR message to TO. The TO then responds with the TRC message. If the responding TRC message has not arrived until the *RTO* timer expires, the TRR message may be sent again.

9.5 Backward data transport

After getting a token from TO, an SU can transmit unicast DT packets to TO. The procedures for the backward unicast data transport are as shown in Figure 27.

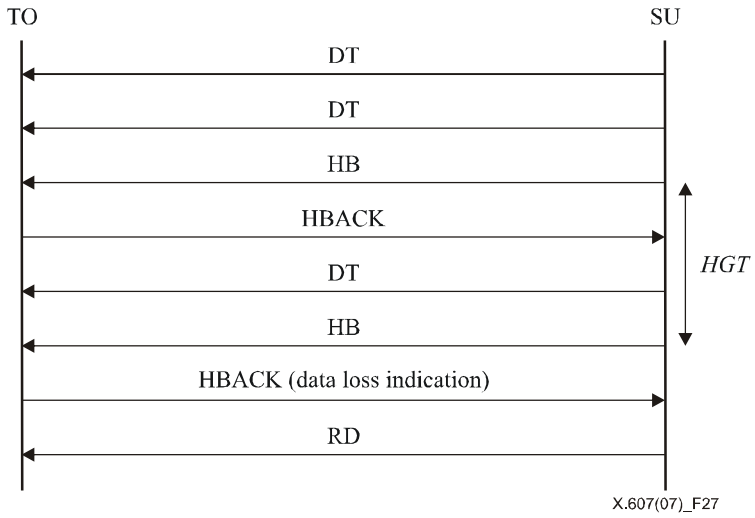

Figure 27 – Backward unicast data transport procedures

In the backward unicast data channel, the initial sequence number (*ISN*) of SU is indicated in the *PSN* field of the header of the TGR (in the Token Get case) or TGC (in the Token Give case). The *ISN* is randomly generated other than '0', as done in the forward data channel.

The DT packets transmitted by SU must indicate the Token ID that is allocated by TO.

9.6 Reliability control

9.6.1 Reliability control for forward data channel

9.6.1.1 Error detection

The checksum field of the base header is used for detection of packet corruption, and the *PSN* field is for detection of a packet loss. When a data packet is received, each receiver examines the checksum. If the checksum field is invalid, the packet is regarded as a corruption and shall be discarded. A corruption is treated as a loss. The loss can be detected as a gap of two consecutive sequence numbers for DT packets. The loss information is recorded into the ACK bitmap, which is attached to the subsequent ACK packets.

ACK packets are used for the retransmission requests. When a receiver detects a gap in the sequence numbers of received packets, it sets to zero the bit of the ACK bitmap that corresponds to the lost DT packet. The ACK bitmap is included into the acknowledgement element, which is attached to the subsequent ACK packet and delivered to the parent by the ACK generation mechanisms.

For a local group, a parent and its children maintain the *Lowest Sequence Number* (*LSN*) variables to determine the status of DT packets. For a child, the *LSN* represents the sequence number of the lowest numbered DT packet that the child has not received. For a parent, the *LSN* represents the sequence number of the lowest numbered DT packet that has not been acknowledged by its children.

To request the retransmissions of lost data, each child makes an acknowledgement element containing the *LSN*, *Valid Bitmap Length* and *ACK Bitmap*. The *ACK Bitmap* specifies a success or a failure of a packet delivery for each DT packet; '1' for success and '0' for failure. Suppose *Bitmap* = 01101111, *LSN* = 15. Then the DT packets with the sequence number 15 and 18 are lost.

9.6.1.2 Retransmission request by ACK generation

Figure 28 shows the error control operations in the forward multicast data channel.

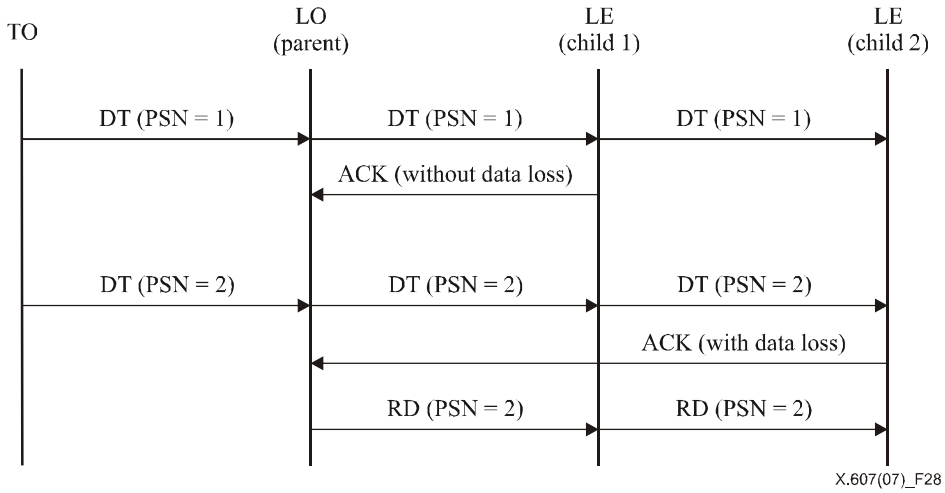

Figure 28 – Error recovery based on control tree for forward data channel

Each child generates an ACK packet by *ACK Generation Number* (*AGN*). Each child sends an ACK packet to its parent every *AGN* number of packets. To do this, a child receives a *Child ID* from its parent in tree configuration, which is contained in the tree membership element of the TC packet.

Each child sends an ACK packet to its parent, if the *PSN* number of a DT packet modulo *AGN* equals *Child ID* modulo *AGN*, i.e., if:

$$PSN \% AGN = Child\ ID \% AGN$$

Suppose *AGN* = 8 and *Child ID* = 2. The child generates an ACK packet for the DT packets whose sequence numbers are 2, 10, 18, 26, etc. This ACK generation rule is applied when the corresponding DT packet is received or detected as a loss by the child.

9.6.1.3 Retransmissions and ACK aggregation by LO

Each parent uses ACK packets to gather status information for the error recovery. Each time a parent receives an ACK packet from any of its children, it records and updates the status information on which packets have been successfully received by its children.

Each DT packet is defined as a 'stable' packet if all of the children have received it. The stable DT packets are released out of the buffer memory of the parent. When a parent receives an ACK packet from one of its children, if one or more packet losses are indicated, the parent transmits the corresponding RD packets to all of its children over its multicast control address.

After a parent retransmits an RD packet, it will ignore any subsequent retransmission requests for the same packet during the *RXT* period.

An ACK packet contains information on the *number of active descendants* (*ADN*). The parent aggregates the ADN variables for all of its children, and sends the aggregated information to its parent (when it sends an ACK to the parent).

9.6.2 Reliability control for backward data channel

The reliability control for the unicast data channel will be done between SU and TO. In the data transmission phase, the SU sends a HB packet to the TO every *HGT* time interval. The TO should respond with the HBACK packet to the SU. The HBACK packet may indicate the retransmission request by containing the 'acknowledgement' element. In this case, the SU sends the corresponding RD packet.

It is noted that the ECTP-3 uses the HBACK packet as the retransmission request from TO to SU, rather than using the ACK packet that is used as the retransmission request in the forward multicast data channel. For the purpose of retransmission request, a HBACK packet contains the 'acknowledgement' element, as described in the 'packet format'

clause. Based on the HBACK and its acknowledgement element received from the TO, the SU can retransmit the lost data packets. It is also noted that the SU can configure its HGT timer used to trigger those HB and HBACK packets.

This design is made from the viewpoint that the control tree is not needed between TO and SU, and thus the acknowledgement element for retransmission request could be piggybacked in the periodic HBACK packets flowing from TO and SU.

The HB and HBACK packets will also be used to calculate the RTT between SU and TO at the SU side.

9.7 Connection management

9.7.1 User leave

Figure 29 illustrates the operations for user-initiated leave and troublemaker ejection.

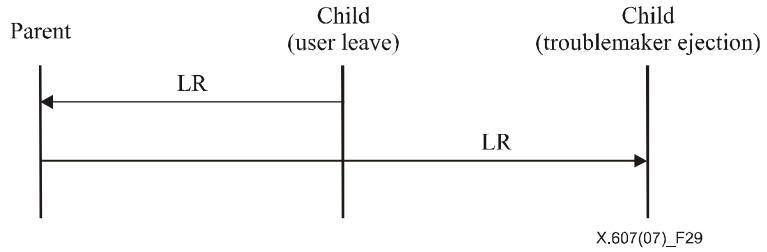

Figure 29 – User leave and troublemaker ejection procedures

In the User Leave case, the child node will send a LR message to its parent (LO or TO). In the Troublemaker Ejection case, the parent will request the concerned child to leave the connection. In both cases, the LR message does not require the corresponding confirm message. It is noted that the User Leave operation is performed between a parent and a child over the control tree, rather than between TO and TS-user. The troublemaker ejection may be applied to the child that has not been responding during a certain time interval in the HB and HBACK operation for tree maintenance. It is not recommended to apply the troublemaker ejection to an LO node that has one or more children.

9.7.2 Connection pause

In ECTP-3, the TO may pause the connection temporarily for a certain reason. For example, when it has no user data to transmit, the TO may pause the connection. The connection may also resume. During the connection pause period, the TO may send ND packets.

Figure 30 shows the operations for connection pause/resumption. The ND packets do not require any confirm messages.

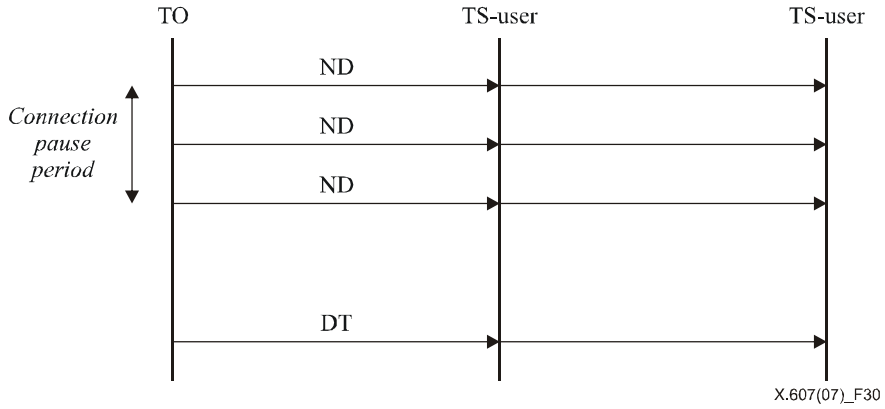

Figure 30 – Connection pause procedures

9.7.3 Connection termination

The TO may terminate the connection when it has completed the data transmission. The TO performs the connection termination by sending a CT message to the group.

Annex A

Timers and parameters

(This annex does not form an integral part of this Recommendation | International Standard)

This annex summarizes the timers and variables used for ECTP-3 for information.

A.1 Timers

a) *CCT* (Connection Creation Time)

TO triggers the *CCT* timer when sending the CCR packet to the group. The TO will process only the CCC packets that arrive from the prospective TUs before the *CCT* timer.

A specific value of CCT will be configured by the TO.

b) *CWT* (CR Waiting Time)

A promising TS user can join the connection via a normal join (by sending a CC packet in response to the CR packet) or a late-join (by sending a LJ packet to the TO). For this purpose, the promising TS-user may try to perform the late-joining operation after waiting for the CR packet from the TO for a pre-configured time, *CWT* (*CR Waiting Time*), which may be configured by each promising TS-user.

A specific value of CWT may be configured by a TS-user locally.

c) *LJT* (Late Join Time)

When a promising late-joining TS-user starts an ECTP connection, if it cannot receive any CCR message during the LJT time, it will try to join the ECTP connection as a late-joiner by sending an LJR message to the TO.

A specific value of LJT will be configured by the TS-user.

d) *HGT* (Heartbeat Generation Time)

Each parent node transmits the HB packet to its children every *HGT* second. Each child will respond with the HBACK packet if the Child ID of the HB packet is equal to its Child ID. The SU also sends the HB packets every *HGT* interval. The TO will respond with the HBACK packet.

The choice of *HGT* depends on the parent node and SU.

e) *RXT* (Retransmission Time)

In ECTP, the *RXT* timer is used by the packet initiator to wait for the corresponding confirm packet. For example, a late joiner TS-user sends an LJR packet to TO and waits for the LJC packet until the *RXT* timer expires. When the timer expires and the confirm packet has not arrived until then, it may send the request packet again. The *RXT* timer can also be used by a parent node to back off the retransmission request from the children for the RD packets that have already been retransmitted.

A specific value of *RXT* depends on the implementation.

A.2 Parameters

a) *ADN* (Active Descendants Number)

This represents the number of the active descendants on the tree. Each LO calculates the ADN value and delivers it via the ACK packet to the parent. In this way, the TO can be informed about the total number of active participants in the connection.

b) *AGN* (ACK Generation Number)

The AGN value will be informed to the TUs via the Connection Information element. This AGN value is referred to by each child node to realize when it should generate its ACK packet toward its parent.

A specific value of *AGN* may depend on the implementation.

c) *FDN* (Failure Detection Number)

The *FDN* number is used for a tree node to detect whether its parent or child node is still alive. In the tree maintenance, the parent may eject a child if the child has not been responding during the FDN consecutive times of HB packets.

d) *PSN* (Packet Sequence Number)

Each DT packet has its own *PSN* value in the header. This is used for the receiving TS-user to check the packet loss event and to rearrange the DT packet in the order transmitted.

ISN (Initial Sequence Number) is the initial PSN of the data sender, whereas the LSN (Lowest Sequence Number) represents the lowest sequence number of the DT packets contained in the buffer.

Annex B

State transition diagrams

(This annex does not form an integral part of this Recommendation | International Standard)

This annex gives a sketch of the state transition diagrams of ECTP-3 nodes, TO and TS-user, so as to facilitate an implementation of the ECTP-3 protocol.

Figure B.1 shows the state transition diagram of ECTP-3 TO, which is described based on the 'ECTP-3 procedures' in the main body of this Recommendation | International Standard.

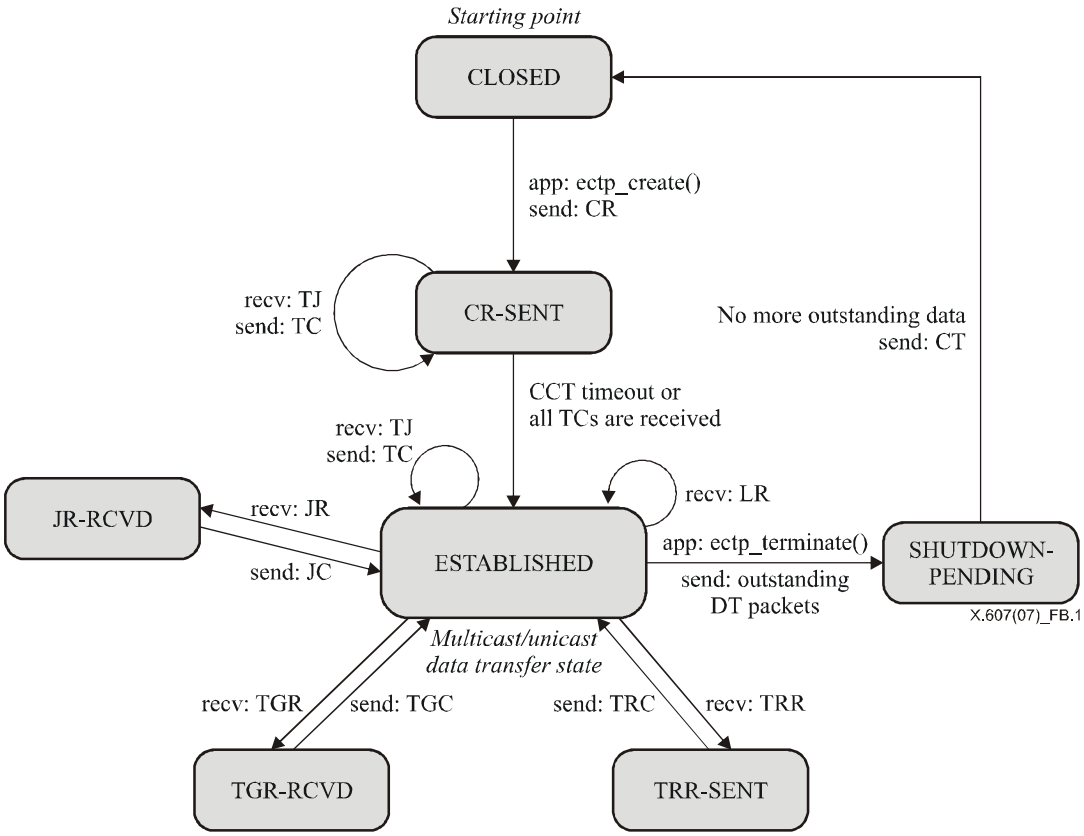

Figure B.1 – State transition diagram for ECTP-3 TO

Figure B.2 shows the state transition diagram of ECTP-3 TS-user, which is described based on the 'ECTP-3 procedures' in the main body of this Recommendation | International Standard.

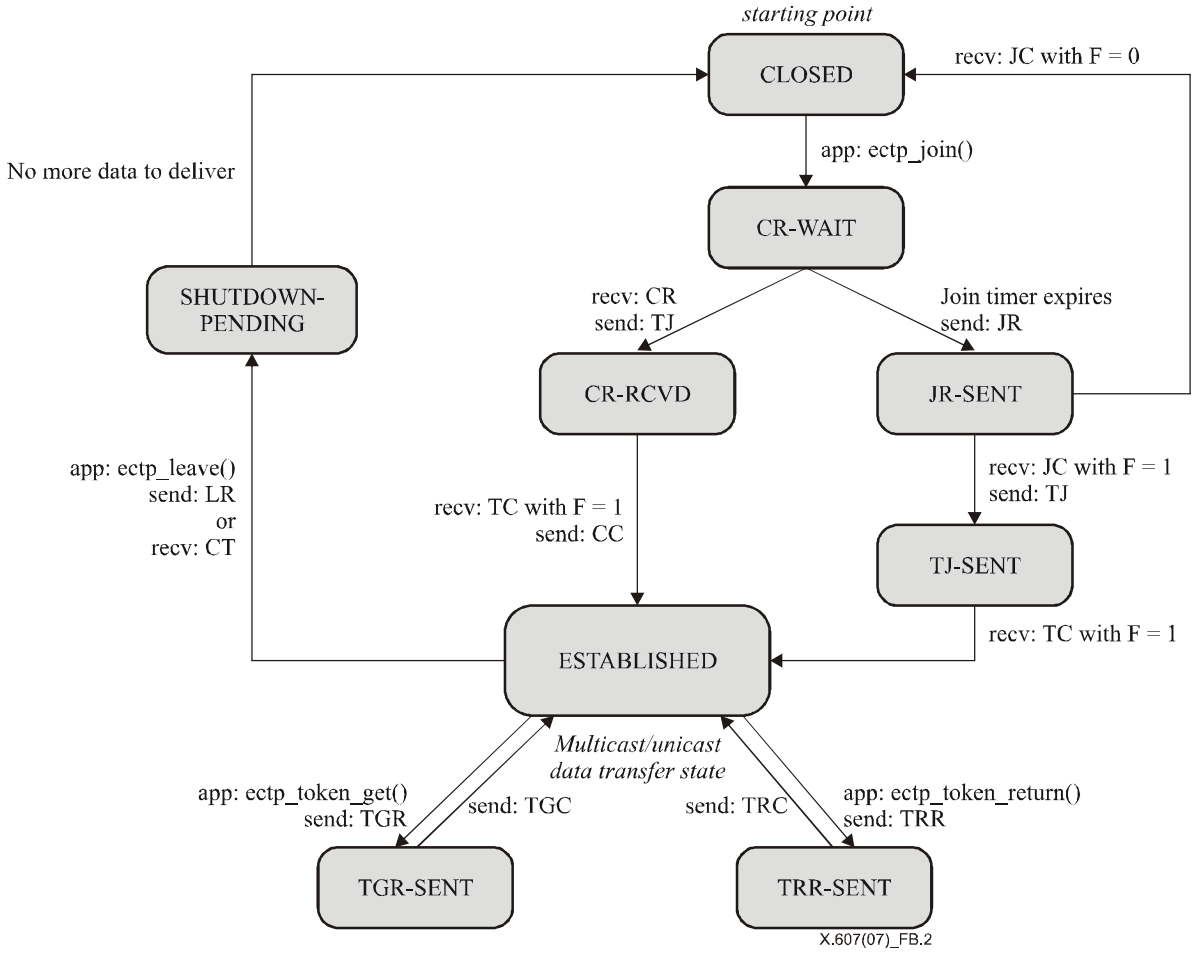

Figure B.2 – State transition diagram for ECTP-3 TS-user

Annex C

Application programming interfaces

(This annex does not form an integral part of this Recommendation | International Standard)

This annex gives the ECTP socket API functions for information that might be used by ECTP-3 applications. The ECTP-3 API functions include:

int msocket(int domain, int role);

- Parameters

  int domain: specifies a communication domain (AF_INET or AF_INET6)

  int role: specifies the role of the participants in the duplex multicast connection (TO or TS-user)
- Description

  msocket creates a socket descriptor for ECTP-3 communication.
- Return

  On success, a new socket descriptor is returned. On error, –1 is returned.

int mcreate(struct sockaddr_storage *local_addr,
struct sockaddr_storage *group_addr, int time);

- Parameters

  struct sockaddr_storage *local_addr:
  points to a sockaddr_storage structure containing the local address

  struct sockaddr_storage *group_addr:
  points to a sockaddr_storage structure containing the multicast group address

  int time: specifies the connection creation time (CCT)
- Description

  mcreate is used to create an ECTP-3 connection by TO.
- Return

  On success, 0 is returned. On error, –1 is returned.

int mjoin(struct sockaddr_storage *local_addr,
struct sockaddr_storage *group_addr, int time);

- Parameters

  struct sockaddr_storage *local_addr:
  points to a sockaddr_storage structure containing the TO's address

  struct sockaddr_storage *group_addr:
  points to a sockaddr_storage structure containing the multicast group address

  int time: specifies the CWT time
- Description

  mjoin is used by TS-user to request a group join.
- Return

  On success, zero is returned. On error, –1 is returned.

int mterminate(int esd);

- Parameters

  int esd: socket descriptor of ECTP-3
- Description

  mterminate is used by TO to terminate the connection.
- Return

  On success, zero is returned. On error, –1 is returned.

int mleave(int esd);

- Parameters

  int esd: socket descriptor of ECTP-3
- Description

  mleave is used by TS-user to leave the connection.
- Return

  On success, zero is returned. On error, –1 is returned.

int msendm(int esd, const void *msg, size_t len);

- Parameters

  int esd: socket descriptor of ECTP-3

  const void *msg: points to a buffer containing the multicast data messages to be sent

  size_t len: specifies the size of the message in bytes
- Description

  msendm is used by TO to transmit a multicast data message.
- Return

  It returns the size of data message transmitted in byte, or –1 if an error occurred.

int msendmsg(int esd, const void *msg, size_t len, int flags,
const struct sockaddr_storage *to, socklen_t tolen);

- Parameters

  int esd: socket descriptor of ECTP-3

  const void *msg: points to a buffer containing the unicast data message to be sent

  size_t len: specifies the size of the messages in bytes

  int flags: represents the flag used in the sendmsg()

  const struct sockaddr_storage *to:

  points to a sockaddr_storage structure containing the destination address

  socklen_t tolen: specifies the length of the sockaddr_storage structure
- Description

  msendmsg is used by SU to transmit a unicast message to TO.
- Return

  It returns the size of messages in byte, or –1 if an error occurred.

int mrecvmsg(int esd, void *msg, size_t len, int flags,
const struct sockaddr_storage *from, socklen_t *fromlen);

- Parameters

  int esd: socket descriptor of ECTP-3

  void *msg: points to the buffer where the message should be stored

  size_t len: specifies the message length in bytes pointed to by the msg argument

  int flags: represents the flags used in the recvmsg()

  const struct sockaddr_storage *from:

  A null pointer, or points to a sockaddr_storage structure associated

  socklen_t *fromlen: specifies the length of the sockaddr_storage structure pointed to
- Description

  mrecvmsg is used by TO or TS-user to receive data message.
- Return

  It returns the number of bytes received, or –1 if an error occurred.

int mtoken_get(int esd);

- Parameters
  int esd: socket descriptor of ECTP-3
- Description
  mtoken_get gets a token by SU from TO.
- Return
  On success, non-negative token id is returned. On error, –1 is returned.

int mtoken_return(int esd, int token_id);

- Parameters
  int esd: socket descriptor of ECTP-3
  int token_id: specifies the Token ID assigned by TO
- Description
  mtoken_return returns a token to TO.
- Return
  On success, zero is returned. On error, –1 is returned.

int msetsockopt(int esd, int level, int optname, void *optval,
socklen_t optionlen);

- Parameters
  int esd: socket descriptor of ECTP-3
  int level: specifies the protocol level associated with the option
  int optname: specifies the name of the option to be set
  void *optval: points to a buffer containing the option value to be set
  socklen_t optionlen: specifies the size of the optval in byte
- Description
  msetsockopt is used by TO to set options on a socket.
  The option of ECTP-3 is as follows.
  level: IPPROTO_ECTP
  optname:
  ECTP_OPT_TCO (for tree configuration)
  ECTP_OPT_AGN (for AGN)
  ECTP_OPT_MSS (for MSS)
- Return
  On success, zero is returned. On error, –1 is returned.

int mgetsockopt(int esd, int level, int optname, void *optval,
socklen_t *optionlen);

- Parameters
  int esd: socket descriptor of ECTP-3
  int level: specifies the protocol level associated with the option
  int optname: specifies the name of the option to get
  void *optval: points to the buffer in which the option value is to be stored
  socklen_t *optionlen: specifies the size of the optval in byte
- Description
  mgetsockopt is used to options (status) for a socket.
  The detailed use is similar to the msetsockopt()
- Return
  On success, zero is returned. On error, –1 is returned.

int mjoin_confirm(int esd, int accept)

- Parameter

  int esd: socket descriptor of ECTP-3

  int accept: specifies the indication of acceptance or not
- Description;

  mjoin_confirm is used by TO to respond to the late join request of TS-user.
- Return

  On success, zero is returned. On error, –1 is returned.

int mtoken_confirm(int esd, int accept);

- Parameters

  int esd: socket descriptor of ECTP-3

  int accept: specifies the indication of acceptance or not
- Description

  mtoken_confirm is used by TO to reply to a token get request.
- Return

  On success, zero is returned. On error, –1 is returned.

Figure C.1 shows an example sequence of the ECTP-3 API functions that is called by TO.

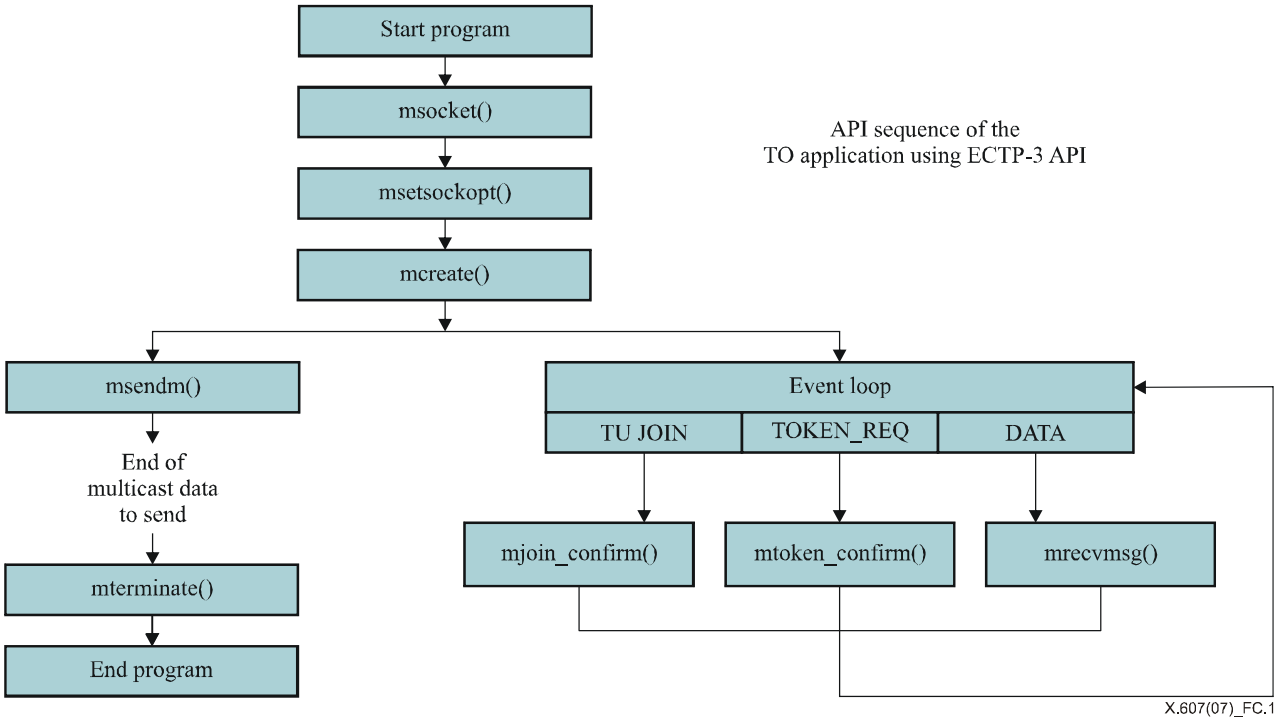

Figure C.1 – Example sequence of ECTP-3 APT functions called by TO

Figure C.2 shows an example sequence of the ECTP-3 API functions that is called by TS-user.

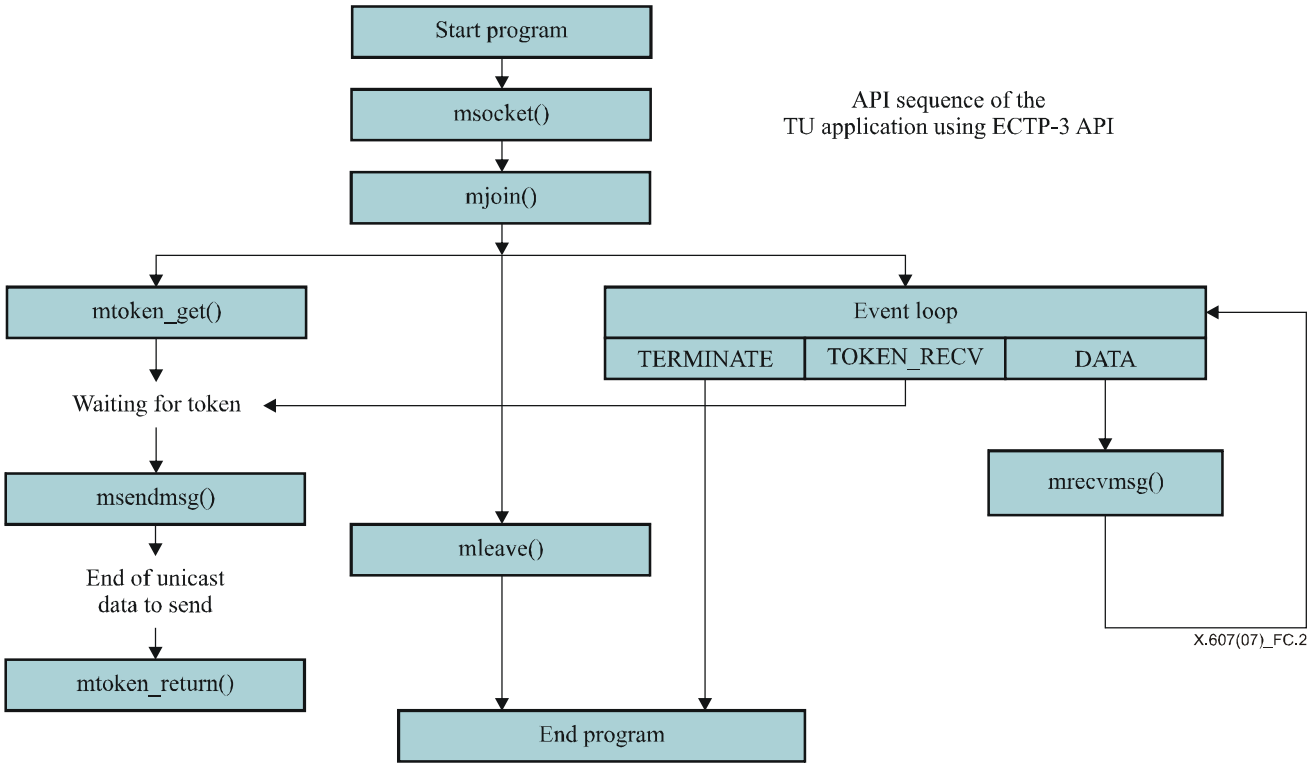

Figure C.2 – Example sequence of ECTP-3 API functions called by TS-user

ICS 35.100.40

Price based on 40 pages