The implementation specification of the OSIRIDE Transport Services and Protocol

Sivlio Comel
Enrico Gregori
Massimo Sartorio

Report C83-18

The impementation specification of the OSIRIDE Transport Services and Protocol

Silvio Comel [2]
Enrico Gregori[2]
Massimo Sartorio[1]

[1] Ing. C. Olivetti & c. - v. Jervis 77 - 10015 IVREA
[2] ITALSIEL - v. Isonzo 21/b - 00155 ROMA

Doc. OSIRIDE/83/TRA/02

September 1983

TABLE OF CONTENTS

LIST OF FIGURES

FIRST PART

1.0 INTRODUCTION

This document is divided in two parts.

The first part describes the Transport Layer. It specifies the services provided to the Session Layer and the protocol utilized to provide such services .

This first part is a subset of OSI DP8072 (Berlin January 83) and DP 8073 (Wien March 83) according to these Draft Proposal. Some points not well specified or for further study in DP 8073 and DP 8072 are here completely described.

1.1 SCOPE OF THE LAYER

The main issue of the Transport Layer is to optimize the communication resourses in order to provide the Quality of Service required by communicating TS users at minimum cost. It also hides from TS users the difference of Quality of Service provided by the Network Service. All the protocols defined in the Transport Layer have an end-to-end significance where the ends are defined as correspondent Transport Entities.

The Transport Layer provides to Session Layer a transport service for transparent data transfer between Session Entities and relieve them from any concern with the detailed way in which reliable and cost-effective transfer of data is achieved.

This part of the document is based on the concepts developed in the Reference Model for Open Systems Interconnection (OSI), DIS 7498, and makes use of the following terms defined in that standard:

1. expedited transport-service-data-unit;;
2. transport-connection;
3. transport-connection endpoint;
4. transport Layer
5. transport-protocol-data-unit;
6. transport Service;
7. transport-service-access-point;
8. transport-service-data-unit;
9. Network Layer;

10. Network Service;

11. network-connection;

12. network-connection-endpoint;

13. network service data unit;

14. interface flow control.

1.2 DEFINITIONS

1. Service-user
An abstract representation of the totality of those entities in a single system that make use of a service.

2. Service-provider
An abstract machine which models the behaviour of the totality of the entities providing the service, as viewed by a service-user.

3. Layer service
A service provided by a layer of the Reference Model.

4. Primitive
An abstract, implementation-independent element of an interaction between a service-user and the service-provider.

5. Request (primitive)
A primitive issued by a service user to invoke some service.

6. Indication (primitive)
A primitive issued by a service provider either

to invoke some service; or

to indicate that a service has been invoked by a service-user.

7. Response (primitive)
A primitive issued by a service-user to complete at a particular service-access-point some procedure previously invoked by an indication at that service-access-point.

8. Confirm (primitive)
A primitive issued by a service-provider to complete, at a particular service-access-point, some procedure previously invoked by a request at that service-access-point.

9. Calling TS user
A TS user that initiates a transport-connection establishment request.

10. Called TS User
A TS user with whom a calling TS user wishes to establish a transport-connection.

11. Sending TS user
A TS user that acts as a source of data during the data transfer phase of a transport-connection[2].

12. Receiving TS user
A TS user that acts as a sink of data during the data transfer phase of a transport-connection[2].

13. Equipment
Hardware or software or a combination of both; it need not be physically distinct within a computer system.

14. Transport service user
An abstract representation of the totality of those entities within a single system that make use of the transport service.

15. Network service provider
An abstract machine which models the totality of the entities providing the network service, as viewed by a transport entity.

16. Local matter
A decision made by a system concerning its behaviour in the Transport Layer that is not subject to the requirements of this protocol.

17. Initiator
A transport entity that initiates a CR TPDU.

18. Responder
A transport entity with whom an initiator wishes to establish a transport connection.[3]

19. Sending transport entity
A transport entity that sends a given TPDU.

20. Receiving transport entity
A transport entity that receives a given TPDU.

21. Preferred class

---

[1] calling TS users and called TS users are defined with respect to a single connection. A TS user can be both a calling and a called TS user simultaneously.

[2] A TS user can be both a sending and a receiving TS user simultaneously.

[3] Initiator and responder are defined with respect to a single transport connection. A transport entity can be both an initiator and responder simultaneously.

The protocol class that the initiator indicates in a CR TPDU as its first choice for use over the transport connection.

22. Alternative class
A protocol class that the initiator indicates in a CR TPDU as an alternative choice for use over the transport connection.

23. Proposed class
A preferred class or an alternative class.

24. Selected class
The protocol class that the responder indicates in a CC TPDU that it has chosen for use over the transport connection.

25. Proposed parameter
The value for a parameter that the initiator indicates in a CR TPDU that it wishes to use over the transport connection.

26. Selected parameter
The value for a parameter that the responder indicates in a CC TPDU that it has chosen for use over the transport connection.

27. Error indication
An N-RESET indication, or an N-DISCONNECT indication with a reason code indicating an error, that a transport entity receives from the NS-provider.

28. Invalid TPDU
A TPDU which does not comply with the requirements of the ISO International Standard for structure and encoding.

29. Protocol error
A TPDU whose use does not comply with the procedures for the class.

30. Sequence number

A. The number in the TPDU-NR field of a DT TPDU which indicates the order in which the DT TPDU was transmitted by a transport entity.

B. The number in the YR-TU-NR field of an AK or RJ TPDU which indicates the sequence number of the next DT TPDU expected to be received by a transport entity.

31. Transmit window
The set of consecutive sequence numbers which a transport entity has been authorised by its peer entity to send at a given time on a given transport connection.

32. Lower window edge
The lowest sequence number in a transmit window.

33. Upper window edge

The sequence number which is one greater that the highest sequence number in the transmit window.

34. Upper window edge allocated to the peer entity
The value that a transport entity communicates to its peer entity to be interpreted as its new upper window edge.

35. Closed window
A transmit window which contains no sequence number.

36. Window information
Information contained in a TPDU relating to the upper and the lower window edges.

37. Frozen reference
A reference which is not available for assignment to a connection because of the requirements of 3.1.17.

38. Unassigned reference
A reference which is neither currently in use for identifying a transport connection nor in a frozen state.

39. Transparent (data)
TS-user data which is transferred intact between transport entities and which is unavailable for use by the transport entities.

40. Owner (of a network connection)
The transport entity that issued the N-CONNECT request leading to the creation of that network connection.

41. Retained TPDU
A TPDU which is subject to the retransmission procedure or retention until acknowledgement procedure and is available for possible retransmission.

1.3 ABBREVIATIONS

TS	: Transport Service
TC	: Transport-connection
TSAP	: Transport-service-access-point
TSDU	: Transport-service-data-unit
QOS	: Quality of Service
TPDU	: Transport protocol data unit
NS	: Network service
NSDU	: Network service data unit

NC	: Network connection
TS-user	: Transport service user
TSAP	: Transport service access point
NS-provider	: Network service provider
NSAP	: Network service access point

1.4 TYPES OF TRANSPORT PROTOCOL DATA UNITS

CR TPDU	Connection request TPDU
CC TPDU	Connection confirm TPDU
DR TPDU	Disconnect request TPDU
DC TPDU	Disconnect confirm TPDU
DT TPDU	Data TPDU
ED TPDU	Expedited data TPDU
AK TPDU	Data acknowledge TPDU
EA TPDU	Expedited acknowledge TPDU
RJ TPDU	Reject TPDU
ER TPDU	Error TPDU

1.5 TPDU FIELDS

LI	Length indicator (field)
CDT	Credit (field)
TSAP-ID	Transport service access point identifier (field)
DST-REF	Destination reference (field)
SRC-REF	Source reference (field)
EOT	End of TSDU mark
TPDU-NR	DT TPDU number (field)
ED-TPDU-NR	ED TPDU NUMBER (field

YR-TU-NR Sequence number response (field)

1.6 TIMER VARIABLES

TTR Time to try reassignment after failure

TWR Time to wait for reassignment

TS1 Supervisory timer for connection establishment

TS2 Supervisory timer for connection release.

2.0 TRANSPORT LAYER MODEL

A transport entity communicates with its TS-users through one or more TSAPs by means of the service primitives as defined by the transport service. Service primitives will cause or be the result of transport protocol data unit exchanges between the peer transport entities supporting a transport connection. These protocol exchanges are effected using the services of the Network Layer as defined by the Network Service Definition N 2610 (October 82) through one or more NSAPs.

Transport connection endpoints are identified in end systems by an internal, implementation dependent, mechanism so that the TS-user and the transport entity can refer to each transport connection.

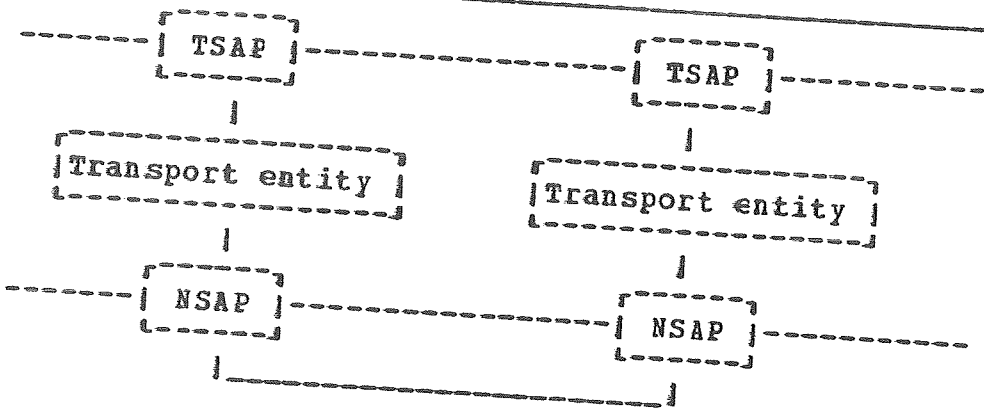


Fig. 1. Transport layer model

2.1 SERVICES ASSUMED FROM THE NETWORK LAYER

The OSIRIDE Network layer is based on the X.25 Italian Public Data Network (ITAPAC).
The basic service of a generally speaking Network Layer is to provide transparent data transfer between transport entities uniquely identified by Network addresses on network connection. The Network Layer provides the means to establish, maintain and release network connections. Network connection features are identified by its Quality of Service parameters. The ITAPAC Quality of Service parameters are:

Throughput (initiator -> responder)

Throughput (responder -> initiator)

Each class of throughput has a transmit window associated. The Network service allows the structure and detailed content of submitted data to be determined exclusively by layers above the Network Layer. The Transport Layer requests such network service by means of the network's primitives listed in Fig. 2.

Primitive		X	Parameters	X/Z
N-CONNECT	request	X	Called Address,	X
	indication	X	Calling Address,	X
	response	X	NS-User data,	Z
	confirm	X	QOS parameter set	X
N-DATA	request	X	NS-User-Data,	X
	indication	X		
N-RESET	request	X		
	indication	X		
	response	X		
	confirm	X		
N-DISCONNECT	request	X	NS-User-Data.	Z
	indication	X		

Fig. 2. Network Service Primitives used by OSIRIDE Transport

KEY:

X - The Transport Protocol assumes that this facility is provided in all networks.

Z - The Transport Protocol does not use this parameter.

For detailed informations see interface 4-3 second part 2.2.

2.2 SERVICES PROVIDED TO THE SESSION LAYER

The Transport Layer provides the means to establish, maintain and release transport connections. Transport connections provide transparent duplex transmission between two Session entities uniquely identified by transport addresses.

More than one transport connection can be established between the same pair of transport addresses. A session entity uses transport-connection-endpoint-identifier provided by the Transport Layer to distinguish between transport-connection-endpoints.

The operation of one transport-connection is independent of the operation of all other except for the limitations imposed by the finite resources available to the Transport Layer.

Primitive		Parameters
T-CONNECT	request indication	Called Address, Calling Address, Expedited Data Option, Quality of Service,
T-CONNECT	response confirm	Responding Address, Quality of Service, Expedited Data option,
T-DATA	request indication	TS User-Data.
T-EXPEDITED DATA	request indication	TS User-Data.
T-DISCONNECT	request	
T-DISCONNECT	indication	Disconnect reason.

Fig. 3. Transport Service Primitives

2.3 TRANSPORT SERVICE CHARACTERISTICS

The Transport Service provides transparent transfer of data between TS users. It relieves these TS users from any concern about the detailed way in which supporting communications media are utilized to achieve this transfer.

The Transport Service provides for the following:

1. Quality of Service selection:

   The Transport Layer is required to optimize the use of available communications resources to provide the Quality of Service required by communicating TS users at minimum cost. Quality of Service is specified through the selection of values for Quality of Service parameters. The parameters supported by OSIRIDE release one are throughput and failure probability.

2. End-to-end significance:

The Transport Service provides for the transfer of data between two TS users.

3. Transparency of transferred information:

The Transport Service provides for the transparent transfer of octet-aligned TS user-data and/or control information. It neither restricts the content, format or coding of the information, nor does it ever need to interpret its structure or meaning.

4. TS user addressing:

The Transport Service utilizes a system of addressing which is mapped into the addressing scheme of the supporting Network Service. Transport-addresses can be used by TS users to refer unambiguously to TSAPs.

2.3.1 General Description

The Transport Service offers the following features to a TS user.

1. The means to establish a TC with another TS user for the purpose of exchanging TSDUs. More than one TC may exist between the same pair of TS users.

2. Associated with each TC at its time of establishment, the opportunity to request, negotiate, and have agreed by the TS provider a certain Quality of Service as specified by means of Quality of Service parameters.

3. The means of transferring TSDUs on a TC. The transfer of TSDUs which consists of an integral number of octets is transparent, in that the boundaries of TSDUs and the contents of TSDUs are preserved unchanged by the TS provider and there are no constraints on the TSDU content imposed by the TS provider.

4. The means by which the receiving TS user may flow control the rate at which the sending TS user may send TSDUs.

5. The means of transferring separate expedited TSDUs when agreed to by both TS users. Expedited TSDUs transfer is subject to a different flow control from normal data across the TSAP.

6. The unconditional and therefore possible destructive release of a TC.

2.3.1.1 Quality of Transport Service

The term Quality of Service refers to certain characteristics of a TC as observed between the endpoints.

Quality of Service is described in terms of QoS parameters.

These parameters give TS users a method of specifying their needs, and give the TS provider a basis for protocol selection.

The OSIRIDE release one Transport layer supports two Quality of Service parameters: **THROUGHPUT and RESILIENCE.** The Transport user specifies throughput in both directions, (i.e. two different figures are allowed for the two directions of the transport connection). The QoS requested by the calling TS user may be made poorer either by the TS provider following the T-CONNECT request, or by the called TS user, following the T-CONNECT indication. In applying this to some QOS parameters this may mean that:

1. a throughput becomes lower;
2. the failure probability becomes higher.

The so negotiated QOS values them apply throughout the lifetime of the TC.

The view of QOS at each end of an established TC is always the same.

2.3.2 Primitives

Sequence of Transport Service Primitives

The clause defines the constraints on the sequences in which the TS primitives may occur. The constraints determine the order in which TS primitives occur, but do not fully specify when they may occur. Other constraints, such as flow control of data, will affect the ability of a TS user or TS provider to issue a TS primitive at any particular time.

A complete listing of TS primitives appears in Fig. 4 pag. 14

Phase	Service	Primitive	Parameters
TC establish ment	TC establish ment	T-CONNECT request	(Called Address, Calling Address, Quality of Service, T_Expedited data op
tion).			
		T-CONNECT indication	(Called Address, Calling Address, Quality of Service, T_Expedited data op
tion).			
		T-CONNECT	(Quality of Service
,		response	Responding Address, TS User-Data, T_Expedited data op
tion).			
		T-CONNECT	(Quality of Service
,		confirm	Responding Address, T_Expedited data op
tion).			
Data Transfer	Normal Data Transfer	T-DATA request	(TS User-Data)
		T-DATA indication	(TS User-Data)
	Expedited Data Transfer	T-EXPEDITED DATA request	(TS User-Data)
		T-EXPEDITED DATA indication	(TS User-Data)
TC Release	TC Release	T-DISCONNECT request	
		T-DISCONNECT indication	(Disconnect Reason)

Fig. 4. Transport Service Primitives

2.3.2.1 Relation of TS Primitives at the two TC Endpoints

A TS primitive issued at one TC endpoint will, in general, have consequences at the other TC endpoint. The relations of TS primitives of each type to TS primitives at the other TC endpoint are summarized in figure Fig. 5 pag. 16.

However, a T-DISCONNECT request or indication TS primitive may terminate any of the other sequences before completion.

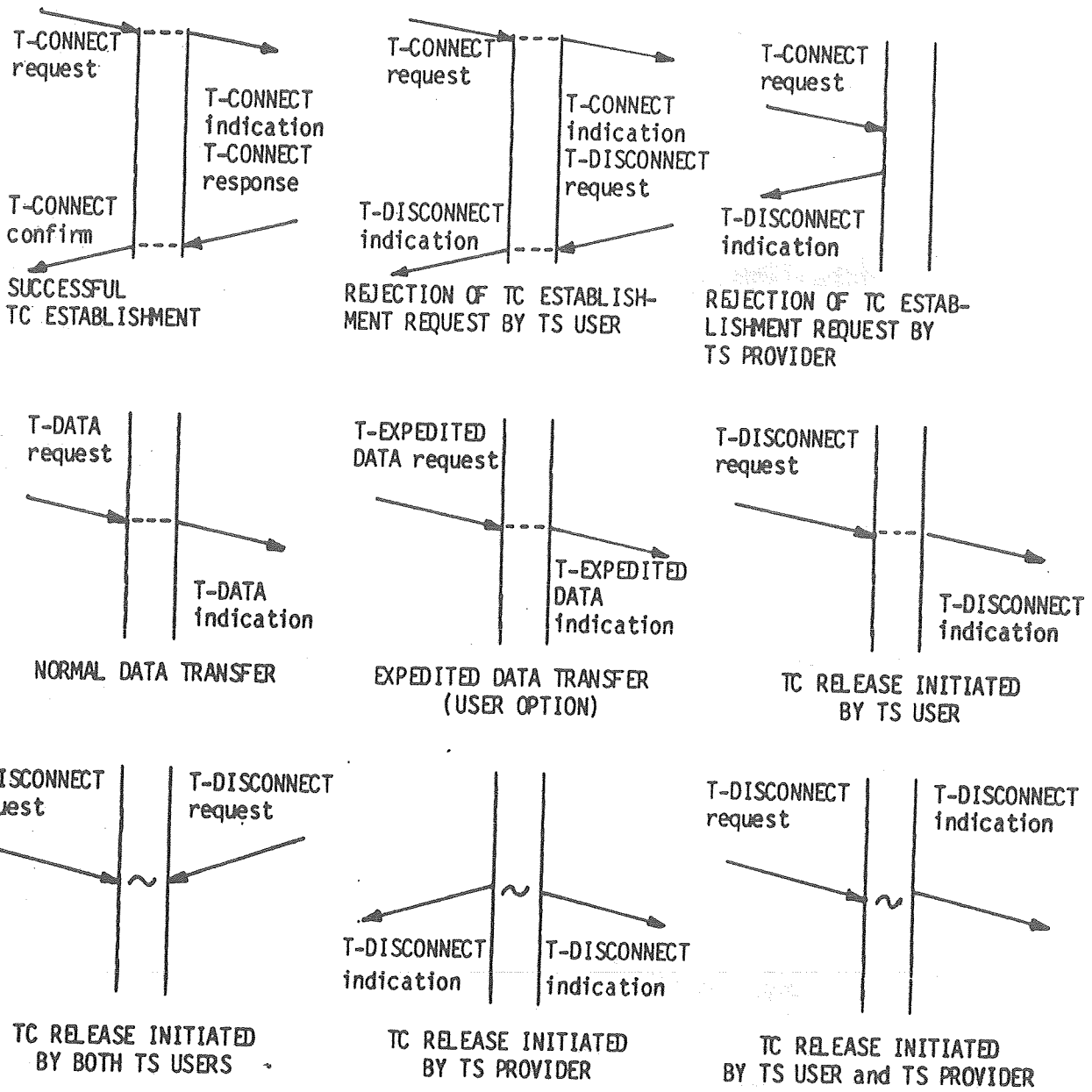


Fig. 5. Summary of Transport Service Primitive Time Sequence Diagram

2.3.3 Transport Conection Establishment

The TC estalbishment TS primitives can be used to establish a TC, provided the TS users exist and are known to the TS provider. Simultaneous T-CONNECT requests at the two TSAPs are handled independently by the TS provider.

Simultaneous T-CONNECT requests typically result in a corresponding number of TCs.

2.3.3.1 Types of TS Primitives and Parameters

The following figure indicates the types of TS primitives and the parameters needed for TC establishment.

TS-Primitive CONNECT Parameter	T-CONNECT request	T-CONNECT indication	T-CONNECT response	T-CONNECT co
Called Address	X	X(=)		
Calling Address	X	X(=)		
Expedited Data Option	X	X(=)	X	
Quality of SERVICE	X	X	X	

Fig. 6. TC Establishment Primitives and Parameters

Key:

1. X: mandatory parameter
2. X(=): the value of that parameter is identical to the value of the corresponding parameter of the preceding TS primitive

Addresses: The parameters which take addresses as values all refer to TSAPs. These addresses are unique within the scope of TSAP addresses. The format of the transport address in OSIRIDE release one transport layer is the following:

Transport Layer model

Called Address: The called address parameter conveys the address of the TSAP to which the TC is to be established.

Calling Address: The calling address parameter conveys the address of the TSAP from which the TC has been requested.

Expedited Data Option: The expedited data option parameter indicates whether the expedited data option is to be available on the TC.
If this service is declared not available, it cannot be used on the TC. The value of the parameter is either 'Expedited Data Service selected' or 'Expedited Data Service not selected'. The values of the various primitives are related so that:

1. in the T-CONNECT request primitive, either of the defined values may occur;

2. in the T-CONNECT indication primitive, the value is equal to the value in the T-CONNECT request primitive;

3. in the T-CONNECT response primitive, the value is either 'Expedited Data Service not selected' or is equal to the value in the T-CONNECT indication primitive;

4. in the T-CONNECT confirm primitive, the value is equal to the value in the T-CONNECT response primitive.

Quality of Service: The OSIRIDE release one Transport layer supports two Quality of Service parameters: THROUGHPUT and RESILIENCE. The Transport user specifies throughput in both directions.
Throughput parameters allowed values are:

1. VERY HIGH

2. HIGH

3. LOW

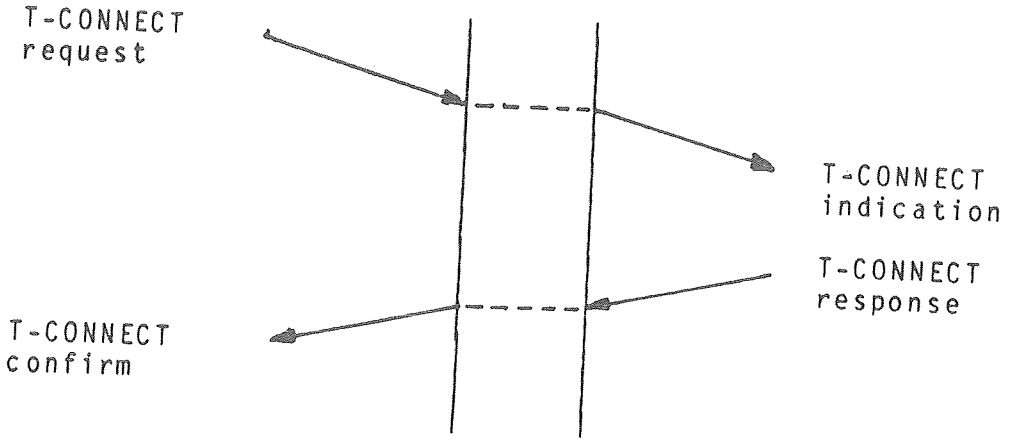
4. VERY LOW

Resilience allowed values are:

1. LOW

2. HIGH

Sequence of TS primitives: The sequence of TS primitives in a successful TC establishemnt is defined by the following time sequence diagram.



The TC establishment procedure may fail either due to the inability of the TS provider to establish a TC or due to the unwillingness of the called TS user to accept a T-CONNECT indication. The TC establishment may also fail due to either of the TS users releasing the TC before the T-CONNECT confirm has been delivered to the calling TS user.

Negotiation of Expedited Data Transfer Service: The expedited TSDU transfer is only made available when specifically requested and agreed to by both TS users when the TC is established. This service is always bidirectional. The procedure for negotiating the use of the expedited TSDU transfer is as follows:

1. the calliag TS user may request or not request the use of the expedited TSDU transfer feature;
2. if the calling TS user does not request the use of the expedited TSDU transfer feature, the called TS user is not allowed to request its use;
3. if the calling TS user does request the use of the expedited TSDU transfer feature, the called TS user may agree to the use of the expedited TSDU transfer on the TC, in which case the TS provider is required to provide it. The called TS user may refuse the use of the expedited TSDU transfer in which case the service will not be used.

2.3.4 Data Transfer Phase

The TS provider provides for an exchange of TSDUs simultaneously in both directions. The TS provider preserves the integrity, the sequence and boundaries of the TSDUs.

2.3.4.1 Types of TS Primitives and Parameters

Fig. 7 indicates the types of TS primitives and the parameters needed for data transfer.

Primitive / Parameter	T-DATA request	T-DATA indication
TS User-Data	X	X(=)

Fig. 7. Data Transfer Primitives and Parameters

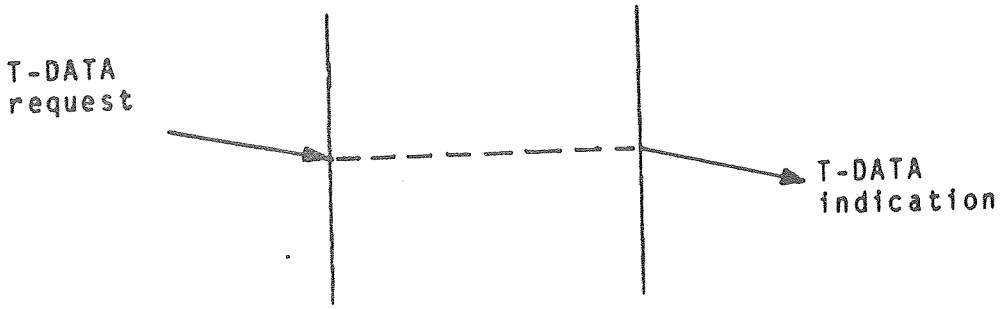
Key:

X: mandatory parameter

X(=): the value of that parameter is identical to the value of the corresponding parameter of the preceding TS primitive.

TS User-Data: The TS User-data parameter is a TSDU. A TSDU consists of an integral number of octets greater than zero.

Sequence of TS Primitives: The operation of the TS provider in transferring TS User-data can be modelled as a queue of unknown size within the TS provider. The ability of a TS user to issue a T-DATA request depends on the state of the queue (see interface 4/5). The ability of the TS provider to issue a T-DATA indication depends on the receiving TS user (see interface 4/5).
The sequence of TS primitives in a successful data transfer is defined in the following time sequence diagram.



2.3.4.2 Expedited Data Transfer Service

The expedited data transfer service provides a further means of information exchange on a TC in both directions simultaneously.

The TS provider guarantees that an expedited TSDU will not be delivered after any sebsequently submitted normal TSDU or expedited TSDU on that TC.

In particular, expedited data will be delivered also when the receiving TS user is not accepting normal data. However, the amount of normal data bypassed cannot be predicted.

Types of TS Primitives and Parameters: Fig. 8 indicates the types of TS primitives and the parameters needed for expedited data transfer.

Primitive / Parameter	T-EXPEDITED-DATA request	T-EXPEDITED-DATA indication
TS User-Data	X	X(=)

Fig. 8. Expedited TS Primitives and Parameters

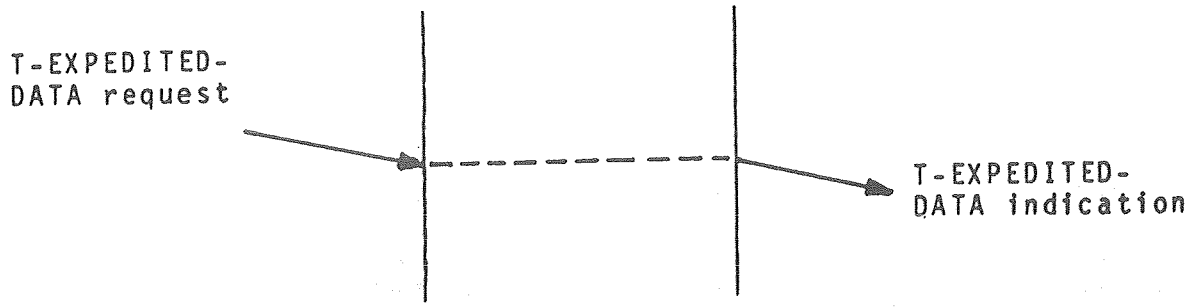
Key:

x: mandatory parameter

x(=): the value of that parameter is identical to the value of the corresponding parameter of the preceding TS primitive

TS User-Data: The TS user-data parameter is an expedited TSDU. An expedited TSDU consists of an integral number of octects between 1 and 16 inclusive.

Sequence of TS Primitives: The sequence of TS primitives in a successful expedited data transfer defined in the following time sequence diagram. It is recommended that the OSIRIDE Session Layer uses careful the expedited data transfer service.



2.3.5 Transport Connection Release Phase

The TC release TS primitives are used to release a TC. The release may be performed:

1. by either or both of the TS users to release an established TC;

2. by the TS provider to release an established TC; all failures to maintain a TC are indicated in this way;

3. by the TS provider, to indicate its inability to establish a requested TC.

A request for release cannot be rejected. The Transport Service does not guarantee delivery of any TS user-data once the release phase is entered.

2.3.5.1 Types of TS Primitives and Parameters

Fig. 9 pag. 23 indicates the types of TS primitives and the parameters needed for TC release.

Primitives / Parameter	T-DISCONNECT request	T-DISCONNECT indication
Reason		X

Fig. 9. TC Release Primitives and Parameters

Key:

X: mandatory parameter

Reason: The reason parameter gives information indicating the cause of the TC release. The reason is one of the following:

1. remote TS user invoked;

2. TS provider invoked.
   This reason may be of transient or permanent nature.

Sequence of TS Primitives when Releasing an Established Transport Connection: The sequence of TS primitives depends on the origin or origins of the TC release action. The sequence may be:

A. invoked by one TS user, with a T-DISCONNECT request from that TS user leading to a T-DISCONNECT indication to the other;

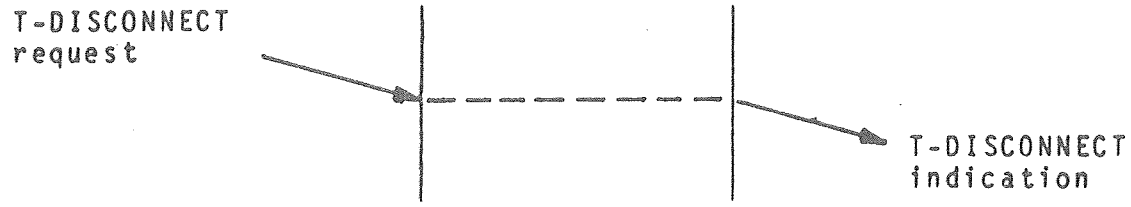
B. invoked by both TS users, with a T-DISCONNECT request from each of the TS users;

C. invoked by the TS provider, with a T-DISCONECT indication to each of the TS users;

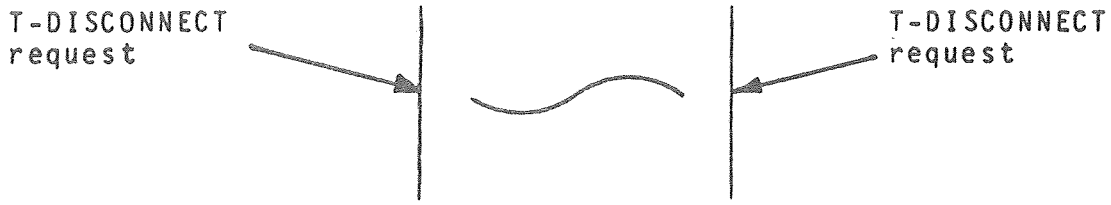
D. invoked independently by one TS user and the TS provider, with a T-DISCONNECT request from the initiating TS user and a T-DISCONNECT indication to the other.

The sequence of TS primitives in these four cases are expressed in the following time sequence diagrams. After a T_CONN request the OSIRIDE Session Layer does not send any T_DISC request until the reception of a T_CONN confirm.

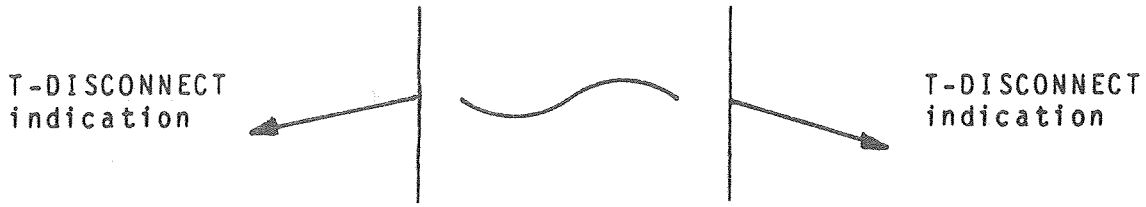
a) TS user invocation



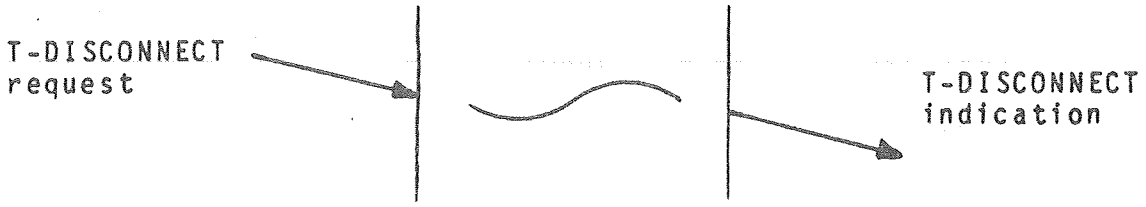
b) simultaneous invocation by both TS users



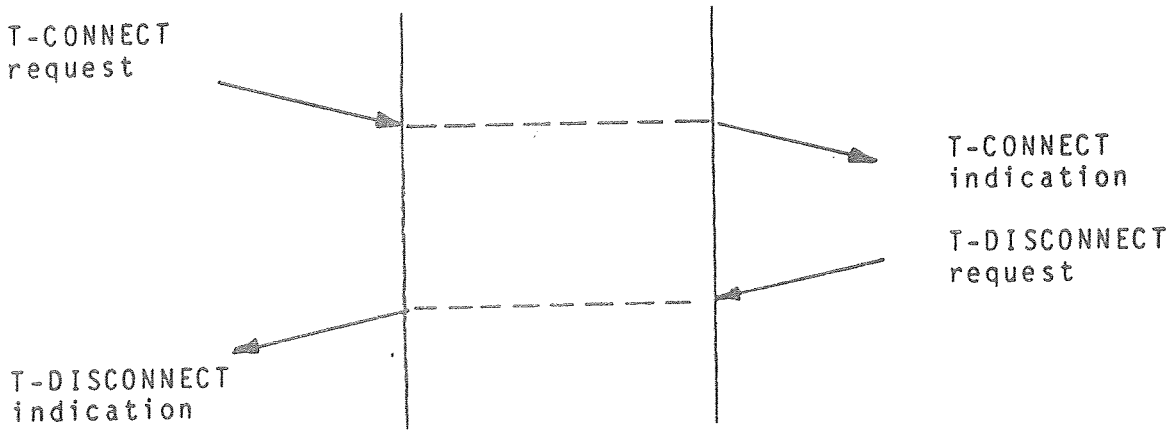
c) TS provider invocation



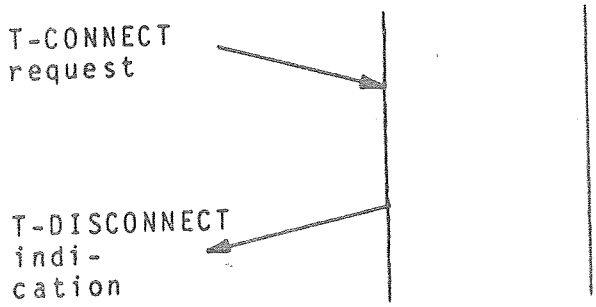
d) simultaneous TS user and TS provider invocations



Sequence of TS Primitives in TS User Rejection of a TC Establishment: A TS user may reject a TC establishment attempt by a T-DISCONNECT request. In the T-DISCONNECT indication the reason parameter will indicate that the called TS user initiated the disconnection. The sequence of events is defined in the following time sequence diagram.



Sequence of TS Primitives in a TS Provider Rejection of a TC Establishment Attempt: If the TS provider is unable to establish a TC, it indicates this to the calling TS user by a T-DISCONNECT indication. The reason parameter indicates that the TS provider is the source of the T-DISCONNECT indication. The sequence of events is defined in the following time sequence diagram.



3.0 TRANSPORT PROTOCOL SPECIFICATION

The transport protocol is a set of elements of procedures which perform the enhancement of the Network services. The ISO/SC 16/WG 6 has "standardized" five classes of transport protocol (see DP 8073).
The OSIRIDE release one Transport protocol supports two classes of protocol (classes 2 and 3) according to the actual ISO conformance statement.
The options supported are:

Expedited Data transfer service (classes 2 and 3)

No explicit flow control (class 2)

3.1 ELEMENTS OF PROCEDURE

This clause contains elements of procedure which are used in the specification of protocol classes in clause XXXX. These elements are not meaningful on their own.

The procedures define the transfer of TPDUs whose structure and coding is specified in chapter 7. Transport entities shall accept and respond to any TPDU received in a valid NSDU and may issue TPDUs initiating specific elements of procedures specified in this clause.

Where NSDUs and TPDUs and parameters used are not significant for a particular element of procedure, they have not been included in the specification.

3.1.1 Assignment to network connection

Purpose

The procedure is used in all classes to assign transport connections to network connections.

Network service primitives

The procedure makes use of the following network service primitives:

1. N-CONNECT

2. N-DISCONNECT

Procedure

Each transport connection shall be assigned to a network connection. The initiator may assign the transport connection to an existing network connection of which it is the owner or to a new network connection which it creates for this purpose (see 3.1.5).

During the resynchronization and reassignment after failure procedures, a transport entity may reassign a transport connection to another network connection joining the same NSAPs, provided that it is the owner of the network connection.

The responding transport entity becomes aware of the assignment when it receives:

1. a CR TPDU during the connection establishment procedure (see 3.1.5) ; or

2. an RJ TPDU or a retransmitted CR or DR TPDU during the resynchronization (see 3.1.14) and reassignment after failure (see 3.1.12) procedures.

When a network connection is created, the quality of service requested is a local matter, although it will normally be related to the requirements of transport connection(s) expected to be assigned to it.
A network connection with no transport connection assigned to it is available only after initial establishment.
When all the transport connections assigned to it have been released, the network connection will be released.
In OSIRIDE the owner of a network connection shall normally release it. Furthermore, if the last TPDU is sent on this connection from the owner side it does not release it immediately after the transmission of the final TPDU - either a DR in response to CR or a DC in response to DR. The OSIRIDE owner of the network connection will wait for an appropriate delay; such delay will allow the TPDU concerned to reach the other transport entity allowing the freezing of any resources associated with the transport connection concerned.

3.1.2 Transport protocol data unit (TPDU) transfer

Purpose

The TPDU transfer procedure is used in all classes to convey transport protocol data units in user data fields of network service primitives.

Network Service Primitives

The procedure uses the following network service primitive:

1. N-DATA

Procedure

The transport protocol data units (TPDUs) defined for the protocol are listed in section 1.4.

Transport entities shall transmit and receive TPDUs as NS-user data parameters of N-DATA primitives.

3.1.3 Segmenting and reassembling

Purpose

The segmenting and reassembling procedure is used in all classes to map TSDUs onto TPDUs.

TPDUs and parameter used

The precedure makes use of the following TPDU and parameter:

1. DT TPDUs;

   End of TSDU.

Procedure

A transport entity may map a TSDU on to an ordered sequence of one or more DT TPDUs. This sequence shall not be interrupted by other DT TPDUs on the same transport connection.

All DT TPDUs except the last DT TPDU in a sequence greater than one must have a length of data greater than zero. Violation of this rule is a protocol error.

The end of TSDU parameter of DT TPDU indicates whether or not there are subsequent DT TPDUs in the sequence. There is no requirement that all the DT TPDUs be of maximium length selected during the connection establishment (see second part Procedure Transmit).

3.1.4 Concatenation and separation

Purpose

The procedure for concatenation and separation is used in classes 2,3 to convey multiple TPDUs in one NSDU.

Procedure

Each transport connection shall be assigned to a network connection. The initiator may assign the transport connection to an existing network connection of which it is the owner or to a new network connection which it creates for this purpose (see 3.1.5).

During the resynchronization and reassignment after failure procedures, a transport entity may reassign a transport connection to another network connection joining the same NSAPs, provided that it is the owner of the network connection.

The responding transport entity becomes aware of the assignment when it receives:

1. a CR TPDU during the connection establishment procedure (see 3.1.5) ; or

2. an RJ TPDU or a retransmitted CR or DR TPDU during the resynchronization (see 3.1.14) and reassignment after failure (see 3.1.12) procedures.

When a network connection is created, the quality of service requested is a local matter, although it will normally be related to the requirements of transport connection(s) expected to be assigned to it.
A network connection with no transport connection assigned to it is available only after initial establishment.
When all the transport connections assigned to it have been released, the network connection will be released.
In OSIRIDE the owner of a network connection shall normally release it. Furthermore, if the last TPDU is sent on this connection from the owner side it does not release it immediately after the transmission of the final TPDU - either a DR in response to CR or a DC in response to DR. The OSIRIDE owner of the network connection will wait for an appropriate delay; such delay will allow the TPDU concerned to reach the other transport entity allowing the freezing of any resources associated with the transport connection concerned.

3.1.2 Transport protocol data unit (TPDU) transfer

Purpose

The TPDU transfer procedure is used in all classes to convey transport protocol data units in user data fields of network service primitives.

Network Service Primitives

The procedure uses the following network service primitive:

1. N-DATA

Procedure

The transport protocol data units (TPDUs) defined for the protocol are listed in section 1.4.

Transport entities shall transmit and receive TPDUs as NS-user data parameters of N-DATA primitives.

3.1.3 Segmenting and reassembling

Purpose

The segmenting and reassembling procedure is used in all classes to map TSDUs onto TPDUs.

TPDUs and parameter used

The precedure makes use of the following TPDU and parameter:

1. DT TPDUs;

   End of TSDU.

Procedure

A transport entity may map a TSDU on to an ordered sequence of one or more DT TPDUs. This sequence shall not be interrupted by other DT TPDUs on the same transport connection.

All DT TPDUs except the last DT TPDU in a sequence greater than one must have a length of data greater than zero. Violation of this rule is a protocol error.

The end of TSDU parameter of DT TPDU indicates whether or not there are subsequent DT TPDUs in the sequence. There is no requirement that all the DT TPDUs be of maximium length selected during the connection establishment (see second part Procedure Transmit).

3.1.4 Concatenation and separation

Purpose

The procedure for concatenation and separation is used in classes 2,3 to convey multiple TPDUs in one NSDU.

Procedure

A transport entity may concatenate TPDUs from the same or different transport connections.

The set of concatenated TPDUs may contain:

1. any number of TPDUs that do not contain user data, provided that these TPDUs come from different transport connections; (DC, AK, RJ, EA)

2. no more than one TPDU containing user data; if this TPDU is present, it shall be placed last in the set of concatenated TPDUs. (CR, CC, ED, DT, DE, ER)

The TPDUs within a concatenated set may be distinguished by means of the length indicator parameter.

The end of a TPDU containing data is indicated by the termination of the NSDU.

3.1.5 Connection establishment

Purpose

The procedure for connection establishment is used in all classes to create a new transport connection.

Network service primitives

The procedure uses the following network service primitive: N-DATA

TPDUs and parameters used

The procedure uses the following TPDUs and parameters:

1. CR TPDU;

   CDT;

   DST-REF (set to zero);

   SRC-REF;

   CLASS and OPTIONS;

   calling TSAP-ID;

   called TSAP-ID;

   TPDU size (only 128)

version number;

throughput I->R (proposed);

throughput R->I (proposed);

additional option selection;

Echo request;

OSIRIDE like;

2. CC TPDU;

CDT;

DST-REF;

CLASS and OPTIONS (selected);

calling TSAP-ID;

called TSAP-ID;

TPDU size (selected);

version number;

throughput I->R (selected);

throughput R->I (selected);

additional option selection;

OSIRIDE like;

Procedure

A transport connection is established by means of one transport entity (the initiator) transmitting a Connection Request (CR) TPDU to the other transport entity (the responder), which replies with a Connection Confirm (CC) TPDU. Before sending the CR TPDU, the initiator assigns the transport connection being created to one network connection. During this exchange, all information and parameters needed for the transport entities to operate shall be exchanged or negotiated.

The initiator starts a timer TS1 at the time the CR TPDU is sent. This timer should be stopped when the connection is considered as accepted or refused. If the timer expires, the initiator releases the network connection and, in class 3 freeze the reference (see 3.1.17) of all the transport connection multiplexed on such network connection.

After receiving the CC TPDU for a class which includes the procedure for retention until acknowledgement of TPDUs the initiator shall acknowledge the CC TPDU as defined in Fig. 11 pag. 40.

The following information is exchanged:

1. references. Each transport entity chooses a reference which is 16 bits long and which is arbitrary except for the following restrictions:

   it shall not already be in use or 'frozen' (see XXXX),

   it shall not be zero.

   Each transport entity is responsible for selecting the reference which the partner will use. This mechanism is symmetrical. This mechanism also provides identification of the transport connection independent of the network connection. The range of references used for transport connections, in a given transport entity, is a local matter.

2. TSAP IDs Indicate the calling and called transport service access points identifiers.

3. initial credit. Only relevant for classes which include the explicit flow control function.

4. protocol class. The initiator should assume when it sends the CR TPDU that its preferred class will be agreed to, and commence the functions associated with that class. The first OSIRIDE's implementation supports class 3 and 2 and according to the ISO standard, it allows negotiation from class 3 to class 2 (with flow control option).

5. format. Only normal format is supported.

6. quality of service parameters. This defines the throughput and resilience. Only those QOS parameters are reflected which are of global significance.

7. TPDU size. The only size supported by OSIRIDE release one is 128.

8. the non-use of explicit flow control in class 2 is negotiated.

9. version number It is the release number of the OSIRIDE Transport layer.

10. Additional option selection It defines the selection to be made as to whether or not additional option are to be used.

11. Echo request It indicates whether this CR has to be used to perform an echo service (see second part Global interface).

12. OSIRIDE like It indicates whether an entity is or not OSIRIDE. It gives significance to Version number and Echo request parameters.

The negotiation rules for the options are such that the initiator may propose either to use or not to use the option. The responder may either accept the proposed choice or select an alternative choice as defined in Fig. 10.

During connection establishment the use of the expedited data parameter field of CR and CC TPDUs allows both TS-users to negotiate the use or non-use of the expedited data transport service as described in the transport service definition.

In class 2, whenever a transport entity requests or agrees to the transport expedited data transfer service, it shall also request or agree (respectively) to the use of explicit flow control.

Option	Proposition Made by the Initiator	Possible Selection by the Responder
Transport expedited data transfer service (Class 2,3)	Yes No	Yes or No No
Non-use of explicit flow control (Class 2 only)	Yes No	Yes or No No

Fig. 10. Negotiation of options during connection Establishment

3.1.6 Connection refusal

Purpose

The connection refusal procedure is used in all classes when a transport entity refuses a transport connection in response to a CR TPDU.

TPDUs and parameters used

The procedure makes use of the following TPDUs and parameters:

1. DR TPDU;

After receiving the CC TPDU for a class which includes the procedure for retention until acknowledgement of TPDUs the initiator shall acknowledge the CC TPDU as defined in Fig. 11 pag. 40.

The following information is exchanged:

1. references. Each transport entity chooses a reference which is 16 bits long and which is arbitrary except for the following restrictions:

   it shall not already be in use or 'frozen' (see XXXX),

   it shall not be zero.

   Each transport entity is responsible for selecting the reference which the partner will use. This mechanism is symmetrical. This mechanism also provides identification of the transport connection independent of the network connection. The range of references used for transport connections, in a given transport entity, is a local matter.

2. TSAP IDs Indicate the calling and called transport service access points identifiers.

3. initial credit. Only relevant for classes which include the explicit flow control function.

4. protocol class. The initiator should assume when it sends the CR TPDU that its preferred class will be agreed to, and commence the functions associated with that class. The first OSIRIDE's implementation supports class 3 and 2 and according to the ISO standard, it allows negotiation from class 3 to class 2 (with flow control option).

5. format. Only normal format is supported.

6. quality of service parameters. This defines the throughput and resilience. Only those QOS parameters are reflected which are of global significance.

7. TPDU size. The only size supported by OSIRIDE release one is 128.

8. the non-use of explicit flow control in class 2 is negotiated.

9. version number It is the release number of the OSIRIDE Transport layer.

10. Additional option selection It defines the selection to be made as to whether or not additional option are to be used.

11. Echo request It indicates whether this CR has to be used to perform an echo service (see second part Global interface).

12. OSIRIDE like It indicates whether an entity is or not OSIRIDE. It gives significance to Version number and Echo request parameters.

The negotiation rules for the options are such that the initiator may propose either to use or not to use the option. The responder may either accept the proposed choice or select an alternative choice as defined in Fig. 10.

During connection establishment the use of the expedited data parameter field of CR and CC TPDUs allows both TS-users to negotiate the use or non-use of the expedited data transport service as described in the transport service definition.

In class 2, whenever a transport entity requests or agrees to the transport expedited data transfer service, it shall also request or agree (respectively) to the use of explicit flow control.

Option	Proposition Made by the Initiator	Possible Selection by the Responder
Transport expedited data transfer service (Class 2,3)	Yes No	Yes or No No
Non-use of explicit flow control (Class 2 only)	Yes No	Yes or No No

Fig. 10. Negotiation of options during connection Establishment

3.1.6 Connection refusal

Purpose

The connection refusal procedure is used in all classes when a transport entity refuses a transport connection in response to a CR TPDU.

TPDUs and parameters used

The procedure makes use of the following TPDUs and parameters:

1. DR TPDU;

source reference;

reason;

2. ER TPDU;

reject code;

rejected TPDU parameter.

Procedure

If a transport connection cannot be accepted, the called transport entity shall respond to the CR TPDU with a DR TPDU. The reason shall indicate why the connection was not accepted. The source reference field in the DR TPDU is set to zero to indicate an unassigned reference.

If a DR TPDU is received the transport entity shall regard the connection as released.

If a supervisory timer TS1 has been set for this connection then the entity shall stop the timer on receipt of the DR or ER TPDU.

3.1.7 Normal Release

The release procedures is used by a transport entity in order to terminate a transport connection.

Network Service Primitives

The procedure makes use of the following network service primitive:

N-DATA.

TPDUs and parameters used

The procedure makes use of the following TPDUs and fields:

1. DR TPDU;

SRC-REF

DST-REF

clearing reason;

2. DC TPDU.

SRC-REF

DST-REF

Procedure

When the release of a transport connection is to be initiated the transport entity

- if it has previously sent or received a CC TPDU shall send a DR TPDU. It shall ignore all subsequently received TPDUs other than a DR or DC TPDU. On receipt of a DR or DC TPDU it shall consider the transport connection released;

- in other cases it shall wait for acknowledgement of the outstanding CR TPDU.

A transport entity that receives a DR TPDU shall

- if it has previously sent a DR TPDU for the same transport connection, consider the transport connection released.

- if it has previously sent a CR TPDU that has not been acknowledged by a CC TPDU, consider the connection refused (see 3.1.6).

- in other cases, send a DC TPDU and consider the transport connection released.

This requirement ensures that the transport entity is aware of the remote reference for the transport connection.

When the transport connection is considered as released the local reference is frozen.

After the release of a transport connection, if it is the last one, the network connection shall be released by its owner.

If a transport entity does not receive acknowledgement of a DR TPDU within time TS2, it should either reset or disconnect the network connection, and freeze all the references of the transport connections multiplexed on such network connection.

3.1.8 Error Release

Purpose

This procedure is used only in class 2 to release a transport connection on the occurrence of a N-DISCONNECT or N-RESET indication.

Network service primitives

The procedure makes use of the following service primitives:

1. N-DISCONNECT indication;

2. N-RESET indication.

Procedure

In class 2 when, on the network connection to which a transport connection is assigned, an N-DISCONNECT or N-RESET Indication occurs, both transport entities shall consider that the transport connection is released and so inform the TS-users.

In class 3 since error recovery is used, the occurrence of an N-RESET indicationor N-DISCONNECT indication will result in the invocation of the error recovery function.

3.1.9 Association of TPDUs with transport connection

Purpose

This procedure is used in all classes to interpret a received NSDU as TPDU(s) and, if possible, to associate each such TPDU with a transport connection.

Network service primitives

This procedure makes use of the following network service primitives:

1. N-DATA indication;

TPDUs and parameters used

This procedure makes use of the following TPDUs and parameters:

1. any TPDU except a CR TPDU;

   DST-REF.

2. CR TPDU;

   SRC-REF.

Procedure

Identification of TPDUs

If the received NSDU cannot be decoded (does not contain one or more correct TPDUs) or is corrupted then the transport entity shall issue an N-RESET request for the network connection and apply for all of the transport connections assigned the procedure

for handling of network signalled reset; or if the NSDU can be decoded and is not corrupted, the transport entity shall invoke the separation procedures and associate each of the individual TPDUs in the order in which they appear in the NSDU.

Association of individual TPDUs

If the received TPDU is a CR TPDU then, if it is a duplicate, as recognized by using the NSAPs in the network connection, and the SRC-REF parameter, then it is associated with the transport connection created by the original value of the CR TPDU; otherwise it is processed as requesting the creation of a new transport connection.

Otherwise, the DST-REF parameter of the TPDU is used to provisionally identify the transport connection. The following cases are distinguished:

1. if the DST-REF is not allocated to a transport connection (spurius TPDU), the transport entity shall respond on the same network connection with a DR TPDU if the TPDU is a CC TPDU, with a DC TPDU if the TPDU is a DR TPDU and will ignore the TPDU if neither a DR TPDU nor CC TPDU. No association with a transport connetion is made.

2. if the DST-REF is allocated to a connection, but the TPDU is received on a network connection to which the connection has not been assigned then there are two cases:

   if the transport connection is not assigned to any network connection (waiting for reassignment after failure) then the association with that transport connection is made.

   Otherwise, the TPDU is considered as having a DST-REF not allocated to a transport connection (case 1).

3. if the TPDU is a DC then it is associated with the transport connection to which the DST-REF is allocated, unless the SRC-REF is not the expected one, in which case the DC TPDU is ignored.

4. if the TPDU is a DR TPDU it is associated with the transport connection identified by the DST-REF parameter.

5. if none of the above cases apply then the TPDU is associated with the transport connection identified by the DST-REF parameter.

3.1.10 Data TPDU numbering

Purpose

Data TPDU numbering is used in classes 2 (when the explicit flow control option is selected) and 3. Its purpose is to enable the use of recovery, flow control and resynchronization functions.

TPDUs and parameters used

The procedure makes use of the following TPDU and parameter:

1. DT TPDU;

   TPDU-NR.

Procedure

A transport entity shall allocate the sequence number zero to the TPDU-NR of the first DT TPDU which it transmits for a transport connection. For subsequent DT TPDUs sent for the same transport connection, the transport entity shall allocate a sequence number one greater than the previous one.

When a DT TPDU retransmission is operated the TPDU-NR parameter will start from the value carried by the RJ TPDU.
TPDU-NR shall be assigned modulo 2**7.

3.1.11 Expedited Data Transfer

Purpose

Expedited data transfer procedures are selected during connection establishment.

Network service primitives

The procedure makes use of the following network service primitive:

1. N-DATA.

TPDUs and Parameters used

The procedure makes use of the following TPDUs and parameters:

1. ED TPDU;

   ED TPDU-NR.

2. EA TPDU;

   YR-TU-NR.

Procedures

The TS-user data parameter of each T-EXPEDITED DATA request is conveyed as the data field of an Expedited Data (ED) TPDU.

Each ED TPDU received must be acknowledged by an Expedited Acknowledge (EA) TPDU.

There may only be one ED TPDU unacknowledged at any time for each direction of a transport connection.

An ED TPDU with a zero length data field is a protocol error.

3.1.12 Reassignment after Failure

Purpose

The reassignment after failure procedure is used in Class 3 to commence recovery from an NS-provider signalled disconnect.

Network service primitives

The procedure uses the following network service primitive:

N-DISCONNECT indication

Procedure

When an N-DISCONNECT indication is received for the network connection to which a transport connection is assigned, the initiator shall either:

1. start its TTR timer (see *) and assign the transport connection to a different network connection. Until the timer runs out the transport entity shall try to assign for every received N-DISCONNECT indication. If the timer runs out the transport entity shall not try a new assignment. When a valid TPDU is received for the transport connection, the TTR timer is stopped. If no assignment is successful within this time then the transport connection shall be considered released. The reference shall be frozen; or

2. (this alternative is only available if a DR TPDU is retained) consider the transport connection as released and freeze the reference.

The responder shall wait for reassignment to take place for a time TWR following the N-DISCONNECT indication. The arrival of the first TPDU related to the transport connection and commencing resynchronization completes the reassignment after failure procedure. If reassignment does not take place within this time, the transport connection is considered released.

If assignment occurs successfully, both transport entities shall continue with resynchronization.

TTR is the Time to Try Reassignment timer. Its value is a local matter, but it must be less than TWR by at least the sum of the maximum disconnect propagation delay and transit delay of the network connections.

The maximum value of TTR is determined by the default value for TWR.

TWR is the Time to Wait for Reassignment timer. If the Reassignment Timer parameter is present in the CR TPDU, TWR may be set to this value plus the sum of the maximum disconnect propagation delay and transit delay of the network connections. If the Reassignment Time parameter is not present, a conventional value of 2 minutes should be used.

3.1.13 Retention until acknowledgement of TPDUs

Purpose

The retention until acknowledgement of TPDUs procedure is used in class 3 to enable and minimise retransmission after possible loss of TPDUs.

Network service primitives

The procedure uses the following network service primitive:

1. N-DATA.

TPDUs and parameters used

The procedure uses the following TPDUs and fields:

1. CR, CC, DR and DC TPDUs;

---

* TTR is not necessarily a timer. There is no need for a precise value so we decide to use a counter of reassignment trials.
If K is the maximum number of reassigment trials the mean value of this logical timer is K*2*(typical network delay).

After every T_DISCONNECT indication the variable that indicates the reassignment trials is incremented and a T_CONNECT request is sent. Reassignment is considered failed when such variable equals K. When a valid TPDU is received for the transport connection the counter for reassignment trials is reset to zero.

2. RJ, AK and EA TPDUs;

   YR-TU-NR.

3. DT TPDU;

   TPDU-NR;

4. ED-TPDU

   ED-TPDU-NR.

Procedures

Copies of the following TPDUs shall be retained upon transmission to permit their later retransmission:

CR, CC, DR, DT and ED TPDUs

except that if a DR is sent in response to a CR TPDU there is no need to retain a copy of the DR TPDU.

After each TPDU is acknowledged, as shown in Fig. 11, the copy need not be retained. Copies may also be discarded when the transport connection is released.

RETAINED TPDU	RETAINED UNTIL ACKNOWLEDGED BY
TPDU	
CR	CC, or ER TPDU
DR	DC, or DR (in case of collision) TPDU
CC	RJ,DT,AK,ED TPDUs
DT	AK or RJ TPDU for which YR-TU-NR is greater than TPDU-NR in the DT TPDU.
ED	EA TPDU for which the YR-TU-NR is equal to ED-TPDU-NR in the ED TPDU.

Fig. 11. Acknowledgement of TPDUs

3.1.14 Resynchronization

Purpose

The resynchronization procedures are used in Class 3 to restore the transport connection to normal after a reset or disconnect signalled by the NS-provider.

Network service primitives

The procedure makes use of the following network service primitive:

1. N-RESET indication.

TPDUs and parameters

The procedure uses the following TPDUs and parameters:

1. CR, DR, CC and DC TPDUs
2. RJ and EA TPDUs;

   YR-TU-NR.
3. DT TPDU;

   TPDU-NR
4. ED TPDU;

   ED TPDU-NR.

Procedure

A transport entity which is signalled of the occurrence of an NS-provider generated reset or disconnect shall carry out the active resynchronization procedures unless any of the following holds:

1. the transport entity is the responder and the error was a signalled disconnect (see [5]);
2. the transport entity is waiting for the T-CONNECT response.
3. the transport entity has elected not to reassign.

If either (a) or (b) (or both) hold, the transport entity carries out the passive resynchronization procedures. If (c) holds then no resynchronization takes place.

---

[5] In this case it is the initiator's responsibility to reassign the connection.

Active resynchronization procedures

The first applicable one of the following actions shall be taken:

1. if a CR TPDU is unacknowledged, then the transport entity shall retransmit it.

2. if a DR TPDU is unacknowledged, then the transport entity shall retransmit it.

3. otherwise, the transport entity shall carry out the data resynchronization procedures.

Passive resynchronization procedures

The transport entity shall not send any TPDUs until a TPDU has been received. When one has been received, the transport entity shall carry out the appropriate one of the following actions, depending on the TPDU:

1. if it is a DR TPDU, then the transport entity shall send a DC TPDU;

2. if it is a repeated CR TPDU then the transport entity shall carry out the action which is appropriate from the following:

   if a CC TPDU has already been sent, and acknowledged: treat as a protocol error

   if a DR TPDU is unacknowledged (whether or not a CC TPDU is unacknowledged): retransmit the DR TPDU, but setting the source reference to zero;

   if the T-CONNECT response has not yet been received from the user: take no action

   otherwise; retransmit the CC TPDU followed by any unacknowledged ED TPDU and any DT TPDU;

3. if it is an RJ TPDU then the first applicable one of the following actions shall be taken:

   if a DR TPDU is unacknowledged, then the transport entity shall retransmit it;

   otherwise, the transport entity shall carry out the data resynchronization procedures .

Data Resynchronization Procedures

If a CC TPDU was unacknowledged, the RJ TPDU should then be considered as acknowledging the CC TPDU. If a CC TPDU was never sent, the RJ TPDU should then be considered as a protocol error.

The transport entity shall carry out the following actions in the following order:

1. transmit RJ TPDU with YR-TU-NR field set to t! · TPDU-NR of the next expected DT TPDU;

2. wait for the RJ TPDU from the other transport e! ity, unless it has already been received; if a DR TPDU is eceived the transport entity shall send a DC, inform the T! user of the disconnection and take no further action

3. (re)transmit any ED TPDU which is unacknowledged,

4. (re)transmit any DT TPDUs which are unacknowledged, subject to any applicable flow control procedures[6]

5. if any duplicate ED TPDUs are received, the transport entity shall acknowledge them with an EA TPDU and then discard the duplicated ED TPDU.

3.1.15 Multiplexing and demultiplexing

Purpose

The multiplexing and demultiplexing procedures are used in classes 2 and 3 to allow several transport connections to share a network connction at the same time.

TPDUs and parameters used

The procedure makes use of the following TPDUs and parameters:

1. CC, DR, DC, DT, AK, ED, EA, RJ and ER TPDUs

   DST-REF

Procedure

The transport entities shall be able to send and receive on the same network connection TPDUs belonging to different transport connections.

---

[6] It is not necessary that the retransmitted DT TPDUs be exactly the same; in fact becouse of concatenation a different segmentation can be useful (see second part Procedure Transmit).

When performing demultiplexing the transport connection to which the TPDUs apply shall be distinguished by the DST-REF parameter except in the case of a CR TPDU.

Multiplexing allows the concatenation of TPDUs belonging to different transport connections to be transferred in the same N-DATA primitive.

3.1.16 Explicit Flow Control

Purpose

The explicit flow control procedure is used to regulate the flow of DT TPDUs independently of the flow control in the other layers.

TPDUs and parameters used

The procedure makes use of the following TPDUs and parameters:

1. CR, CC, AK and RJ TPDUs

   CDT.

2. DT

   TPDU-NR.

3. AK and RJ TPDUs

   YR-TU-NR.

Procedures

The procedures differ in different classes. They are defined in the clauses specifying separate classes.

3.1.17 Frozen References

Purpose

This procedure is used in order to prevent re-use of a reference while TPDUs associated with the old use of the reference may still exist.

Procedure

When a transport entity determines that a particular connection is released it shall place the reference which is allocated to the connection in a frozen state. While frozen, the reference shall not be re-used.

Freezing of the reference is always performed in OSIRIDE network because it is implicit in the mechanism that allows a circular reuse of the references.

3.1.18 Treatment of protocol errors

Purpose

The procedure for treatment of protcol errors is used in all classes to deal with invalid TPDUs.

After a protocol error the OSIRIDE Transport Entity shall transmit an ER TPDU containing the invalid TPDU and freeze the reference.

The receiver of an ER TPDU signals it to the LME and freezes the reference.

4.0 PROTOCOL CLASSES

Fig. 12 gives an overview of which procedure elements are included in each class.

PROCEDURE	CROSS REF	CL. 2	CL. 3
Assignment to N-C	3.1.1	*	*
TPDU Transfer	3.1.2	*	*
Segmenting & Reassembling	3.1.3	*	*
Concatenation & Separation	3.1.4	*	*
Connection Establishment	3.1.5	*	*
Connection Refusal	3.1.6	*	*
Normal release	3.1.7	*	*
Error Release	3.1.8	*	
Association of TPDUS with TC	3.1.9	*	*
DT TPDU Numbering	3.1.10	*(1)	*
Expedited data transfer	3.1.11	*(1)	*
Reassignment after failure	3.1.13		*
Retention until Ack. of TPDU	3.1.14		*
Resynchronisation	3.1.15		*
Multiplexing and demultiplexing	3.1.16	*	*
Explicit flow control	3.1.17	*	*
No Explicit flow control	3.1.17	*	
Frozen references	3.1.19		*
Treatment of prot. Errors	3.1.23	*	*

Fig. 12. Element of procedure

Key to Fig. 12:

1. * Procedure always included in class

2. (1) Not applicable in class 2 when no use of explicit flow control is selected.

5.0 SPECIFICATION FOR CLASS 2 - MULTIPLEXING CLASS

5.1 FUNCTIONS OF CLASS 2

Class 2 provides transport connections with or without individual flow control - no error detection or error recovery is provided.

If the network resets or clears, the transport connection is terminated without the transport release procedure and the TS-user is informed.

When explicit flow control is used a credit mechanism is defined allowing the receiver to inform the sender of the exact amount of data he is willing to receive and expedited data transfer is available.

5.2 PROCEDURES FOR CLASS 2

5.2.1 Procedures applicable at all times

The transport entities shall use the following procedures:

1. association of TPDUs with transport connection (see 3.1.9);
2. TPDU transfer (see 3.1.2);
3. treatment of protocol errors (see 3.1.18);
4. concatenation and separation (see 3.1.4);
5. error release (see 3.1.8).

additionally the transport entities may use the following procedure:

1. multiplexing (see 3.1.15).

When the no-flow control option is selected the OSIRIDE transport entity shall not use multiplexing even if it is allowed.

5.2.2 Connection establishment

The transport entity shall use the following procedures:

1. assignment to network connection (see 3.1.1); then

2. connection establishment (see 3.1.5) and, if applicable connection refusal (see 3.1.6).

5.2.3 Data transfer when non use of explicit flow control has been selected

If this option has been selected as a result on the connection establishment, the transport entities shall use the segmenting procedure (see 3.1.3).

The TPDU-NR field of DT TPDUs is not significant and may take any value.

Expedited data transfer is not applicable (see 3.1.11).

If the owner of the Network connection is an OSIRIDE transport entity it shall not use multiplexing.

5.2.3.1 Data transfer when use of explicit flow control has been selected

The sending transfer entity shall use:

1. segmenting (see 3.1.3); then

2. DT TPDU numbering (see 3.1.10);

The receiving transfer entity shall use:

1. DT TPDU numbering (see 3.1.10); if a DT TPDU is received which is out of sequence it shall be treated as a protocol error); then

2. reassembling (see 3.1.3).

5.2.3.2 Flow control

The transport entities shall send an initial credit (which may be 0) in the CDT field of the CR or CC TPDU. This credit represents the initial value of the upper window edge allocated to the peer entity.

The transport entity that receives the CR or the CC TPDU shall consider its lower window edge as zero, and its upper window edge as the value of the CDT field in the received TPDU.

In order to authorize the transmission of DT TPDUs, by its peer, a transport entity may transmit an AK TPDU at any time, subject to the following constraints:

1. the YR-TU-NR field shall be one greater (modulo 128) than the TPDU-NR field of the last received DT TPDU or must be zero if no DT TPDU has been received;

2. the YR-TU-NR field shall not be lower than that in the previously sent AK TPDU, or lower than zero modulo 128 if no AK TPDUs have been sent;

3. the sum of the YR-TU-NR and CDT fields shall not be lower than the upper window edge allocated to the remote entity. This means that credit reduction is not applicable.

A transport entity which receives an AK TPDU shall consider the YR-TU-NR field as its new lower window edge, and the sum of YR-TU-NR and CDT as its new upper window edge. If either of these have been reduced or if the lower window edge has become greater than the TPDU-NR of the last transmitted DT TPDU, this shall be treated as a protocol error (see 3.1.18).

A transport entity shall not send a DT TPDU with a TPDU-NR outside the transmit window.

This means that a transport entity is required to stop sending if the TPDU-NR field of the next DT TPDU which would be sent would be the upper window edge. Sending of DT TPDU may be resumed if an AK TPDU is received which increases the upper window edge.

The rate at which a transport entity progresses the upper window edge allocated to its peer entity constrains the throughput on the transport connection.

5.2.3.3 Expedited data

ED and EA TPDUs are not subject to the flow control procedures in section 5.2.3.2. The ED-TPDU-NR and YR-TU-NR fields of ED and EA TPDUs respectively are not significant and may take any value.

5.2.4 Release

The transport entities shall use the release procedure in section 3.1.7.

6.0 SPECIFICATION FOR CLASS 3: ERROR RECOVERY AND MULTIPLEXING CLASS

6.1 FUNCTIONS OF CLASS 3

This class provides the functionality of Class 2 (with use of explicit flow control) plus the ability to recover after a failure signalled by the Network Layer without involving the user of the transport service.

6.2 PROCEDURES FOR CLASS 3

6.2.1 Procedures applicable at all times

The transport entities shall use the following procedures:

1. association of TPDUs with transport connections (see 3.1.9);
2. TPDU transfer (see 3.1.2) and retention until acknowledgement of TPDUs (see 3.1.13);
3. treatment of protocol errors (see 3.1.18);
4. concatenation and separation (see 3.1.4);
5. reassignment after failure (see 3.1.8), together with resynchronization (see 3.1.14);

additionally, the transport entities may use the following procedures:

1. multiplexing (see 3.1.15).
2. frozen references (see 3.1.17).

6.2.2 Connection Establishment

The transport entity shall use the following procedures:

1. assignment to network connections (see 3.1.1); then
2. connection establishment (see 3.1.5) together with connection refusal (see 3.1.6).

6.2.3 Data Transfer

The sending transport entity shall use the following procedures:

1. segmenting (see 3.1.3); then

2. DT TPDU numbering (see 3.1.10); after receipt of an RJ TPDU (see 3.1.14) the next DT TPDU to be sent may have a value which is not the previous value of TPDU-NR plus one.

The receiving transport entity shall use the following procedures:

1. DT TPDU numbering (see 3.1.10); the TPDU-NR field of each received DT TPDU shall be treated as a protocol error if it exceeds the greatest such value received in a previous DT TPDU by more than one;

2. reassembling (see 3.1.3); duplicated TPDUs shall be eliminated before reassembling is performed.

6.2.3.1 Use of RJ TPDU

A transport entity may send an RJ TPDU at any time in order to invite retransmission. OSIRIDE transport entity will never use RJ to reduce credit even if it can accept a credit reduction.

When an RJ TPDU is sent, the following constraints shall be respected:

1. the YR-TU-NR parameter shall be at most one greater than the greatest such value received in a previous DT TPDU, or shall be zero if no DT TPDU has yet been received;

2. the YR-TU-NR parameter shall not be lower than that in the previously sent AK or RJ TPDU or lower than zero if no AK or RJ TPDUs have been sent.

When a transport entity receives such an RJ TPDU :

1. the next DT TPDU to be transmitted, or retransmitted, shall be that for which the value of the TPDU-NR parameter is equal to the value of the YR-TU-NR parameter of the RJ TPDU;

2. the sum of the values of the YR-TU-NR and CDT-parameters of the RJ TPDU becomes the new upper window edge

An RJ TPDU can also be sent as part of the resynchronization (see 3.1.14) and reassignment after failure (see 3.1.14) procedures.

The YR-TU-NR parameter shall be equal to the TPDU-NR parameter of the next expected DT TPDU.

6.2.3.2 Flow control

The procedures shall be as defined in section 5.2.3.2, except that:

1. receipt of a DT TPDU with a TPDU-NR parameter whose value is not, but would have been for a credit reduction, less than the upper window edge allocated to the remote entity, shall not be treated as a protocol error.

2. the fact of sending a RJ TPDU implies that all the AK TPDUs eventually created but not yet sent are discarded.

An AK TPDU with an YR-TU-NR field smaller than a preceding RJ is a protocol error.

6.2.3.3 Expedited Data

The sending transport entity shall not allocate the same ED-TPDU-NR to successive ED TPDUs.

The receiving transport entity shall transmit an EA TPDU with the same sequence number in its ED-TPDU-NR field. If, and only if, this number is different from that of the previously received ED TPDU, it will generate a T-EXPEDITED DATA indication to convey the data to the TS-user.

No other significance is attached to the ED-TPDU-NR field. The values in OSIRIDE transport will be consecutive modulo 128. This procedure ensures that the TS-user does not receive data corresponding to the same ED TPDU more than once.

6.2.4 Release

The transport entities shall use the release procedure described in section 3.1.7.

7.0 STRUCTURE AND ENCODING OF TPDUS

7.1 VALIDITY

Fig. 13 specifies those TPDUs which are valid for each class and the code for each TPDU

	validity within classes		Code
	2	3	
CR Connection Request	X	X	1110XXXX
CC Connection Confirm	X	X	1101XXXX
DR Disconnect Request	X	X	10000000
DC Disconnect Confirm	X	X	11000000
DT Data	X	X	11110000
ED Expedited Data	NF	X	00010000
AK Data Acknowledgement	NF	X	0110XXXX
EA Expedited Data Acknowledgement	NF	X	00100000
RJ Reject		X	0101XXXX
ER TPDU error	X	X	01110000

Fig. 13. - TPDU codes

Key:

XXXX (bits 4-1): used to signal the credit

N.F. Not available when the non explicit flow control option is selected.

7.2 STRUCTURE

All the transport protocol data units (TPDUs) shall contain an integral number of octets. The octets in a TPDU are numbered starting from 1 and increasing in the order they are put into an NSDU. The bits in an octet are numbered from 1 to 8, where bit 1 is the low-ordered bit.

When consecutive octets are used to represent a binary number, the lower octet number has the least significant value.

When the encoding of a TPDU is represented using a diagram in this clause, the following representation is used:

1. octets are shown with the lowest numbered octet to the left, higher numbered octets being further to the right.

2. within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

TPDUs shall contain, in the following order:

1. the header, comprising:

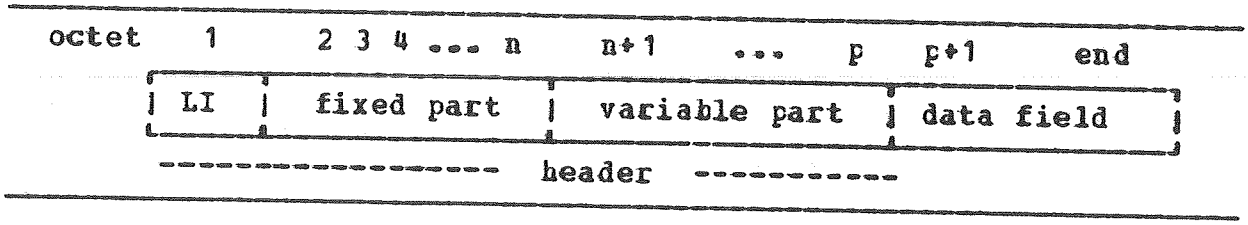
   the length indicator (LI) field;

   the fixed part;

   the variable part, if present;

2. the data field, if present.

This structure is illustrated below:



7.2.1 Length indicator field

This field is contained in the first octet of the TPDUs. The length is indicated by a binary number, with a maximum value of 254 (1111 1110). The length indicated is the header length in octets including parameters, but excluding the length indicator field and user data, if any. The value 255 (1111 1111) is reserved for possible extensions.

7.2.2 Fixed part

General

The fixed part contains frequently occuring parameters including the code of the TPDU. The length and the structure of the fixed part are defined by the TPDU code and in certain cases by the protocol class in use defined by bits 5 to 8 of the second octet of the header.

TPDU code

This field contains the TPDU code and is contained in octet 2 of the header. It is used to define the structure of the remaining header. This field is a full octet except in the following cases:

1110 XXXX	Connecting Request	CR
1101 XXXX	Connection Confirm	CC
0101 XXXX	Reject	RJ
0110 XXXX	Data Acknowledgement	AK

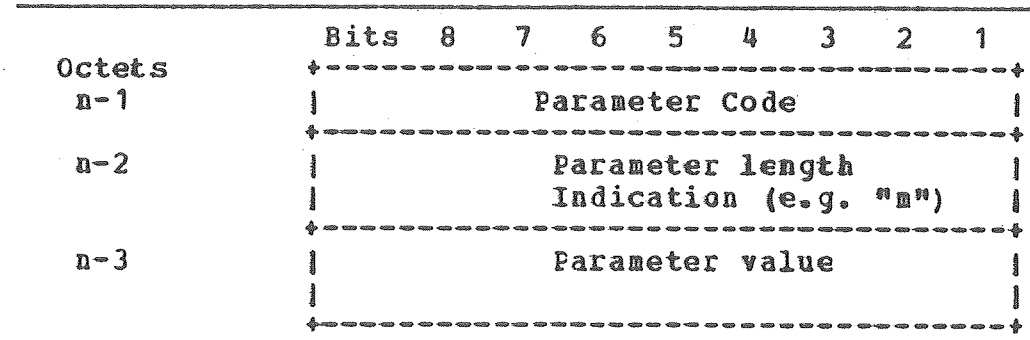
where XXXX (bits 4-1) is used to signal the CDT.

Any other bit pattern may be used to define a TPDU code.

7.2.3 Variable part

The variable part is used to define less frequently used parameters. If the variable part is present, it shall contain one or more parameters. The number of parameters that may be contained in the variable part is indicated by the length of the variable part which is a LI minus the length of the fixed part.

Each parameter contained within the variable part is coded as follows:



OSIRIDE transport layer when inserting parameters in the variable part shall use the order given in the following sections, but it shall accept them in any order when receiving. If any parameter is duplicated the later value will be used.

A parameter not defined in OSIRIDE shall be:

ignored in CR and DR

treated as a protocol error in all other situations

Parameters with bites 8 and 7 equal to 00 are only OSIRIDE internal use.

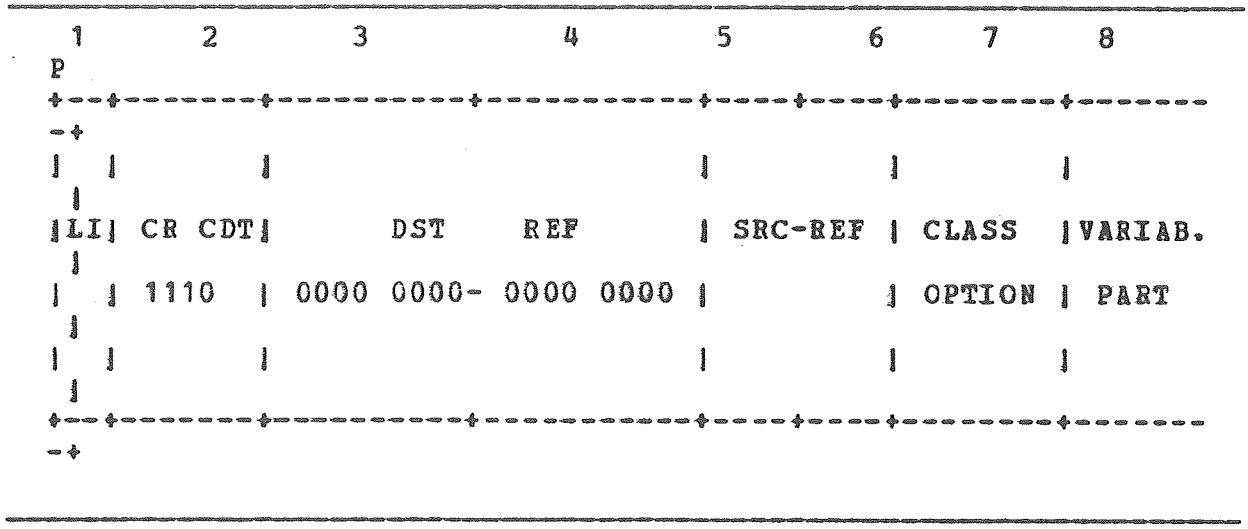
7.2.4 Data field

This field contains transparent user data. Restrictions on its size are noted for each TPDU. It is not used by OSIRIDE in CR, CC, DR; if present in such a received TPDU it will be ignored.

7.3 CONNECTION REQUEST (CR)

The length of the TPDU shall not exceed 128 octets.

Structure



7.3.1 LI

See section 13.2.1.

7.3.2 Fixed Part (Octets 2 to 7)

The structure of this part shall contain:

1. CR: Connection Request Code: 1110. Bits 8-5 of octet 2

2. CDT: Initial Credit Allocation. Bits 4-1 of octet 2.

3. DST-REF: Set to zero

4. SRC-REF: Reference selected by the transport entity initiating the CR TPDU to identify the requested transport connection

5. CLASS and OPTION: Transport Protocol class to be operated over the requested Transport connection. This field may take one of the following values:

   0010 class 2

   0011 class 3

The CR TPDU contains the first choice of class in the fixed part as above. Second choices are not foreseen, only class 2 with flow control as an answer to a class 3 request is allowed. variable part if required.

Bits 4-1 of octect 7 are reserved for option to be used on the requested transport connection.

The use of bits 4-1 is as follows:

BIT		OPTION
4	0	always
3	0	always
2	0	always
1	0	use of explicit flow control in class 2
	1	no use of explicit flow control in class 2

The connection establishment procedure doesn't allow to request the use of transport expedited data transfer service and no use of explicit flow control in class 2.

7.3.3 Variable part (octets 8 to p)

The following parameters are in the variable part; OSIRIDE Transport ahll use all of them when coding the CR TPDU.

1. Transport Service Access Point IDentifier -TSAP-ID- (mandatory)

   Parameter code 11000001 for the identifier of the calling TSAP.
   11000010 for the identifier of the called TSAP.
   Parameter length: two octets
   Parameter value: identifier of the calling or called TSAP respectively

   If a TSAP-ID is given in the request it may be returned in the confirmation.

2. TPDU size (optional)

   This parameter defines the proposed maximum TPDU size (in octets including the header) to be used over the requested transport connection. The ITAPAC (Italian X.25) supports actually only 128. The coding of this parameter is:

   Parameter Code 11000000
   Parameter length: 1 octet
   Parameter value field
   00000111 128 octets (always in OSIRIDE release one)

3. version number (optional)

Parameter code 1100 0100
Parameter length: 1 octet
Parameter value field 0000 0001

Default value is 0000 0001

4. Throughput Parameter code: 10001001 (optional)
Parameter Length : 12
Parmeter Value :
1st 3 octets : Target value, calling-called user direction
2nd 3 octets : Min. acceptable, calling-called user direction
3rd 3 octets : Target value, called-calling user direction
4th 3 octets : Min. acceptable, called-calling user direction
Values are expressed in octets per second.

5. Additional option selection (optional) Used to ask the expedited service

Parameter code : 1100 0110
Parameter Length : 1 octet
Parameter value : 1 use of Transport Expedited
0 no of Transport Expedited
default value : 1

6. Echo Request (optional) Used only in CR TPDU.

Parameter code : 0000 0001
Parameter Length : 1
Parameter value : 0 not requested
1 not requested
default value : 0

This parameter indicates whether this CR TPDU has to used to preforms an eco service (see second part global interface).

7. OSIRIDE like (optional) Used only in CR TPDU.

Parameter code : 0000 0010
Parameter length : 1
Parameter value : 0 I am OSIRIDE
default value : 1

This parameter indicates whether an entity is or not OSIRIDE. It gives significance to OSIRIDE Version number and Echo request parameters.

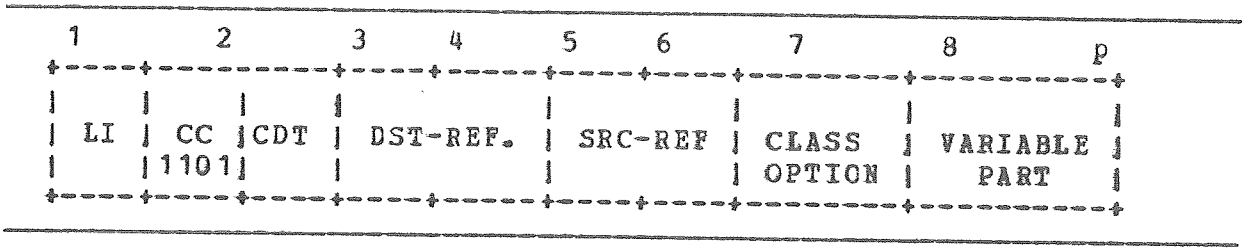
8. OSIRIDE version number (optional) Used only in CR TPDU.

Parameter code : 0000 0011
Parameter length : 1
Parameter value : 0000 0001 for OSIRIDE version 1

This parameter indicates the version of the OSIRIDE software.

7.4 CONNECTION CONFIRM (CC)

7.4.1 Structure



7.4.2 LI

See Section 13.2.1.

7.4.3 Fixed Part (Octets 2 to 7)

The fixed part shall contain:

a) CC : Connection Confirm Code: 1101. Bits 8-5 of octet 2.

b) CDT : Initial Credit Allocation. Bits 4-1 of octet 2.

c) DST-REF: Reference identifying the requested transport connection at the remote transport entity

d) SRC-REF: Reference selected by the transport entity initiating the CC TPDU to identify the confirmed transport connection.

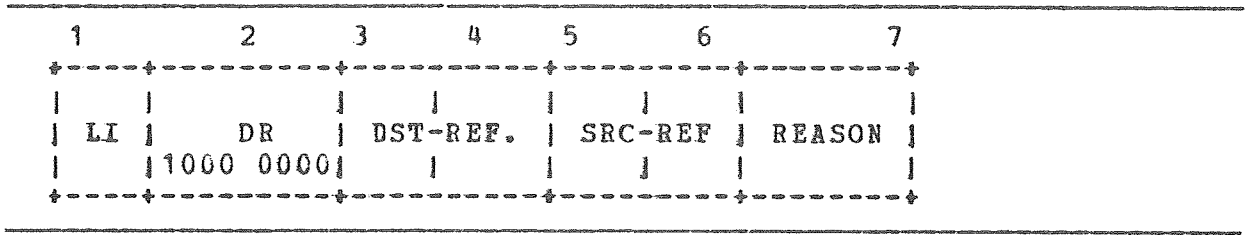
e) CLASS and OPTION: Defines the selected transport protocol class and option to be operated over the accepted transport connection according to the negotiation rules specified in section .....

7.4.4 Variable Part (Octet 8 to p)

The parameters are defined in section 7.2.3 and are subject to the constraints stated in section 3.1.5. Parameters ruled out by selection of an alternative class and option shall not be present.

7.5 DISCONNECT REQUEST (DR)

7.5.1 Structure



7.5.2 LI

See Section 13.2.1.

7.5.3 Fixed Part (Octets 2 to 7)

The fixed part shall contain:

a) DR : Disconnect Request Code: 1000
b) DST-REF: same as in section 7.3.3.
c) SRC-REF: same as in section 7.3.3.
d) REASON : Defines the reason for disconnecting the transport connection. This field shall take one of the following values:

The following values may be used:

1) 128 + 0 - Normal disconnect initiated by session entity
2) 128 + 1 - Remote transport entity congestion at connect request time
3) 128 + 2 - Connection negotiation failed (i.e. proposed class(es) not supported or parameter not agreed)

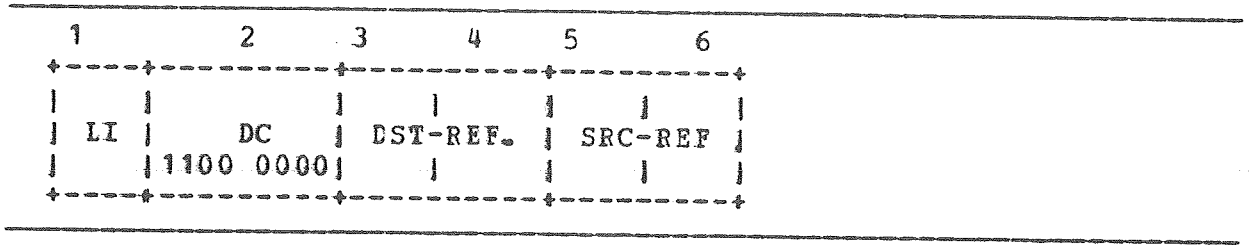
4) 128 + 3 - Duplicate connection detected
5) 128 + 4 - Mismatched references
6) 128 + 5 - Protocol error
7) 128 + 6 - Destination Session not available
8) 128 + 7 - Reference overflow
9) 128 + 8 - Connection request refused on this network connection
10) 128 + 9 - Not used
11) 128 + 10- Header or parameter length invalid
12) 128 + 11- Session protocol error
13) 128 + 12- Echo answer
14) 0 - Reason not specified
15) 1 - Congestion at TSAP
16) *2 - Session entity not attached to TSAP
17) *3 - Address unknown

Reason marked with an asterisk (*) may be reported to the TS-user as persistent, other reasons as transient.

7.6 DISCONNECT CONFIRM (DC)

7.6.1 Structure

The structure of DC TPDU shall be as follows:



7.6.2 LI

See section 13.2.1.

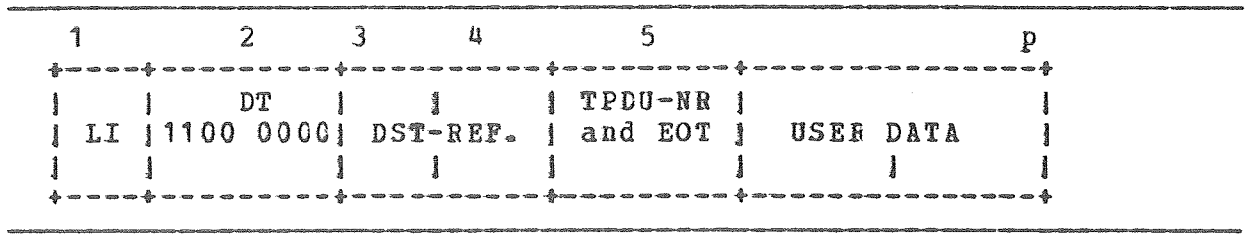
7.6.3 Fixed Part (Octets 2 to 6)

The fixed part shall contain

a) DC : Disconnect Confirm Code: 1100
b) DST-REF: same as in section 7.3.3.
c) SRC-REF: same as in section 7.3.3.

7.7 DATA (DT)

7.7.1 Structure



7.7.2 LI

See section 13.2.1.

7.7.3 Fixed part

The fixed part shall contain:

a) DT : Data Transfer Code: 1111
b) DST-REF: same as in section 7.3.3.
c) EOT : When set to ONE, indicates that the current DT TPDU is the last data unit of a complete DT TPDU sequence (End of TSDU). EOT is bit 8 of octet 5 .
d) TPDU-NR: TPDU send Sequence Number, may take any value in Class 2 without explicit flow control. Bits 7-1 of octet 5.

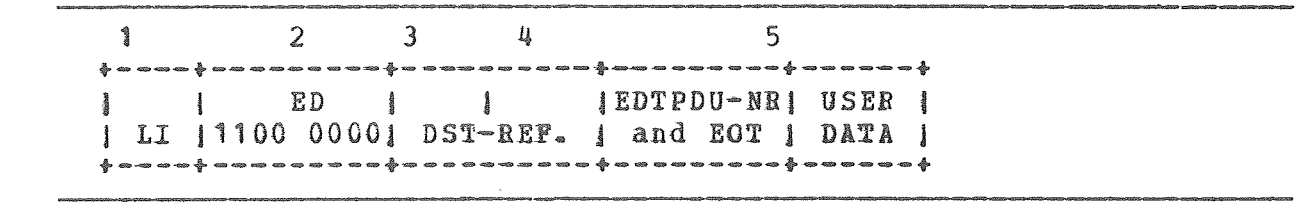
7.7.4 User Data Field

This field contains data of the TSDU being transmitted. The length of this field is limited to the negotiated TPDU size for this transport connection minus 5 octets.

7.8 EXPEDITED DATA (ED)

The ED TPDU shall not be used in class 2 when the no explicit flow control option is selected.

7.8.1 Structure



7.8.2 LI

See section 13.2.1.

7.8.3 Fixed Part

The fixed part shall contain:
a) ED : Expedited Data code: 0001
b) DST-REF : same as in section 7.3.3.
c) ED-TPDU-NR: Expedited TPDU identification number (used in class 3; may take any value in Class 2). Bits 7-1 of octet 5.
d) EOT : end of TPDU always set to 1 (bit 8 of octet 5).

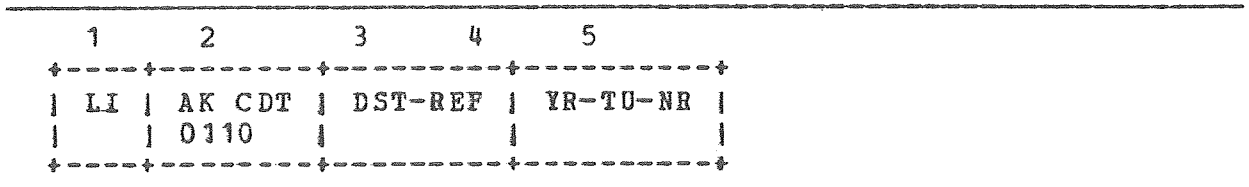
7.8.4 User data field

This field contains an expedited TSDU (1 to 16 octets).

7.9 DATA ACKNOWLEDGEMENT (AK)

This TPDU shall not be used in class 2 when the no explicit flow control option is selected.

7.9.1 Structure



7.9.2 LI

See section 13.2.1.

7.9.3 Fixed Part

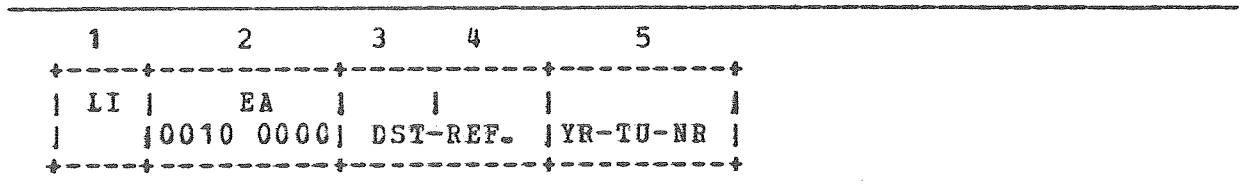
The fixed part shall contain (in octet 2 to 5 the following parameters:

a) AK : Acknowledgement code: 0110
b) CDT : Credit Value. Bits 4 to 1 of octet 2.
c) DST-REF : Same as in Section 7.3.3.
d) YR-TU-NR : Sequence number indicating the next expected DT TPDU number.

7.10 EXPEDITED DATA ACKNOWLEDGEMENT (EA)

This TPDU shall not be used for Class 2 when the no explicit flow control option is selected.

7.10.1 Structure



7.10.2 LI

See section 13.2.1.

7.10.3 Fixed Part

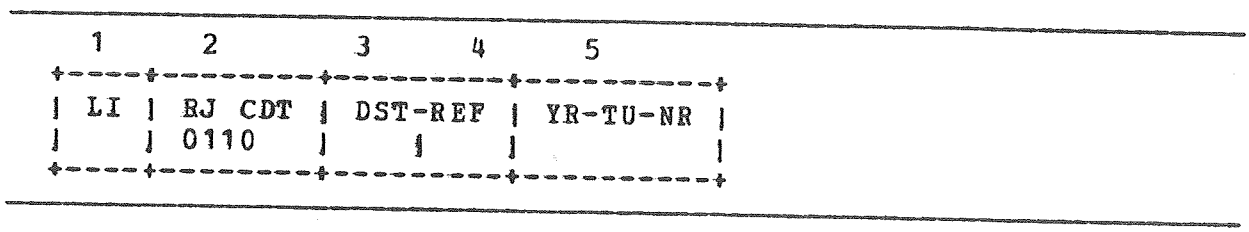
The fixed part shall contain (in octets 2 to 5):

a) EA : Acknowledgement code: 0010
b) DST-REF : Same as in Section 7.3.3.
c) YR-TU-NR : Identification of the ED TPDU being acknowledged. May take may value in Class 2.

7.11 REJECT (RJ)

The RJ TPDU shall not be used in Class 2.

7.11.1 Structure



7.11.2 LI

See section 13.2.1.

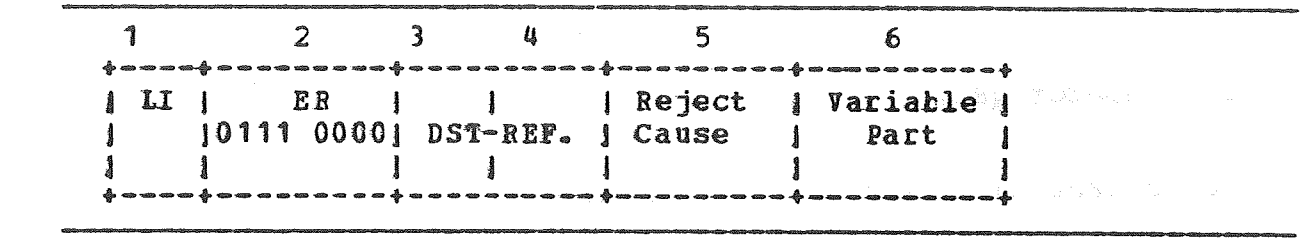
7.11.3 Fixed Part

The fixed part shall contain (in octets 2 to 5)

a) RJ : Reject Code: 0101. Bits 8-5 of octet 2.
b) CDT : Credit Value. Bits 4-1 of octet 2.
c) DST-REF : Same as in Section 7.3.3.
d) YR-TU-NR : Sequence number indicating the next expected TPDU from which retransmission should occur.

7.12 TPDU ERROR (ER)

7.12.1 Structure



7.12.2 LI

See section 13.2.1.

7.12.3 Fixed Part

The fixed part shall contain:

a) ER : TPDU Error Code: 0111
b) DST-REF : Same as in section 7.3.3.
c) REJECT CAUSE : 0000 0000 Reason not specified
0000 0001 Invalid parameter code
0000 0010 Invalid TPDU type
0000 0011 Invalid parameter value

7.12.4 Variable Part (Octets 6 to the end)

The variable part may contain the following parameters:

a) Invalid TPDU
   Parameter code: 1100 0001

Parameter Value Field:

Contains the bit pattern of the rejected TPDU up to and including the octet which caused the rejection.

SECOND PART

1.0 INTRODUCTION

The first part of the document informally describes the transport Layer.

The OSIRIDE project is extremely hetereogeneous; it will enable the linking of differents computers - operating systems and computers from different manifactures - and it will probably be implemented by different teams (i.e. one per computer manifacturer). Therefore the implementation specifications must be extremely detailed, unambiguous and above all cover univocally the entity as a whole - protocol, management and interfaces.
These are the reasons why in this second part we give a more detailed description of the entity, data structure, interfaces and some algorithms.

2.0 INTERNAL LAYER'S STRUCTURE

From a functional point of view the Transport Entity is formed by:

- monitor
- global 'G' machine
- instances of a 'T' FSM
- instances of a 'N' FSM.

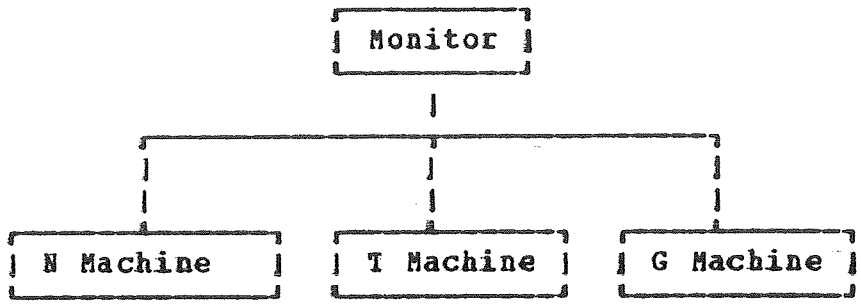


Fig. 14. Transport layer structure

2.1 MONITOR

Monitor Functionalities are:

- initialization and close down of the entity
- managing communication pipes
- managing common resources:buffers, memory, data blocks etc.
- scheduling of FSM on the basis of input messages
- initialization of transport and network connection data blocks
- linking and unlinking of TCDBS to NCDBS

From an implementation point of view part of these monitor functions may be centralized in a 'Monitor Module' while the other part, particularly those related to initialization of transport and network connection data blocks and to interaction between the FSM can be given in the form of FSM callable monitor primitives.

2.2 G-MACHINE

The global machine manages messages from/to LME. The events it has to manage are:

- data collection messages
- echo request/indication
- alarm messages

It can interact with other FSM, for instance Alarm sending is a procedure invoked by a T or N Machine.

2.3 N-MACHINE

The N instances of N-machine every of which is linked to a NCDB data block manage events related to network connections.

Every instance of N-machine is linked to the X instances of T-machine related to transport connections multiplexed over that network connection. The N-machine functionalities are the management of events related to network connections:

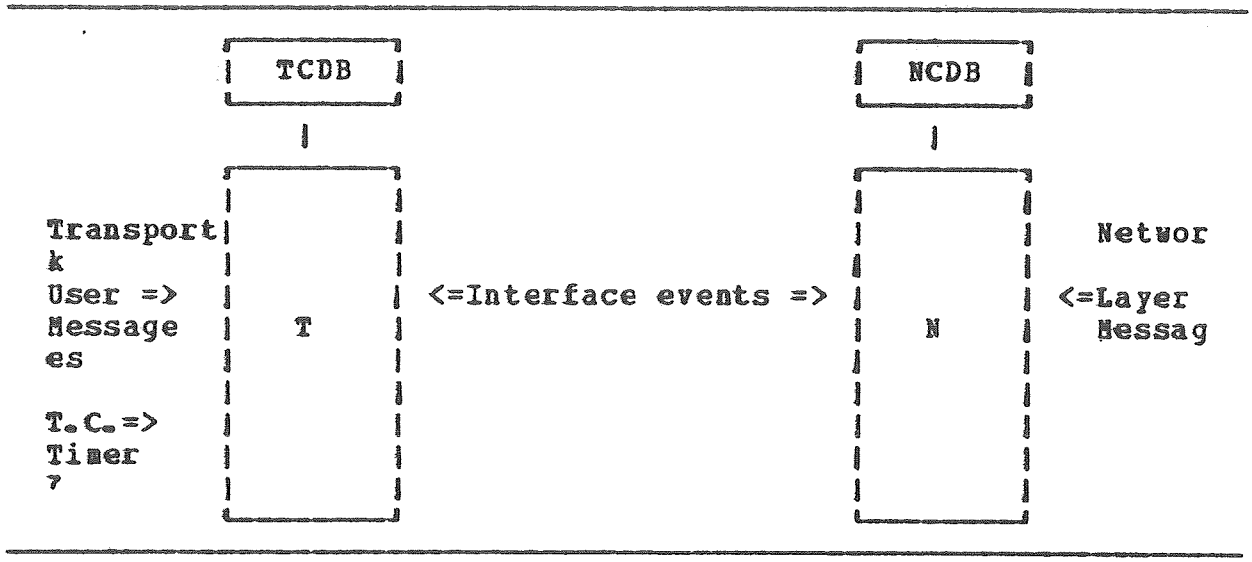
- events from/to network layer
- events from/to T-machine

2.4 T-MACHINE

The M (M>=N) instances of T-machine, everyone of which is linked to a TCDB data block manage events related to transport connection. Every instance of T-machine is linked to the instance of N-machine over which it is multiplexed. The T-machine functionalities are the management of events related to transport connections:

- events from/to transport user
- time out events
- end to end events (interfacing to peer entity)
- events from/to N-machine

The N-machine and T-machine functionalities are summarized in the following figure.



7 Supervisory timer on DR TPDU (TS2), supervisory timer on CR TPDU (TS1) or Time to wait for reassignment (TWR).

3.0 ACTIVATION AND DEACTIVATION OF TRANSPORT ENTITY

The procedure for activating/deactivating OSIRIDE communication software will be described in LMS Specifications; the aim of this paragraph is to describe the transport interface with LMS related to activation/deactivation and the action taken by transport entity when sending/receiving these messages/commands.

Initialization-OK message

From transport to LME. Transport when activated supposes that all global areas have already been initialized by LMS.

It initializes its own areas, sends this indication and waits for the "work" message to enter the maintenance phase.

Work command

From LMS to transport. Transport enters its maintenance phase.

Refuse new connections command

From LMS to transport

After having received this command transport will refuse all T connect request, N-connect request and answer DR to all incoming CRs.

EXIT command

From LME to transport; when receiving this command transport disconnects all eventually open connections and deactivates itself.

The command should be used only after the "refuse new connections" and after having verified that no more connections are active (using the "Data collection command" for instance or an explicit indication from transport) so realizing a soft close-down.

Anyway, it can be used also for a hard closedown.

4.0 INTERFACE LOGICAL STRUCTURE

The transport entity structure has been thought as divided into several functional machines for the purpose of managing the local interface protocols as well as the end-to-end transport protocol.

Thus the management of the layer4/layer3 interface protocol (with regard to the network connection establishment maintenance and clearing) is performed by the N-machine whilst the management of the layer4/layer 5 interface protocol and the management of the end-to-end transport protocol are performed by the T-machine.

Most of the commands and responses supported by the interface protocols map one to one with the network and transport service primitives. Moreover other local functions like local interface flow control and delivering of empty buffer credits to the transport layer are supported by the interface.

An enable transmission/enable reception mechanism is provided at layer 4/layer 5 interface to apply flow control between transport and transport user. Similarly an enable transmission/enable reception is also provided at layer 4/layer 3 interface.

Class 2 with Flow control and class 3 transport connections will use a complete flow control based on interactions between local layer 4/layer 5 and end-to-end protocol flow control mechanisms.

Class 2 without Flow control will use the network flow control and will map such flow control with the backpressure mechanism.

4.1 LAYER 4/LAYER 5 INTERFACE.

4.1.1 Layer 4/layer 5 establishment and clearing commands

This set of commands regards the establishment and clearing phases of single transport connections which can be dynamically established across the transport layer/transport user interface.

More than one transport connection can be established between the same pair of transport addresses. The means by which a session entity can distinguish the transport-connection-endpoints are provided at the time of connection establishment by the session entity itself in terms of TCEP-identifier. This identifier has only local significance and will be mapped one to one with the reference number provided by the transport layer. A "your reference-my reference" mechanism will be used at this layer interface.

The establishment phase is locally initiated by a session entity requesting a transport connection by means of a T-CONNECT request command. This command, if accepted by the transport layer, will cause the transmission of a CR TPDU onto a previously opened network connection. This network connection could be an already existing one on which multiplexing is still possible or a new network connection just opened for this circumstance.

On reception and acceptance of the incoming CR TPDU the remote transport entity will notify the called session entity by means of a T-CONNECT indication.

The called session entity may either accept this command by passing a T-CONNECT response across the interface or refuse the connection by passing a T-DISCONNECT request. If a T-CONNECT response is received the transport entity will form a CC TPDU and will send it to the initiator transport entity. Otherwise, if a T-DISCONNECT request is received, a DR TPDU is formed and sent to the initiator transport entity.

On reception of the CC TPDU the local transport entity will send a T-CONNECT confirm to the calling session entity and the transport connection establishment phase is thus completed.

The clearing phase of a transport connection may be initiated during the data phase either by session entity issuing a T-disconnect -request or by the local transport entity transferring a T-disconnect -indication across the interface.

The termination phase of the transport connection does not guarantee the delivery of data that precedes the transport connection clearing and not yet acknowledged.

4.1.2 Layer 4/ Layer 5 interface flow control

A session entity sends data to transport delivering a buffer that contains the TSDU. If transport entity has credits it sends it to the remote entity (eventually using segmenting function), otherwise it waits for credits. As long as the whole TSDU is not transmitted, the transport entity does not release the buffer; this means that the session is disabled to transmission. The release of the buffer is achieved by the enable transmission command.

The buffer will be structured in this way:

- pointer to next buffer ("nil" if last)
- buffer length
- data

At the receiving side, Session enables transport to reception delivering a buffer (where reassembling function will be

performed) by means of an enable reception command. Transport will give credits to his peer entity on the base of the buffer length.

Receiving TPDU's, the transport entity will reassemble the TSDU, delivering it after receiving the end of TSDU without releasing the residual free space of the buffer. In this remaining area, the transport entity will perform a new reassembly function of the subsequent incoming TSDU. If residual free space is enough to contain a new TSDU, transport will deliver it to his user, otherwise it will wait for new memory to issue other credits and so on.

Initially the CDT value will be calculated as (session buffer length divided by the TPDU length). If this credit is not sufficient for the completion of a TSDU, other credits will be sent on the basis of residual area dimension; if this is less than one TPDU length than credit will be sent one by one until completion of TSDU. If the buffer length is not capable of containing a complete TSDU, it would mean a session protocol error and the transport connection should be closed with appropriate reason code.

Conversely, if the first TSDU were shorter than the buffer length, other N credits could then be issued if the remaining free area is ≥ (N*(TPDU length)).

Initially, T-connect-request and T-connect-response primitives will give transport entity a sufficient buffer area to receive connection protocol messages.

This buffer management is also able to identify the segmenting/reassembling mechanism.

4.1.3 Acknowledgement mechanisms

It is recommended to adopt, when possible, also the following acknowledgement mechanisms. We send AK in the following cases:

1. for class 2 (with flow control) and class 3 transport connections, every time it is possible to concatenate an AK-TPDU (5 octets) in the residual free space of a NSDU.

2. for class 3 transport connections after reception of more than N DT TPDU without sending an AK (N is computed during the connection establishment).

3. when an End of TSDU (EoT) is received.

   Note: In each of these cases the receiving window will not be changed.

4. when the receiving window increases.

4.1.4 Establishment and clearing commands

The commands described in this section are:

T-CONNECT request

T-CONNECT confirm

T-CONNECT indication

T-CONNECT response

T-DISCONNECT request

T-DISCONNECT indication

4.1.4.1 T-CONNECT request

The transport user request to establish a new transport connection. In the message carrying the request, the transport user must specify the local and remote transport addresses, the TCEP-identifer of the new transport connection and the quality of service parameters.

The transport user shall grant a memory area to issue an initial permission to receive. It means a better use of resources while utilizing the CDT field of CR and CC TPDU's.

When cost-saving is requested the cheapest connection has to be selected.

By means of the remote transport address, the transport entity shall be able to find the corresponding network address in order to provide a network connection with the remote transport entity.

parameters:

- TCEP-ID
- Local-transport-address
- Remote-transport-address
- QoS
  - throughput (I->R)
  - throughput (R->I)
  - resilience
- Expedited data option
- cost-saving

- memory area for reception (Pointer, length)

4.1.4.2 T-CONNECT confirm

The transport layer positive response to a previous T-CONNECT request. This command will inform the transport user that the requested transport connection has been established. In the message carrying the response the transport layer will specify the local TCEP-ID of the T.C. (as previously assigned by the transport user by the T-CONNECT request command), the reference number (as assigned by the transport layer after reception of the T-CONNECT request) and the negotiated values of the Q.o.S parameters and options.

parameters:

- Tcep-id
- Reference-number
- QoS
  - throughput (I->R)
  - throughput (R->I)
  - resilience
- Expedited data option

4.1.4.3 T-CONNECT indication

The transport layer indication to the transport user notifying an incoming transport connection request resulting from the reception of a CR-TPDU.

In the message carrying the command the transport layer will specify the reference number of the new transport connection, the transport address of the remote transport user, the QoS parameters as resulting from the CR-TPDU and the kind of network connection which will support the transport connection.

paramters:

- reference number
- remote transport address
- QoS
  - throughput (I->R)
  - throughput (R->I)

resilience

- Expedited data option

4.1.4.4 T-CONNECT response

The acknowledgement response from the transport user to accept an incoming transport connection request. In the message carrying the response, the transport user must specify the Tcep-identifier (assigned after reception of the T-CONNECT indication), the reference number of the established transport connection (as assigned by the transport layer after reception of the CR-TPDU and carried by the T-CONNECT indication), its local transport address, the accepted values for QoS, options and a memory area to grant initial reception of protocol connection messages.

parameters:

1. Tcep-identifier
2. Reference-number
3. Local-transport address
4. QoS
   - throughput (I->R)
   - throughput (R->I)
   - resilience
5. Expedited data option
6. memory area for reception (pointer, length)

4.1.4.5 T-DISCONNECT request

The transport user request to refuse or to clear a transport connection. In the message carrying the request the transport user will specify the Tcep-identifier if assigned (there is no Tcep-identifier if the T-DISCONNECT request is a response to a T-CONNECT indication) and the reference number of the involved transport connection.

parameter:

- Tcep-id
- Reference-number
- Reason

4.1.4.6 T-DISCONNECT indication

The transport layer indication to the transport user of the refusal or clear of a transport connection. In the message carrying the indication the transport layer will specify the tcep-identifier, and the reference number (if assigned, see 3.1.9) of the involved transport connection.

parameters:

- Tcep-id
- Reference-number
- Reason code [8]

4.1.5 Layer 4/layer 5 data transfer commands

This set of commands regard the flow of data which crosses the layer 4/layer 5 interface during the data transfer phase of a single transport connection.

In particular primitives are provided for transmission and reception of normal TSDU (T-DATA request, T-DATA indication), for transmission and reception of expedited TSDU (T-EX-DATA request, T-EX-DATA-indication) and for performing interface flow control (enable transmission, enable reception).

4.1.6 Data transfer commands

The commands described in this section are:

- T-data request
- T-data indication
- TSDU-transmitted
- T-EX-data request
- T-EX-data indication

---

[8] Reason code of disconnect-request or net failure

4.1.6.1 T-DATA request

The transport user requests to send data. Data to be sent will be a TSDU contained in a memory area provided by the transport user.

In the message carrying the request and pointing to the memory area, the transport user will specify the Tcep identifier, the reference number of the involved transport connection, and the pointer to the memory area containing the TSDU.

parameters:

- Tcep-id
- Ref-number
- Tsdu (pointer, length)

4.1.6.2 T-DATA indication

The delivery of one TSDU to the transport user. In the message carrying the indication and pointing to the TSDU the transport layer will specify the Tcep identifier and the reference number of the involved transport connection and pointer and the length of the TSDU.

parameters:

- Tcep-id
- Ref-number
- TSDU (pointer, length)

4.1.6.3 TSDU-transmitted (enable transmission)

With this command the transport layer will inform the transport user that a complete TSDU has been sent and will return to the user the memory area where the TSDU was.

parameters:

- Tcep-id
- Ref-number
- pointer to memory area, length

4.1.6.4 Enable reception

This command enables transport layer to receive from his peer entity and gives it the memory area where (eventually) the TSDU

is reassembled.

parameters:

- Tcep-id
- Ref-number
- pointer to memory area, length

4.1.6.5 T-EX-DATA request

The transport user requests to send an expedited TSDU. In the message carrying the request the transport user must specify the Tcep-identifier, the reference number and the expedited data (max size = 16 octets).

parameters:

- Tcep-id
- Ref-number
- Expedited-data

4.1.6.6 T-EX-DATA indication

The delivery of an expedited TSDU to the transport user. In the message carrying the indication the transport layer will specify the Tcep-identifier, the reference number and the expedited TSDU (max size =16 octects).

parameters:

- Tcep-id
- Ref-number
- Expedited-data

4.2 LAYER 4/LAYER 3 INTERFACE

4.2.1 Layer 4/layer 3 establishment and clearing command set

This set of commands regards the establishment and clearing phases of a single network connection which can be dynamically established across the transport/network layers interface.

The local unambiguous identification of the network connection across the transport/network layers interface is obtained by a "your reference-my reference" mechanism, other than at session/transport layers interface and end-to-end [9].

The establishment phase is locally initiated by a transport entity requesting a network connection to a remote entity by means of a N-CONNECT request command sent to the network layer. This command, if accepted, will cause the network entity to start an attempt to open a new network connection.

If this attempt ends successfully, this fact will be acknowledged to the calling transport entity by means of a N-CONNECT confirm which ends the establishment phase. On the contrary a failed attempt will be signalled by a N-DISCONNECT indication.

Remotely the called transport entity will be notified of the incoming connection request by means of a N-CONNECT indication. The called transport entity may either accept (passing a N-CONNECT response across the interface) or refuse this connection (passing a N-DISCONNECT request across the interface). After sending a N-CONNECT indication the called entity is in data transfer phase.

The clearing phase may be initiated in any phase (establishment or data phases) either by the transport entity (sending a N-DISCONNECT request across the interface) or by the network entity issuing a N-DISCONNECT indication).

Command described in this section:

- N-CONNECT request
- N-CONNECT confirm
- N-CONNECT indication
- N-CONNECT response

---

9 A "my reference/your reference" mechanism is neessary in order to identify a N-connection by a non-owner transport entity before the assignment of the NCEP-ID. and to improve global efficiency.

- N-DISCONNECT request
- N-DISCONNECT indication

4.2.1.1 N-CONNECT request

The transport layer request to the network layer to establish a new network connection

Parameters:

- ncep-id
- called-network-address (remote DTE address)
- QoS
  - throughput (I->R)
  - throughput (R->I)

4.2.1.2 N-CONNECT confirm

The positive response from the network layer to a previous N-CONNECT request.

Parameters:

- ncep-id
- net-ref
- called-network-address (remote DTE address)
- QoS
  - throughput (I->R)
  - throughput (R->I)

4.2.1.3 N-CONNECT indication

The network layer notifies to the remote transport entity an incoming network connection indication.

Parameters:

- calling network address
- net-ref
- QoS

throughput (I->R)

throughput (R->I)

4.2.1.4 N-CONNECT response

Positive ack to a previous N-CONNECT indication

Parameters:

- ncep-id
- net-ref
- QoS

  throughput (I->R)

  throughput (R->I)

4.2.1.5 N-DISCONNECT request

The transport layer request to refuse or to clear a network connection

Parameters:

- ncep-id
- net-ref

4.2.1.6 N-DISCONNECT indication

The network layer indicates to the transport layer to refuse or to clear a network connection.

Parameters:

- ncep-id
- net-ref
- originator[10].
- reason code

---

10 Either user request or network provider initiated.

4.2.2 Layer3/layer 4 data transfer command set

This set of command regards the flow of data which crosses the layer 4/layer 3 interface during the data phase of a single network connection. In particular commands are provided for transmitting and receiving NSDU's and for resetting the network connection.

The flow of NSDU's crossing the interface is a synchronous flow. This means that every time the transport layer sends a NSDU (by means of a N-DATA req command) to the network layer it cannot send another NSDU before receiving an enable-transmission command. Similarly the network layer cannot deliver another NSDU to the transport layer before receiving an enable-reception command from the transport layer. At starting time both transport layer and netowrk layer are able to send the first NSDU.

Command described in this section:

- N-DATA request
- Enable transmission
- N-DATA indication
- Enable reception
- N-RESET request
- N-RESET confirm
- N-RESET indication
- N-RESET response

4.2.2.1 N-DATA request

The transport layer requests to send data. Data to be sent are contained in a buffer pointed by the message carrying the request. Each buffer should be considered as a single NSDU.

Parameters:

- ncep-id
- net-ref
- data-buffer-pointer

4.2.2.2 Transmission enable

The response from the network layer acknowledging the transmission of the NSDU enables the transmission of a new NSDU from

the transport layer

Parameters:

- Ncep-id
- net-ref

4.2.2.3 N-DATA indication

Network layer delivers a NSDU. Data to be received are contained in a buffer pointed by the message.

Parameters:

- ncep-id
- net-ref
- data-buffer-pointer

4.2.2.4 Enable reception

Transport layer enables network layer to issue a new N-DATA indication

Parameters:

- ncep-id
- net-ref

4.2.2.5 N-RESET request

The transport layer requests to reset a specified network connection

Parameters:

- ncep-id
- net-ref

4.2.2.6 N-RESET indication

Network layer indicates a network connection reset. This indication can result from either a reset generated by the network system or from a N-RESET req issued by the remote transport entity.

Parameters:

- ncep-id
- net-ref

4.2.2.7 N-RESET response

The transport layer acknowledges to a previous N-RESET ind.

Parameters:

- ncep-id
- net-ref

4.2.2.8 N-RESET confirm

The confirm to a previous N-RESET request.

parameters:

- ncep-id
- net-ref

4.3 GLOBAL INTERFACE

The Global interface handles the exchange of servive primitives between the Transport entity and the LMS during maintenance. These service primitives provide the following functions:

- Statistical calculation
- Alarm handling
- Echo

4.3.1 Echo Function

The LMS asks the Global machine to carry out the echo function specifying the remote transport address required. The echo function has the job of supplying the LMS with the value of time needed for the exchange of a couple of messages between the end points of a network connection with the remote specified in the request.

Upon reception of the echo request, the Global machine sets the echo flag in the global information table active as only one echo function can be executed every time. Then the Echo-request proce-

dure is called (analogous to the T-connect request except for the presence of an echo-flag in the TCDB).

The remote user may be:

a) another OSIRIDE transport

b) a non-OSIRIDE transport

In the first case, the arrival of a CR-TPDU with echo parameter will provoke the sending of a DR-TPDU (see "incoming new-CR) procedure). When the initiator receives this TPDU, the instant of arrival will be registered, the difference from the instant of departure calculated, and the Echo-answer sent to the LMS.

In the second case, the remote will, in normal conditions, accept the connection sending a CC-TPDU to the initiator, this latter being related to the echo connection (as evidenced by the TCDB referred to), and echo will be signalled to the LMS and the connection (with a DR-TPDU) will be regularly closed.

4.3.1.1 Echo-request

The LMS asks the Transport Entity to execute the echo function supplying the relative remote transport address.

Parameters: Remote-transport-address

4.3.1.2 Echo-answer

The Transport Entity supplies the LMS with the result of the echo operation.

Paremeters: remote transport address
accepted flag = true/false
reason code
time

If acceptee flag = true the reason code is not significant. If accepted flag = false time is not significant. The possible reason code for accepted flag = false are:

- echo already active
- all those provided for the T-DISC indication

4.3.2 Data collection

The LMS sends an information request (Data collection request) to the Transport entity (global machine). The structure of the

request can be unique except for a parameter which indicates the type of information required:

- global on the tranport entity
- specific for transport connection
- specific for network connection

As the number of possible information is large, mechanisms must be provided for the selection of the information needed. As an alternative, it can also be assumed that every time the transport entity gives all the possible information, and the LMS selects some significant information. With the response (Data Colllection response) the Transport Entity gives all the information required to the LMS.

4.3.2.1 Data collection request

The LMS asks the transport entity for entity or transport/network connection information.

data collection type (entry, TC, NC)

connection identificator.

If type equals :

- entity, the connection identifier has no significance
- TC or NC , a way to identify the connection must be provided by the LMS.

4.3.2.2 Data Collection Answer

The entity supplies the LMS with the information requested.

Parameters:

data collection type (entity, TC, NC)

connection identificator

information

In case of possibilities of information selection, a TLV structure can be assumed for the parameters otherwise a positional value structure can be assumed. The useful information may be the following :

1. information on Transport Entity

A. Bytes transmitted to network

B. Bytes received by network

C. Bytes transmitted to session

D. Bytes received by session

E. number of N-CONNECTION "open" (request satisfied)

F. number of N-CONNECTION not "open" (request not satisfied)

G. number of T-CONNECTION "open" (request satisfied)

H. number of T-CONNECTION not "open" (request not satisfied)

I. N-CONNECTION "open" at present

J. T-CONNECTION "open" at present

K. Number of protocol error

L. Number of reset.indication (Network provider generated)

M. Number of disconnect.indication (Network provider generated)

N. Use of transmission primitives

O. Time of Entity initialization

P. Use of concatenation

Q. Use of multiplexing

2. **Information on transport connection ii**

A. Time of initialization

B. Time of termination

C. Bytes transmitted to network

D. Bytes received by network

E. Bytes transmitted to session

F. Bytes received by session

G. Number of reset.indication (Network provider initiated)

H. Number of disconnect.indication (network Provider initiated)

I. Number of data TPDUs transmitted (DT/ED)

J. Number of control TPDUs transmitted

K. mean size of transmittedDT TPDUs

L. Number of T-DATA request

M. Number of times the window is closed to me

N. Number of times I closed the window

3. **Information on Network Connection**

A. Time of initialization

B. Time of termination

C. Reason of termination (normal, Network provider initiated, reception of a non-decodable NSDU, etc.)

D. Time of arrival of the first Reset indication

E. Number of reset

F. Number of multiplexed TC

4.3.2.3 Data collection indication

The entity supplies the LMS with a series of information about a transport or network connection(see 4.3.2.2):

Parameters:

data-coll-ind-type (NC, TC)

message type (statistic, abnormal)

identificator

information (see data collection request.information)

---

11 This information is passed automatically to the manager every time a transport/network connection is closed through the Data Collection indication.

4.3.3 Alarm signalling

In particular situations of error or of external events impeding the normal running of the entity, an alarm message is sent to the LMS with the situation specified and all the useful information for the diagnosis given. In some cases, even a data collection indication may be sent (message type - abnormal).

4.3.3.1 Alarm indication

The entity sends to the LMS an alarm indication.

Parameters:

Alarm type

information

The following is an example of informations passed with an alarm message (indication).

Alarm caused by:

- an N-DISCONNECT indication (Network provider initiated)
  1. Time when error was detected
  2. error type
  3. routine were error was detected
  4. data block relative to the network connection
  5. other fields (FFS)
- a protocol error
  1. Time when error was detected
  2. Error type
  3. Routine were error was detected
  4. Packet which has generated the protocol error
  5. Information characterizing the type of protocol error (TCDB)
- the reception of a non-decodable NSDU
  1. NSDU
  2. Time when error was detected

3. Error type

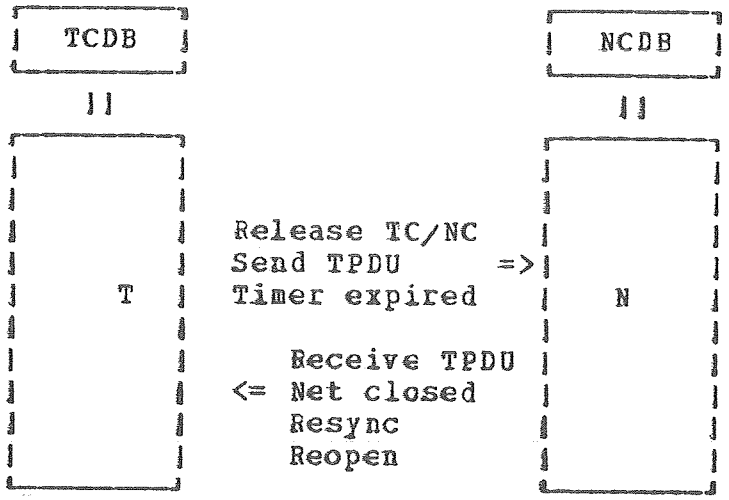
4. Routine were error was detected

5. Information on the Network Connection (FFS)

- a wrong event from Session
- buffer shortage
- Timers started
- other

4.4 INTERNAL INTERFACE

The N-machine/T-machine interface shall be as described in the following figure.



The evidenced events are in effect T machine routines called by N machine and viceversa.

4.4.1 Description of interface mechanisms between the two machines

4.4.1.1 Release TC/NC

T machine tells N machine to break the link between a particular TCDB and a NCDB; if it is possible N-machine will close the N.C.

4.4.1.2 Send XX TPDU

N-machine is informed that there is a TPDU to be sent: in particular for CR, CC, AK, EA, RJ, DR the related flag is set in the TCDB. The TPDU will be built using information contained in the TCDB when creating the NSDU to be sent. This has some welcome side effects: for instance the number of AK may be reduced and the AK, RJ will always be updated to the latest received values, no extra memory is required for retained TPDUs.

For ED TPDUs the ED is created and linked to an ED anchor whose length indicates if there are EDs to be sent, for DT TPDUs a'TSDU to be sent' flag is set.

For DC, ER, DR in response to CR the TPDU is created and linked to garbage queue.

4.4.1.3 Receive TPDU (TPDU)

N-machine checks an N-data-indication for validity. If valid, it performs separation functions and passes incoming TPDUs to related T.C. notifying this event.

Inside the routine implementing this event there will be a a scheduler splitting the event into incoming XX-TPDU events (XX: CC, DR, ...).

4.4.1.4 Net closed

All situations where N.C. is closed and reassignment is not possible, also after N-reset-indication for class 2 T.C.

4.4.1.5 Resync

Only for class 3 T.C.

All situations where it is necessary to issue RJ TPDU without performing reassignment. In particular after N-reset-indication or after network provider initiated disconnect on the owner side, N-machine will notify this event to the T-machine.
This event will reset CR, CC, AK, NMAK, EA, RJ and DR flags.

4.4.1.6 Reopen

Only for class 3 T.C

All situations where it is necessary to perform reassignment. In particular after a network provider initiated disconnect on the non owner side, N-machine will notify T-machine this event. This event will reset CR, CC, AK, NMAK, EA, RJ and DR flags.

4.4.1.7 Timer expired

T-machine informs N-machine that a timer has expired (TS1, TS2). N-machine will issue net-closed to all multiplexed TCs and closes the NC informing the LMS.

5.0 FINITE STATE MACHINES

Conventions used

The FSM are represented as two-dimension matrices in which the conventions are as follows:

Horizontal axis The horizontal dimension is a set of all the relevant states. If for any given state there is no valid event, then this state does (may) not appear in the table.

Vertical axis The vertical dimension of each table is the set of all the relevant events. For each event there is an entry, i.e. a row.

Intersection Each valid intersection contains:

- one or more conditions (where relevant)
- one or more actions (where relevant)
- the new state or state transition (if absent means same state)

Some intersection are exploited in detailed description and the intersection contains the appropriate reference.
An invalid event-state combination is shown by an empty intersection; these situations will be treated as protocol errors only if caused by a peer generated event and anyway signalled to LMS.
Null action is indicated by NOP. An action equal to an adjacent intersection is indicated by i.e. and an arrow pointing to that intersection.

The aim of the high level description is not to limit implementation but to give a detailed descrition of the transport layer.

5.1 LOWER LAYER INTERFACE STATE MACHINE

Transmission and reception are implicitly enabled after N-CONNECT response,confirm and N-RESET response, confirm.

Free TCDB or NCDB means release such data block and release reference (NCEP-ID or TC-REF).

N-MACHINE (1)

Event/State	Wait N-connect-confirm Col. 1	Data Transfer Col. 2	Wait N-reset-confirm Col. 3 Only class 3	Reopening (only owner class 3) Col. 4
Send TPDU row 1		Perform transmit	NOP	NOP
Release link TC/NC row 2		Perform 5.1.1	Perform 5.1.1	
N-connect confirm row 3	Perform 5.1.2			Set transmission flag=true; Perform transmit; --> Data transfer after reopen
N-data indication; row 4		Perform 5.1.3		
N-reset indication row 5		Perform 5.1.4	Collision: set transmission flag=true; Perform transmit; --> Data Transfer	-
N-enable transmission row 6		Perform 5.1.11		
N-reset confirm row 7			Set transmission flag=true; Perform transmit; --> Data Transfer	
N-disconnect indication row 8	Perform 5.1.5	Perform 5.1.6	Perform 5.1.6	Perform 5.1.5
Timeout row 9		Perform 5.1.9		see note
Closing time-out; row 149				
Receive data row 150		Perform 5.1.12		

N-MACHINE (2)

Event/State	Data transfer after reopen (only owner class 3); col. 5	Wait N-reset-confirm after reopen (only owner class 3);col.6	Closing col.150
Send TPDU row 1	Perform transmit;	NOP	
Release link TC/NC; row 2			
N-connect confirm row 3			
N-data indication; row 4	Perform 5.1.7		
N-reset indication row 5	Perform 5.1.8	Collision: set transmission flag=true; Perform transmit; --> data transfer after reopen	N-Disconnect request; free NCDB --> Idle
N-enable transmission row 6	Set transmission flag=true; Perform transmit		
N-reset confirm row 7		Set transmission flag=true; Perform transmit; --> Data Transfer after reopen	NOP
N-disconnect indication row 8	Perform 5.1.10	Perform 5.1.10	free NCDB --> Idle
Timeout row 9	see note	see note	
Closing Timeout row 149			N-DISC request;free NCDB -->Idle
Receive data row 150			

Note:
This event never appears because TS1 and TS2 are much greater then TTR.

5.1.1 ROW2COLUMN2

```
PROCEDURE ROW2CCLUMN2
BEGIN
  Unlink TC from NC;
  IF (number of active transport connection = 0) and
     (garbage queue empty) and (owner = true)
    THEN
  BEGIN
    Start closing timer;
    NC-status:= Closing;
  END;
END;
```

5.1.2 ROW3COLUMN1

```
PROCEDURE ROW3COLUMN1
BEGIN
   NC-status:=Data transfer;
   Position at the beginning of scanning
   REPEAT
     signal NC-assigned to TC;
   UNTIL end of scanning of the multiplexed TC
   set transmission flag;
   perform transmit;
END (*of PROCEDURE ROW3COLUMN1 *)
```

Because of multiplexing, there is a series of serialized events which cannot be separated logically, for instance, signalling the T-machine of the completed assignment of network connection. Since every T-machine receiving the signal of connection is, from that moment onwards, capable of sending to the N-machine a SEND-TPDU, it is necessary to switch to in data transfer state before beginning assignment loop in order to avoid a possible state-event incongruence (arrival of send TPDU in wait state for N-CONN confirm).

During loop of assignment, the transmission is disabled as the initialization procedure of NCDB must set the transmission flag to "false". Once the loop of assignment is finished the N-machine is abled to transmission and the respective procedure is called.

5.1.3 ROW4COLUMN2

```
PROCEDURE ROW4COLUMN2
BEGIN
   REPEAT
     separate TPDU; (* performs separation *)
     IF TPDU not decoded THEN
     BEGIN
      position at the beginning of scanning of multiplexed TCDB;
      REPEAT
         IF class = 2 THEN
            net closed to TC;
         ELSE
            resync to TC;
      UNTIL end of scanning
      IF (no class 3 T.C. and OWNER) THEN
      BEGIN
        N-DISC request;
        NC-status:=idle;
        free NCDB;
      END
      ELSE
      BEGIN
        N-reset-request;
        NC=status:= wait for N-reset-confirm;
      END;
   END
   ELSE
     IF owner = true THEN
     BEGIN
        position at the beginning of scanning of multiplexed TCDBs;
        REPEAT
          IF TCDB is found THEN
             Send incoming TPDU
          ELSE
             position on the next TCDB
        UNTIL TCDB is found or end of scanning
        IF not found THEN
          SPURIUS TPDU;
       (* see first part 3.1.9 *)
     END
     ELSE
     BEGIN
        position at the beginning of scanning of multiplexed TCDBs;
        REPEAT
          IF TCDB found THEN
             send incoming TPDU;
          ELSE
             position on the next TCDB
        UNTIL TCDB is found or end of scanning
        IF TCDB not found THEN
        BEGIN
          IF CR TPDU THEN
          BEGIN
            Verify-CR (Kind) (* see 6.2 *)
```

```
            IF new CR THEN
              Incoming new CR (CR-TPDU);
            ELSE
              Retransmitted CR (CR-TPDU);
          END
          ELSE
          BEGIN
            Perform reassignment(TPDU,success);
            IF success THEN
              send incoming XX-TPDU;
            else
              spurius TPDU;
          END;
        END;
      END;
    UNTIL (end of TSDU or TPDU not decoded).
    IF End of NSDU THEN
      IF (a DT-TPDU has been received on a class <> 2NFC TC) THEN
        Enable reception 4-3
      ELSE (* Backpressure*)
        NOP;
END;(*of PROCEDURE ROW4COLUMN2 *)
```

The pending transport connections are provided with TCDB but do not possess a link to a NCDB and are identifiable by

- state

- presence in a special list -reopening anchor (see Data structure)

The reassignment procedure tests this situation according to the DST-ref of the TPDU received but not assigned to any TCDB for the lack of linkage. If the result is positive, the procedure sets the TC/NC link and resets the timer TWR relative to the TC concerned.

This procedure applies to all TPDUs with the exception of the CR TPDUs for which (as these are dst-ref = 0 by definition) a testing is necessary using the source-ref and the remote transport address. This test is done for every CR-TPDU as it is not possible to discover in advance if the connection is new or it is a repeated CR because of a failure (see Verify-CR).

Because of multiplexing, there is a series of serialized events which cannot be separated logically, for instance, signalling the T-machine of the completed assignment of network connection. Since every T-machine receiving the signal of connection is, from that moment onwards, capable of sending to the N-machine a SEND-TPDU, it is necessary to switch to in data transfer state before beginning assignment loop in order to avoid a possible state-event incongruence (arrival of send TPDU in wait state for N-CONN confirm).

During loop of assignment, the transmission is disabled as the initialization procedure of NCDB must set the transmission flag

to "false". Once the loop of assignment is finished the N-machine is abled to transmission and the respective procedure is called.

5.1.4 ROW5COLUMN2

```
PROCEDURE ROW5COLUMN2
BEGIN
  IF number of class 3 transport connection <> 0 THEN
  BEGIN
     position at the beginning of scanning of multiplexed TCDBs;
     REPEAT
        IF class = 2 THEN
           net closed to TC;
        ELSE
           resync to TC;
     UNTIL end of scanning of multiplexed TCDBs
     N-reset-response;
  END
  ELSE
  BEGIN
     position at the beginning of scanning of multiplexedTCDBs;
     REPEAT
        net-closed to TC;
     UNTIL end of scanning
     IF owner = true THEN
     BEGIN
        N-disconnect-request;
        NC-status:= idle;
        free NCDB;
     END
     ELSE
         N-reset-response
   END;
END;(*of PROCEDURE ROW5COLUMN2 *)
```

Because of multiplexing, there is a series of serialized events which cannot be separated logically, for instance, signalling the T-machine of the completed assignment of network connection. Since every T-machine receiving the signal of connection is, from that moment onwards, capable of sending to the N-machine a SEND-TPDU, it is necessary to switch to in data transfer state before beginning assignment loop in order to avoid a possible state-event incongruence (arrival of send TPDU in wait state for N-CONN confirm).

During loop of assignment, the transmission is disabled as the initialization procedure of NCDB must set the transmission flag to "false". Once the loop of assignment is finished the N-machine is abled to transmission and the respective procedure is called.

5.1.5 ROW8COLUMN1

```
PROCEDURE ROW8COLUMN1
BEGIN
   counter:=counter + 1;
   IF counter < max. value THEN
   BEGIN
      N-connect-request;
   END
   ELSE
   BEGIN
      position at the beginning of scanning;
      REPEAT
         signal Net-closed(reason=NC not assigned);
      UNTIL end of scanning of multiplexed TCDBs;
      NC-status:= idle;
      free NCDB;
   END;
END;(*of PROCEDURE ROW8COLUMN1 *)
```

---

12 The maximum value of the counter of the attempts to open is a configuration parameter.

5.1.6 ROW8COLUMN2

```
PROCEDURE ROW8COLUMN2 BEGIN
   IF network provided THEN
   BEGIN
      IF number of class 3 TC <> 0 THEN
      BEGIN
         IF owner = true THEN
         BEGIN
            position at  the beginning of scanning  of multiplex-
edTCDBs;
            REPEAT
               IF class = 2 THEN
                  net closed to TC;
               ELSE
                  resync to TC;
            UNTIL end of scanning
            N-connect-request;
            counter:=counter + 1;
            NC-status:=reopening;
         END
         ELSE (* owner = false *);
         BEGIN
            positioned at the beginning of scanning of multiplex-
edTCDBs;
            REPEAT
               IF class = 2 THEN
                  net closed to TC;
               ELSE
                  Reopen to TC;
            UNTIL end of scanning
            NC-status:= idle;
            free NCDB;
         END
         ELSE (*no class 3 TC *)
         BEGIN
            position at the beginning  of scanning of multiplexed
TCDBs;
            REPEAT
               Net closed to TC;
            UNTIL end of scanning
            NC-status:= idle;
            free NCDB;
         END
         ELSE
         BEGIN
            NC-status:= idle;
            free NCDB;
         END; END;(*of PROCEDURE ROW8COLUMN2 *)
```

5.1.7 ROW4COLUMN5

```
PROCEDURE ROW4COLUMN5
BEGIN
   REPEAT
      separate TPDU; (* perform separation *)
      IF TPDU not decoded then
      BEGIN
         position at the beginning of scanning of multiplexed TCDBs;
         REPEAT
            resync to TC;
         UNTIL end of scanning of multiplexed TCDBs;
         N-reset-request;
         NC-status:= wait for N-reset-confirm after reopen;
      END
      ELSE (* TPDU not decode *)
      BEGIN
         position at the beginning of scanning of multiplexedTCDBs;
         REPEAT
           IF TCDB found THEN
           BEGIN
              reset counter;(* it is the counter used in place of TTR *)
              send incoming XX-TPDU;
            END
            ELSE
              position on the next TCDB;
         UNTIL TCDB found or end of scanning;
         IF TCDB not found THEN
           Spurius TPDU;
      END;
   UNTIL end of NSDU or TPDU not decoded;
   IF (End of NSDU ) and (no spurius TPDU) THEN
   BEGIN
     NC-status:= data transfer;
     Enable reception;
   END;
END;(*of PROCEDURE ROW4COLUMN5 *)
```

5.1.8 ROW5COLUMN5

```
PROCEDURE ROW5COLUMN5
BEGIN
   position at the beginning of scanning of multiplexed TCDBs;
   REPEAT
      resync to TC;
   UNTIL end of scanning;
   N-reset-response;
END;(*of PROCEDURE ROW5COLUMN5 *)
```

5.1.9 ROW9COLUMN2

```
PROCEDURE ROW9COLUMN2
BEGIN
   positioned at the beginning scanning TC multiplexed
   REPEAT
      net closed to TC
   UNTIL end of scanning
   N-DISC request;
   informs LMS of the timeout signal;
(*this is a warning signal *)
   free NCDB;
END;(*of PROCEDURE ROW9COLUMN2 *)
```

5.1.10 ROW8COLUMN6

```
PROCEDURE ROW8COLUMN6
BEGIN
   counter:=counter + 1;
   IF counter < max. value THEN
   BEGIN
      Resync to all TCs;
      N-connect-request;
      NC-status:= reopening
   END
   ELSE
   BEGIN
      position at the beginning of scanning;
      REPEAT
         signal Net-closed(reason=NC not assigned);
      UNTIL end of scanning of multiplexed TCDBs;
      NC-status:= idle;
      free NCDB;
   END;
END;(*of PROCEDURE ROW9COLUMN2 *)
```

5.1.11 ROW6COLUMN2

```
PROCEDURE ROW6COLUMN2
BEGIN
  IF (number of active TC =0) and (owner = true)
      and (G.Q empty) THEN
  BEGIN
     Start closing timer;
     NC-status:=Closing;
  END ELSE
  BEGIN
     Transmit-flag=true;
     Perform transmit;
  END
END;(*of PROCEDURE ROW6CCLUMN2 *)
```

5.1.12 ROW150COLUMN2

```
PROCEDURE ROW150COLUMN2
BEGIN
  IF in baskpressure 4-3 THEN
     Enable Reception 4-3;
(* This procedure is used in class 2 NFC and is called after a Session/
   Transport backpressure when the Session gives reception buffers ***)
END;(*of PROCEDURE ROW150COLUMN2 *)
```

5.2 UPPER LAYER INTERFACE STATE MACHINE AND TRANSPORT PROTOCOL MACHINE (T-MACHINE)

T-Machine LOCAL OPEN (initiator side)

Event/State	Wait for assignment of NC col. 7	Wait for CC col. 8
NC-assigned row 10	send CR-TPDU; start TS1; --> Wait for CC	
Incoming CC row 12		Perform 5.2.1
Incoming DR row 13		Perform 5.2.2
Resync row 14		send CR-TPDU
Unknown TPDU row 15		Perform 5.2.3
TS1 Timer row 16		Timeout
Incoming ER row 17		Peform 5.2.2
Net closed row 18	T-DISC indication (*) free TCDB --->idle	Perform 5.2.2

(*) Reason will be n.c. not assigned.

T-Machine REMOTE OPEN (Responder Side) (1) the second part is in the next page

Event/State	Wait T-connect-response col. 9	Wait T-connect-response then send T-discon.-indication col. 10	Wait CR after Resync (only class 3) col. 11	Wait CR after Resync & after T-connect-response (only class 3) col. 12	Wait CR after Resync & after T-disconn. request (only class 3) col. 13
T-connect Response row 19	Parameter calculation; send CC TPDU; --> Wait first TPDU	T-disconnect-indication Free TCDB; -> Idle	--> Wait CR after resync & after T-connect-response		
Incoming CR row 20			-->wait T-connect-response	Parameter calculat. send CC TPDU --> Wait first TPDU	send DR TPDU with SRC-REF=0 on g.g.; release link TC/NC; free TCDB;--> Idle
T-disconnect request row 21	send DR TPDU (SRC-REF =0) on g.g.; release link TC/NC; free TCDB; --> Idle	free TCDB --> Idle	-->wait CR after resync & after T-disc. request	-->Wait CR after resync & after T-discon.-request	
Net closed row 22	release link TC/NC; -->Wait T-connect-resp. then send T-discon-ind.		release link TC/NC; -->wait T-con.-resp. then send T-discon. indication (Note 6)	T-discon.-ind.; release link TC/NC; free TCDB; Note 6 --> Idle	release link TC/NC; free TCDB; Note 6 --> Idle
Resync row 23	--> wait CR after resyn		NOP Note 6	NOP Note 6	NOP Note 6
Unknown TPDU row 24	send ER TPDU on g.g.; release link TC/NC; --> Wait T-con.-respon. then send T-discon.-ind		send ER TPDU on g.g release link TC/NC; free TCDB; Note 6 T-discon-indication --> Idle	<--- i.e.	send ER TPDU on g.g release link TC/NC; free TCDB; Note 6 --> Idle
Reopen row 25	unlink TC;link TC to reopen anchor; start TWR; -->Wait CR after resync		unlink TC; link TC to reopen anchor; Note 1,6	unlink TC; link TC to reopen anchor; Note 1,6	unlink TC; link TC to reopen anchor Note 1,6
TWR Timer row 26			unlink TC from reopen anchor; inform LMS; -->wait T-con. response then send T-discon.-indication	unlinkTC from reopen anchor; T-disconnect indica.; free TCDB inform LMS --> Idle	unlink TC from reopen anchor; free TCDB; inform LMS ---> Idle

Event/State	Wait first TPDU (Note 5) col. 14	Wait CR,RJ,DR col. 15 (only class 3)	Closing after CR,RJ,DR col. 16 (only class 3)
Incoming CR row 27		send CC-TPDU --> wait first TPDU	send DR-TPDU on g.g. (source ref=0); release link TC/NC; free TCDB; --> Idle
Incoming DR row 28	T-disconnect-indication; SEND DC-TPDU on g.g.; free TCDB release link TC/NC;-->Idle	<-- i.e.	send DC-TPDU on g.g.; release link TC/NC; free TCDB; --> Idle
Incoming DT row 29	Perform 5.2.9 and then (note 3) -->Data Transfer		
Incoming ED row 30	T-EX-data-indication; (note 4); send EA-TPDU; Inform LMS;--> Data Transfer		
Incoming AK row 31	Update TX-CDT --> Data Transfer (note 4)		
Incoming RJ row 32	Update TX-CDT; (note 7) --> Data Transfer	Update TX-CDT; Send RJ-TPDU;-->Data Transfer	Send DR-TPDU; start TS2; --> wait DC
Incoming ER row 33	T-disconnect indication; release link TC/NC free TCDB; --> Idle	<-- i.e.	release link TC/NC; free TCDB; --> Idle
Net closed row 34	T-disconnect indication; free TCDB; --> Idle	<-- i.e. Note 6	free TCDB --> Idle Note 6
Resync row 35	--> wait CR,RJ,DR	NOP Note 6	NOP Note 6
T-disconnect request row 36	send DR-TPDU; start TS2; --> wait DC	--> closing after RJ, DR, CR	
Unknown TPDU row 37	send ER-TPDU on g.g. release link TC/NC; free TCDB; T-disconnect-ind.; --> Idle	<-- i.e.	Send ER-TPDU on g.g.; release link TC/NC; free TCDB; --> Idle
Reopen row 38	Start TWR; unlink TC link TC to reopen anchor --> wait CR, RJ, DR	Start TWR; unlink TC link TC to reopen anchor Note 6	start TWR; unlink TC; link TC to reopen anchor Nota 6
TWR Timer row 39		unlink TC from reopen anchor; T-disconnect-indication; free TCDB; inform LMS; --> Idle	unlink TC from reopen anchor; free TCDB; inform LMS --> Idle

T-Machine DATA TRANSFER (1)

Event/State	Data Transfer col. 17	(Note 2) Data Tranfer T.D. (trans.disabled) col. 18	(Note 11) Data Transfer WEA (wait for Expedi. ack) col 19	Wait RJ (class 3 only) col. 20	Wait RJ after reopen (class 3 responder's side only) col. 21	Wait RJ after T-di request(class 3 on responder's side) row 22
T-data-req. row 40	set TSDU to be sent flag; send DT-TPDU (Note 8)	set TSDU to be sent flag;	<-- i.e.	<-- i.e.	<-- i.e.	
T-EX-data-req. row 41	perform 5.2.10	Perform 5.2.10	queue ED-TPDU in EX queue	queue ED-TPDU in EX queue	queue ED-TPDU in EX-queue	
Incoming AK row 42	Perform 5.2.11	Perform 5.2.8	Perform 5.2.11			
Incoming EA row 43	Perform 5.2.6	Perform 5.2.6	Perform 5.2.4			
T-discon.-request row 44	send DR-TPDU; start TS2; --> wait DC	<--- i.e.	<-- i.e.	<-- i.e.	--> wait RJ after T-disc.-request	
Enable reception 54 row 45	Perform 5.2.12	<-- i.e.	<-- i.e.	<-- i.e.	<-- i.e.	
Net closed row 46	T-discon.-ind. free TCDB; --> Idle	<-- i.e.	<-- i.e.	<-- i.e.		
Resync row 47	send RJ-TPDU --> wait RJ	<-- i.e.	<-- i.e.	send RJ-TPDU		
Incoming RJ row 48	Perform 5.2.5	Perform 5.2.5	Update TSDU current pointer; update TX-CDT Note 7	Perform 5.2.7	Send RJ-TPDU and then perform 5.2.7	Send DR-TPDU; star TS2 --> wait D

T-Machine DATA TRANSFER (2)

Event/State	Data Transfer col. 23	Data Transfer T.D. col. 24	Data Transfer W. EA col. 25	Wait RJ col. 26	Wait RJ after reopen col. 27	Wait RJ after T-disc.-request col. 28
Incoming DR row 49	T-disconnect-indication; release link TC/NC; free TCDB; send DC TPDU on g.q.: --> Idle	<-- i.e.	<-- i.e.	<-- i.e.	<-- i.e.	release link TC/NC; free TCDB send DC-TPDU on g.q. --> Idle
Incoming DT row 50	Perform 5.2.9	<-- i.e.	<-- i.e.			
Incoming ED row 51	T-ex-data-ind send EA TPDU	<--- i.e.	<-- i.e.			
Incoming ER row 52	T-discon-ind. release link TC/NC; free TCDB; inform LMS; --> Idle	<-- i.e.	<-- i.e.	<-- i.e.	<-- i.e.	Release link TC/NC; free TCDB inform LMS; --> Idle
Unknown TPDU row 53	send ER TPDU on g.q.; release link TC/NC; free TCDB; T-disc. ind.; inform LMS --> Idle	<-- i.e.	<-- i.e.	<-- i.e.	<-- i.e.	send ER TPDU on g.q; release link TC/NC; free TCDB inform LMS; --> Idle
Reopen row 54	Start TWR; unlink TC; link TC to reopen anchor; -->Wait RJ after reopen	<-- i.e.	<-- i.e.	<-- i.e.		
TWR Timer row 55					T-disconnect ind.; release link TC/NC; free TCDB; inform LMS --> Idle	release link TC/NC; free TCDB; inform LMS; --> Idle

T-Machine CLOSING

Event/State	Wait DC Col. 29
Net closed row 56	Free TCDB ----> Idle
Resync row 57	send DR TPDU
Incoming DR row 58	(Collision) Release link TC/NC; free TCDB; ----> Idle
Incoming DT/ED/AK/EA/RJ row 59	NOP
Incoming ER row 60	Release link TC/NC; free TCDB; inform LMS; ----> Idle
Unknown TPDU row 61	send ER TPDU (on g.q.); release link TC/NC; free TCDB, inform LMS; ----> Idle
Incoming DC row 62	Release link TC/NC; free TCDB; ----> Idle
TS2 Timer row 63	Timeout
Reopen row 64	Start TWR; unlink TC/NC; Link TC to reopen anchor; --> Wait RJ after T-disconnect request
TWR Timer row 65	Unlink TC from reopen anchor; Free TCDB; inform LMS; ----> Idle

5.2.1 ROW12COLUMN8

```
PROCEDURE ROW12COLUMN8
BEGIN
   perform parameter test;
   IF parameter test is OK THEN
   BEGIN
      reset timer TS1;
      T-connect-confirm;
      TC-status:= data transfer;
   END
   ELSE (* parameter test  not OK *)
   BEGIN
      T-disconnect-indication;
      send DR TPDU;
      reset TS1;
      start TS2;
      TC-status:= wait DC;
   END;
END (* of PROCEDURE ROW12COLUMN8*);
```

5.2.2 ROW13COLUMN8

```
PROCEDURE ROW13COLUMN8
BEGIN
   T-disconnect-indication;
   Release link TC/NC;
   Reset timer TS1;
   TC-status:= idle;
   Free TCDB;
END (* of PROCEDURE ROW13COLUMN8*);
```

5.2.3 ROW15COLUMN8

```
PROCEDURE ROW15COLUMN8
BEGIN
   send ER-TPDU on garbage queue;
   reset timer TS1;
   T-disconnect-indication;
   Release link TC/NC;
   Inform LMS;
   TC-status:= idle;
   Free TCDB;
END (* of PROCEDURE ROW15COLUMN8 *);
```

5.2.4 ROW43COLUMN19

```
PROCEDURE ROW43COLUMN19
BEGIN
   IF class 2 THEN
   BEGIN
      send ED-TPDU;
      IF no other ED-TPDU in ED-QUEUE THEN
      BEGIN
         IF window closed THEN
            TC-status:= Data Transfer T.D.;
         ELSE
            IF TSDU to be sent flag THEN
            BEGIN
               send DT-TPDU;
               IF End of TSDU THEN
                  Enable transmission (4-5);
            END
            TC-status:=Data Transfer;
      END
   END
   ELSE
   BEGIN
      IF EA-TPDU has the number of the first TPDU in ED-QUEUE THEN
      BEGIN
         release that TPDU from EX-QUEUE;
         send ED-TPDU;
         IF no other ED-TPDU in ED-QUEUE THEN
         BEGIN
            IF window closed THEN
               TC-status:= Data Transfer T.D.;
            ELSE
               IF TSDU to be sent flag THEN
               BEGIN
                  send DT-TPDU;
                  TC-status:= data transfer;
               END;
         END;
      END
      ELSE
         protocol error;
   END;
END (* of PROCEDURE ROW43COLUMN19 *);
```

5.2.3 ROW15COLUMN8

```
PROCEDURE ROW15COLUMN8
BEGIN
   send ER-TPDU on garbage queue;
   reset timer TS1;
   T-disconnect-indication;
   Release link TC/NC;
   Inform LMS;
   TC-status:= idle;
   Free TCDB;
END (* of PROCEDURE ROW15COLUMN8 *);
```

5.2.4 ROW43COLUMN19

```
PROCEDURE ROW43COLUMN19
BEGIN
   IF class 2 THEN
   BEGIN
      send ED-TPDU;
      IF no other ED-TPDU in ED-QUEUE THEN
      BEGIN
         IF window closed THEN
            TC-status:= Data Transfer T.D.;
         ELSE
            IF TSDU to be sent flag THEN
            BEGIN
               send DT-TPDU;
               IF End of TSDU THEN
                  Enable transmission (4-5);
            END
            TC-status:=Data Transfer;
      END
   END
   ELSE
   BEGIN
      IF EA-TPDU has the number of the first TPDU in ED-QUEUE THEN
      BEGIN
         release that TPDU from EX-QUEUE;
         send ED-TPDU;
         IF no other ED-TPDU in ED-QUEUE THEN
         BEGIN
            IF window closed THEN
               TC-status:= Data Transfer T.D.;
            ELSE
               IF TSDU to be sent flag THEN
               BEGIN
                  send DT-TPDU;
                  TC-status:= data transfer;
               END;
         END;
      END
      ELSE
         protocol error;
   END;
END (* of PROCEDURE ROW43COLUMN19 *);
```

5.2.5 ROW48COLUMN17

```
PROCEDURE ROW48COLUMN17
BEGIN
   TSDU current pointer:=YR-TU-NR of RJ-TPDU)
   update TX-CDT;
   IF TX-CDT = 0 THEN
      TC-status:= Data Transfer T.D.;
   ELSE
   BEGIN
      IF TSDU to be sent THEN
      BEGIN
         SEND DT-TPDU;
      END;
      TC-status:=Data Transfer;
   END;
END (* of PROCEDURE ROW48COLUMN17 *);
```

5.2.6 ROW43COLUMN17

```
PROCEDURE ROW43COLUMN17
BEGIN (*see note 10 *)
   IF class 2 THEN
      set EX-flag;
   ELSE (* class 3 *)
   BEGIN
     IF (EA-TPDU number equals the number of the first ED-TPDU
         in EX-QUEUE ) THEN
     BEGIN
        Release that ED-TPDU from the EX-QUEUE;
        set EX-flag;
     END
     ELSE(* mismatched number *)
        Protocol Error;
   END;
END (* of PROCEDURE ROW43COLUMN17 *);
```

5.2.7 ROW48COLUMN20

```
PROCEDURE ROW48CCLUMN20
BEGIN
   set Ex-flag;
   TSDU current pointer:=YR-TU-NR of RJ-TPDU;
   IF EX-QUEUE not empty THEN
   BEGIN
      send ED-TPDU;
      reset EX-flag
      IF other EX-TPDU in EX-QUEUE THEN
         TC-status:= data transfer wait for EA;
      ELSE (*only one ED-TPDU in EX-QUEUE *)
      BEGIN
         IF window closed THEN
            TC-status:= Data Transfer T.D.;
         ELSE
         BEGIN
            IF TSDU to be sent flag THEN
            BEGIN
               send DT-TPDU;
            END;
            TC-status:= Data Transfer;
         END;
      END;
   END
   ELSE
   BEGIN
      IF TSDU to be sent flag THEN
      BEGIN
         send DT-TPDU;
      END;
      TC-status:= Data Transfer;
   END;
END (* of PROCEDURE ROW48COLUMN20 *);
```

5.2.8 ROW42COLUMN18

```
PROCEDURE ROW42COLUMN18
BEGIN
   Update TX-credit;
   IF class 3 THEN
   BEGIN
      Delete DT-TPDU mark confirmed;
      IF the whole TSDU has been confirmed THEN
      BEGIN
         Enable Transmission (4-5);
      END;
   END;
   IF TSDU to be sent flag THEN
   BEGIN
      send TPDU
   END;
   IF Window not closed THEN
      TC-status:=Data Transfer;
   ELSE
      Signal this impossible event to LMS;
(* see note 9 *)
END (* of PROCEDURE ROW42COLUMN18 *);
```

5.2.9 ROW5OCOLUMN23

```
PROCEDURE ROW5OCOLUMN23
BEGIN
   enqueue user-data of DT-TPDU in reassembling buffer;
   IF EOT of DT-TPDU  THEN
   BEGIN
      T-data-indication;
(* the remaning area of receiving buffer is kept by the Transport
   for receiving other TPDUs *)
   END;
   IF class 3 THEN
   BEGIN
      IF ((TPDU-NR of DT-TPDU) - (last RX-TPDU-ACKED))
            greater than N THEN
         (* N is a function of initial configuration *)
        Send AK-TPDU
       ELSE
          send NMAK; (* sets NMAK flag (Not Mandatory AK flag) without
                        issuing SEND TPDU to N-Machine *)
   END
   ELSE
      IF (class <> 2 N.F.) or ((clas = 2 N.F.) and
      (receiving area >= 121 bytes) THEN
        Receive Data (4-3);
END (* of PROCEDURE ROW5OCOLUMN23 *)
```

5.2.10 ROW41COLUMN17

```
PROCEDURE ROW41COLUMN17
BEGIN
   IF EX-flag THEN
   BEGIN
      Send ED-TPDU;
      Reset EX-flag;
   END
   ELSE
   BEGIN
      Enqueue ED-TPDU in EX-QUEUE;
      TC-status:= Data Transfer W.E.A.;
   END;
END (* of PROCEDURE ROW41COLUMN17 *)
```

5.2.11 ROW42COLUMN17

```
PROCEDURE ROW42COLUMN17
BEGIN
   Update TX-credit;
   IF class 3 THEN
   BEGIN
      Delete confirmed DT TPDU mark;
      IF ( the whole TSDU has been confirmed ) THEN
         Enable Transmission (4-5);
   END;(* see note 9 *)
END (* of PROCEDURE ROW42COLUMN17 *)
```

5.2.12 ROW45COLUMN17

```
PROCEDURE ROW45COLUMN17
BEGIN
   IF class 2 N.F. THEN
      Receive Data (4-3);
(* N-machine will ignore the Receive Data if not in backpressure *)
   ELSE
   BEGIN
      Update TX-credit;
      Send AK-TPDU;
   END;
END (* of PROCEDURE ROW42COLUMN17 *)
```

Note

1. Event sequence:

   A. arrival CR, wait for T-connect-response

   B. N-reset-indication to NCDB, resync to TCDB, passage to W.CR state after Resync.

   C. N-disconnect-indication (Net provided) to NCDB, Reopen to TCDB.

   The following two events may take place before c)

   A. T-connect-response

   B. T-disconnect-request

2. The state of disabled transmission is induced by the transmit routine when the transmission window is closed.

3. the reception of DT-TPDU as the first TPDU implies that the session has given an implicit enable reception using the parameters of the T-connect-response (to be mapped in the adequate credit in the CC-TPDU).

4. It is impossibile. In fact, the first data TPDU should be the session connect request PDU which use normal flow. Anyway we want to be open.

5. The "wait first TPDU" state does not foresee the T-data-request event in fact the first data TPDU is sent by the initiator of the transport connection (Session Connect Request).

6. The OSIRIDE initiator doesn't use reassignement procedures in the connection establishment phase. Anyway we want to be open.

7. The state transitions (same state or Data Transfer T.D.) are performed implicitly by Transmit routine.

8. in class 3 DT marks are cancelled after acknoledgment(AK-TPDU)

9. in class 3 ED-TPDU are cancelled after acknoledgment (EA-TPDU).

10. It has been choiced to stop the flow of all DT-TPDUs if there are more than one ED-TPDU in EX-QUEUE. This greatly simplify FSM.

11. Send NMAK does not correspond to a SEND XX-TPDU event to N-machine; it only sets the NMAK flag.

6.0 DATA STRUCTURES

The data structures used at the transport layer are:

- global information relative to the transport layer
- local information relative to the particular entity
- configuration files
- NCDB (Network Connection Data Block)
- TCDB (Transport Connection Data Block)
- Messages in input/output (see interface description 4/3, 5/4, global)
- network buffers
- statistic counters (see global interface description)

It is necessary to provide a data structure capable of finding all data blocks (TCDB, NCDB) for the management of global events (close down etc.).

6.1 GLOBAL INFORMATION AND LOCAL INFORMATION

- max. number of transport connections
- max. number of network connections [13]
- value of timers
- TPDU size (128 octets)
- reopening anchor
- refuse new connection flag
- Itapac resilience (refer to network provider initiated Reset-indication or Disconnect-indication).

[13] Maximum number of logical circuits supported by ITAPAC = 4096.

6.2 CONFIGURATION FILES

Table of conversion from transport addresses (to the knowledge of transport users) to Itapac addresses

6.3 NETWORK CONNECTION DATA BLOCK (NCDB)

Data block representing the interactions with the network layer, referred by NCEP-id. There is one block for every network connection and n TCDBs are linked to each block, one TCDB for every multiplexed transport connection on the network.

The NCDB is a record type containing the fcllowing fields:

1. owner : boolean
2. NC-status : list of possible stati
3. max-multiplexable-TCs : integer
4. number of active TCs : integer
5. number of class 3 TCs : integer
6. net-ref : integer
7. ncep-id : integer
8. multiplexed-TCs-anchor : Top,Tail pointers and length
9. garbage-anchor : Top,Tail pointers and length
10. X.25 ITAPAC addr.(remote): array of bytes
11. TTR counter : integer
12. Q.o.S. of network connection
    - throughput(I->R) : List of values
    - throughput(R->I) : List of values
13. transmission flag : boolean (3-4 backpressure)
14. pointer begin scanning : pointer (includes the G.Q.)
15. reception flag : boolean (4-3 backpressure)

6.4 TRANSPORT CONNECTION DATA BLOCK (TCDB)

Data block representing the protocol mechanism and interactions with the T.U..There is one block for every Transport Connection. It is referred by TC-reference.

It is a record type containing the following fields:

1. initiator flag : boolean
2. TC-status : list of states
3. protocol class : list of possible classes (2FC,2NF,3)
4. expedited option : boolean
5. local Tcep-id : integer
6. local TC-reference : integer (16 bits)
7. remote TC-reference : integer "
8. local T-address : array of bytes
9. remote T-address : array of bytes

For reception:

10. RX-cdt : integer (credit in reception)
11. next-RX-TPDU-NR : integer (next expected data TPDU)
12. last-RX-EX-data : integer (last expedited will be checked in class 3
13. last RX-TPDU-NR-NOT ACKED: integer (last TPDU for which AK has been sent)
14. max-DT-not-acked : integer (max. receivable DTs before sending AK)

In transmission:

15. Last Tx-ex-data : last expedited created
16. Ex-flag : boolean (able flag to EX-data sending)
17. TSDU top pointer : pointer at beginning of data buffer received by session
18. TSDU current pointer: pointer at first data byte to be sent

6.2 CONFIGURATION FILES

Table of conversion from transport addresses (to the knowledge of transport users) to Itapac addresses

6.3 NETWORK CONNECTION DATA BLOCK (NCDB)

Data block representing the interactions with the network layer, referred by NCEP-id. There is one block for every network connection and n TCDBs are linked to each block, one TCDB for every multiplexed transport connection on the network.

The NCDB is a record type containing the fcllowing fields:

1. owner : boolean
2. NC-status : list of possible stati
3. max-multiplexable-TCs : integer
4. number of active TCs : integer
5. number of class 3 TCs : integer
6. net-ref : integer
7. ncep-id : integer
8. multiplexed-TCs-anchor : Top,Tail pointers and length
9. garbage-anchor : Top,Tail pointers and length
10. X.25 ITAPAC addr.(remote): array of bytes
11. TTR counter : integer
12. Q.o.S. of network connection
    - throughput(I->R) : List of values
    - throughput(R->I) : List of values
13. transmission flag : boolean (3-4 backpressure)
14. pointer begin scanning : pointer (includes the G.Q.)
15. reception flag : boolean (4-3 backpressure)

6.4 TRANSPORT CONNECTION DATA BLOCK (TCDB)

Data block representing the protocol mechanism and interactions with the T.U..There is one block for every Transport Connection. It is referred by TC-reference.

It is a record type containing the following fields:

1. initiator flag : boolean
2. TC-status : list of states
3. protocol class : list of possible classes (2FC,2NF,3)
4. expedited option : boolean
5. local Tcep-id : integer
6. local TC-reference : integer (16 bits)
7. remote TC-reference : integer "
8. local T-address : array of bytes
9. remote T-address : array of bytes

For reception:

10. RX-cdt : integer (credit in reception)
11. next-RX-TPDU-NR : integer (next expected data TPDU)
12. last-RX-EX-data : integer (last expedited will be checked in class 3
13. last RX-TPDU-NR-NOT ACKED: integer (last TPDU for which AK has been sent)
14. max-DT-not-acked : integer (max. receivable DTs before sending AK)

In transmission:

15. Last Tx-ex-data : last expedited created
16. Ex-flag : boolean (able flag to EX-data sending)
17. TSDU top pointer : pointer at beginning of data buffer received by session
18. TSDU current pointer: pointer at first data byte to be sent

19. TSDU residual length: (remaining data length to be sent)

20. TSDU to be sent : flag informing on the presence or absence of data to be sent.

21. Table TSDU-mark-DT : (list of DT-TPDU created and not yet confirmed, in the following form:

- pointer at beginning : pointer
- TPDU-number : integer
- TPDU-length : integer

Note: In class 3 the arrival of an AK-TPDU provokes the deletion from this list of the confirmed TPDU-MARKs.

22. Last-TX-YR-TU-NR : Last progressive used for the DT-TPDU transmission

23. TX-CDT : Credits in transmission. It may be incremented by an AK-TPDU It may be incremented or decremented by an RJ-TPDU

24. EX-anchor : top pointer, tail pointer anchor length

25. CR-flag : boolean

26. CC-flag : boolean

27. AK-flag : boolean

28. EA-flag : boolean

29. RJ-flag : boolean

30. DR-flag : boolean

31. NMAK-flag : boolean

7.0 MAIN PROGRAM AND PROCEDURES DESCRIPTION

7.1 MAIN PROGRAM

The transport entity is formed by:

- an initialization section to manage the start up phase
- a maintenance phase to manage communications events
- a closedown phase to orderly deactivate the entity.

The global structure of the entity shall be of this kind:

Program transport

```
PROCEDURE initialize;
BEGIN
   Initialize global areas;
   Send INITIALIZATION OK to LMS;
END;

PROCEDURE maintenance;
BEGIN
   REPEAT
      Receive(message);
  UNTIL (message = WORK);
   REPEAT
      Receive(message);
      Process(message);
   UNTIL (message=EXIT);
END;

PROCEDURE closedown;
BEGIN
   Orderly close all activities;
   Release resources;
END;

  BEGIN
(* Main program code*)
   Initialize;
   Maintenance;
   Closedown;
END;
```

The software organization of maintenance routine foresees one module to receive input messages and activate appropriate management routines and N routines to manage the single events.

The scheduling module will distinguish the input messages on the basis of which FSM will process them and then activate the related FSM.

The FSM are formed by several managing routines anyone of which manages a single state event combination. Routines shall be described in the FSM pseudocode.

Every managing routine receives a message in input, processes it and returns the control to caller.

These routines will be described in more detail in the FSM paragraph.

So the maintenance module 'process(message)' routine shall be of this kind:

```
PROCEDURE process (message);
BEGIN
   CASE 'message shall be managed by FSM'
        N  : activate N-machine;
        T  : activate T-machine;
        G  : activate G-machine;
   END (*of case *)
END;
```

The structure of the X-machine will be

```
  PROCEDURE X-machine;
BEGIN
   CASE Message of
     XXX  : Routine for managing XXX event;
     YYY  : Routine for managing YYY event;
     ZZZ  : Routine for managing ZZZ event;
     TTT  : Routine for Managing TTT event;
   END(*of case *)
END;
```

The structure of 'Routine for managing XXX event' shall be:

```
PROCEDURE managing-XXX-event;
BEGIN
   CASE state of
        A   : A-XXX;
        B   : B-XXX;
        C   : C-XXX;
   END(*of case*)
END;
```

where A-XXX etc. are described by the FSM.

As an alternative to nested case, if using assembler code, a branch table structure can be used (see Fig. 15 pag. 142).

Event / STATE	XXX	YYY	ZZZ	TTT
A	A-XXX	A-YYY	A-ZZZ	A-TTT
B	B-XXX	B-YYY	B-ZZZ	B-TTT
C	C-XXX	C-YYY	C-ZZZ	C-TTT
D	D-XXX	D-YYY	D-ZZZ	D-TTT

Fig. 15. Branch table

A-XXX etc. are the addresses of managing routines the scheduling can be done by an instruction like

```
JMP TABLE(STATE,EVENT)
```

The preceding considerations mean that the maintenance phase is strictly external event driven: the entity is activated by external events and processes them, in a FIFO order.

If some other external event is necessary to proceed processing the situation of the particular connection is frozen by means of a state and resumed as soon as the required event happens.

This choice has been preferred against a 'circular priority on connection basis' because it doesn't introduce any privilege between events and connections. Every event gets the same priority that's to say that, in normal situations, an event is immediately processed as soon as it happens so giving a very good and fast service: no time is spent in realizing circular scheduling.

On the other hand it's to point out that in unbalanced situations where a connection generates much higher traffic than other this connection will get more service and this could lead to worse global service than using the 'circular priority' in very heavy traffic situations where a great amount of CPU load is devoted to communication.

As a conclusion we can say that the selected solution is simpler to realize and a bit faster under normal traffic, may lead to a worst service under very heavy and unbalanced traffic.

It's also to note that the selected solution doesn't exclude the association of priorities to events: this may be realized sending messages to different priority queues and processing the queues by priority.

In the following we give a Pascal-like description of some procedures, this is only a way to formally describe algorithms and we don't want to limit the implementation; ew only want descrobe what a procedure shall do, not how.

7.2 N-MACHINE PROCEDURES FOR PERFORMING REASSIGNMENT OF T.CS TO N.CS

In this note we'll describe the 'verify-CR' and 'Perform-Reassignment' routines called in the N-data-indication, data-transfer intersection of N-machine.
We suppose that for all incoming CRs 'VERIFY CR' will be called and that for all other TPDUs not related to multiplexed TCDBs the 'PERFORM REASSIGNMENT' will be called.

We briefly recall the management of N-data-indication, data-transfer intersection:

Separation is applied to the received NSDU, if a TPDU can't be decoded then class 2 T.C.S. are closed, resync is sent to class 3 T.C.S., N-reset-request is issued.
If the TPDU is decoded first it is necessary to split management of CRS, that we'll see later from that of other TPDUs. Then if we are the owner of the N.C. these are signalled to related T.Cs as incoming-XX-TPDU; if we are not owner of the N.C. things are a bit more complicated.

In fact first we'll verify if the TPDUs are for multiplexed T.Cs, if they are then we'll signal them as incoming-XX-TPDU otherwise we'll call the 'Perform-Reassignment' routine because they could be TPDUs retransmitted after a network failure that yet have to reassigned. If the exit status of 'perform-reassignment' is success:=true then the TPDUs will be signalled as incoming-XX-TPDU otherwise a DR with SRCREF:=0,Reason:= mismatched references will be sent to peer entity together with an alarm-indication to LMS.

```
PROCEDURE perform-reassignment (TPDU, success);
BEGIN
  IF ((the DST_REF carried by this TPDU exists) and
     (the related TCDB is linked to reopening anchor))
  THEN
  BEGIN
     unlink TCDB from reopening anchor;
     link-TC-to-NC;
     reset TWR;
     (*as TWR is a T-machine event it should be reset
     there: doing it here we can in most situations treat
     reset and reopen in the same way so reducing the
     number of stati of T-machine*)
     success:=true;
  END;
```

```
   ELSE
      success:=false;
END; (*of procedure perform-reassignment*)
```

For CRs it is necessary tc verify if it is a new CR (in this case the monitor procedure 'incoming-new-CR' is called) or if it is a retransmitted CR because of a reset or a reassignment after failure; in this case it will be normally signalled with an 'incoming repeated CR TPDU' to the related instance of T-machine.

It is not possible to use the 'perform-reassignment' because the DST-REF in CR is 0 so we'll use the SRC-REF jointly with remote transport address end eventually T.C. status:

```
PROCEDURE verify-CR(kind);
(*kind will be true if new CR, false if retransmitted CR*)

BEGIN
   Kind:=true;
   (*verify for CR repeated after reset*)
   REPEAT
     Position on next multiplexed T.C.;
     (*first time it's the first one*)
     IF (TPDU SRC-REF=TCDB's remote REF) and
        (TPDU remote transport address=TCDB's remote
         transport address)
     THEN
        kind:=false;(*retransmitted CR*)
   UNTIL (kind=false) or (scanned all multiplexed TCDBs);
   IF kind=true(*is not a repeated CR after reset*) THEN
   BEGIN
      (*Verifies if is a repeated CR after a network
      provider initiated disconnect*)
      REPEAT
         Position on next T.C. of reopening anchor;
         (*first time it's the first one*)
         IF (TPDU SRC-REF=TCDB's remote REF) and
             (TPDU remote transport address=TCDB's remote
             transport address) THEN
         BEGIN
            Kind:=false;(*retransmitted CR*)
            unlink TCDB from reopening anchor;
            link-TC-to-NC;
            Reset TWR;
            (*as TWR is a T-machine event it should be reset
             there: doing it here we can in most situations treat
             reset and reopen in the same way so reducing the
             number of T-Machine's states*)
         END;
         UNTIL (Kind=false) or (scanned all TCDBs in
           reopening anchor);
      END;
END;(*of procedure*)
```

7.3 MONITOR PROCEDURES RELATED TO INITIALIZATION OF TRANSPORT AND NETWORK CONTROL DATA BLOCKS

Aim of this note is to describe those monitor procedures that are related to starting an instance of a T-machine or N-machine. That's to say to initialization of a NCDB or of a TCDB. These procedures are called in four particular situations:

1. When a T-connect-request is issued by T.U. (initialization of a TCDB)

2. When an incoming CR is detected (initialization of a TCDB)

3. When T-machine request to link the TCDB to a N.C. (may be initialization of a NCDB)

4. When a N-connect-indication is issued by N.L. (initialization of a NCDB).

These four situations will be managed by four different procedures. We'll not describe details as 'get-data-block' routines also because they might be implementation dependent. However it's posible to think we have a data block pool from which to get data blocks when needed. These routines imply the assignment of a data block (or better connection) identifier, NCEP-ID or TC-REF; the exit status of them, a boolean var we call success, is function of:

- available memory for TCDBs, NCDBs

- number of active T.Cs, N.Cs

7.3.1 Procedures description

```
Procedure T-connect-request (remote transport address, QoS requested)
(*manages T-CONNECT request issued by T.U. see interface description*)

BEGIN
 IF not(refuse new connection flag) THEN
  IF (Remote transport address exists and is reachable)
  (*this function will access configuration
  tables and check the remote transport address for
  validity, that's to say it must exist and be reachable*)
    THEN
  BEGIN
     Allocate reference(success,ref)
     IF success THEN
     BEGIN
       Get-TCDB(success);
       IF success(*A TCDB has been found*) THEN
       BEGIN
         CHOICE CLASS(QoS params passed by SESSION,class number,options);
```

```
            Initialize-TCDB;
            (*sets all values to default initial values, sets
            reference number*)
            set-initiator;
            T-status:=wait-for-NC-assigned;
            Assign-NC(remote-Transport-address, QoS requests);
            (*ASKS monitor to assign the T.C. to a N.C. that's to
            say to multiplex on an existing one or to open a new
            N.C.*)
          END
          ELSE
            Signal T-disconnect-indication (Reason:=NO TCDB available);
            (*to T.U.*)
        END
        ELSE
          Signal T-disconnect-indication (reason:=no ref available);
                                                  (*TO T.U.*)
    END
    ELSE
      Signal T-disconnect-indication (reason:=remote transport
      Address invalid); (*TO T.U.*)
  ELSE
  Signal T-disconnect indication (reason:=refusing new connections);
END;(*of procedure T-connect-request*)
```

```
PROCEDURE incoming-new-CR(CR-TPDU)
(*called when a new incoming CR is detected i.e. not
valid for retransmitted CRs*)

BEGIN
 Reason:=OK;
 IF not(refuse new connection flag) THEN
  IF (DST-address exists) THEN
  BEGIN
     Test-CR-parameters-for-validity(success):
     IF success(*params are valid*) THEN
     BEGIN
        IF (Echo-request not set) THEN
        BEGIN
           Negotiate (CR-TPDU-Params, output params, success);
           (*this procedure will set params to be passed to T.U.
           in the T-connect-indication basing on
           -remote's requests
           -local resources
           An output boolean VAR success will indicate whether
           negotiation has been performed successfully or not
           see note 14*)
           IF success (*negotiation OK*) THEN
           BEGIN
              alloc ref(success,ref);
              IF success THEN
              BEGIN
                Get-TCDB(Success);
                IF success(*a TCDB has been found*)THEN
                BEGIN
                  Initialize-TCDB;
                  (*sets all values to default initial value, sets
                  reference number*)
                  Link-TC-to-NC;
                  (*cross pointers on the data blocks are set*)
                  Signal T-connect-indication(output params of
                  negotiation);(*To T.U.*)
                  T-Status:=wait-T-connect-response;
                END
                ELSE reason:=ND TCDB available;
              END ELSE
                reason:= no ref available;
           ELSE reason:=negotiation failed;
        ELSE reason:=echo request;
     ELSE reason:=parameters invalid;
  ELSE reason:=destination address not active;
 ELSE reason:=refuse new connections;
END;
IF reason<>OK THEN send-DR-TPDU(SRC:=),Reason);
END;(*of procedure incoming new CR*)
```

---

14 OSIRIDE Transport will always accept the class selection made by the remote, if acceptable (3,2FC,2NFC), and the class 3 will map always in resilience high, class 2 will map in resilience high or low basing on N.C character-

```
PROCEDURE assign-NC(remote transport address, Qos requests);
(*called by T-connect-request procedure*)

  BEGIN
  IF multiplexing-is-possible(remote transport address,QoS requests)
        THEN
(*this boolean function will scan all existing N.C. to
check if it is possible to multiplex on an existing one*)
  BEGIN
    Link-Tc-to-NC;(*cross pointer on the data blocks are set*)
    IF N-status=data transfer THEN
       signal NC-assigned; (*To T-machine*)
  END
  ELSE
  BEGIN (*it wasn't possible to multiplex*)
    allocate ref(success,ref);
    IF success THEN
    BEGIN
      Get-NCDB(success);
      IF success THEN
      BEGIN
        MULTIPLEXING?
(* decides which kind of network connection has to be opened *)
        Initialize-NCDB;
        Set-owner;
        Link-TC-to-NC;
        Signal N-connect-Request; (*to N.L.*)
```

---

istics,EXPEDITED will be set according to option selection. The parameters carried by CR will be managed in the following way:

1. dest TSAP-id will be checked to see if we are that

2. remote TSAP-id may be checked to see if enabled

3. TPDU size in the first version must be 128

4. throughput will map in QoS parameters passed to SESSION in T-connect indication

5. additional option selection used to see if expedited is required or not

6. Version number significant only if I am OSIRIDE is set,version should be the same

7. Echo significant only if I am OSIRIDE is set

8. I am OSIRIDE useful to verify if version number and echo are significant or not.

The same applies to parameters carried by CC, class 2FC may be accepted versus a class 3 proposed, the remote SESSION may refuse an expedited service request.

```
          NC-status:=wait N-connect-confirm;
        END
        ELSE
          Signal NC not assigned (reason = no NCDB available);
      END
      ELSE
        signal NC not assigned (reason = no reference available);
    END
END;  (*of procedure assign-NC*)
```

```
FUNCTION multiplexing-is-possible(remote transport address,
QoS requests,NCEP-ID): boolean;
(*boolean function*)

  BEGIN
 Success:=false;
 IF class <> 2NFC THEN
 BEGIN
  REPEAT
  Position on next NCDB;
(* the first time is the first one *)
(* only Network Connections in stati (Wait for N-Connect confirm and
  Data Transfer) will be considered *)
  IF remote-transport-address=remote-transport address of this NCDB
  THEN
   IF cost-saving THEN
      success:=true
   ELSE
    IF QoS-requests match with this N.C. THEN
      IF available bandwidth is sufficient THEN
        BEGIN
          Success:=true;
          NCEP-ID:=this NCDB's NCEP-ID;
        END;
  UNTIL (scanned all N.Cs)or(Success);
 END;
Multiplexing is possible:=success;
END;(*of procedure*)
```

```
PROCEDURE N-connect-indication;
(*manages N-connect-indication issued by N.L., see
interface description*)

BEGIN
 reason:=ok;
 IF not (refuse new connection flag) THEN
 BEGIN
  alloc ref (success,ref);
  IF success THEN
  BEGIN
    get-NCDB(Success);
    IF success(*NCDB found*)THEN
    BEGIN
      Initialize-NCDB;
      (*set all values to default initial values,set
        NCEP-ID*)
      Signal N-connect-response;(*To N.L.*)
      NC-Status:=Data Transfer;
    END
    ELSE(*no NCDB available*)
      reason:=no NCDB available;
  END
  ELSE
    reason:=no ref available;
 END
 ELSE
   reason:= refusing new connections;
 IF reason <> ok THEN
   Signal N-disconnect-request(reason);(* to Network Layer*)
END;(*of procedure N-connect-indication*)
```

7.4 THE GARBAGE QUEUE MECHANISM

To every network connections a dummy transport connection is linked, it is actually only an anchor of TPDU, called Garbage queue (G.Q.).

This G.Q. has the purpose of collecting all TPDUs to be sent which have to be assigned to the NC as a whole because they cannot or can no longer be related to a Transport Connection (i.e. to a TCDB).

These TPDUs are:

- DR in answer to a CR because the TCDB does not exist
- DC because this TPDU is sent only as an answer to a DR, and after the reception of a DR the T.C. (and so the TCDB) is released.
- ER because when recognizing an error the T.C. (and so the TCDB) is released and an ER is sent.

This G.Q. shall be managed as a normal T.C. from the point of view of concatenation, remembering that ER shall not be concatenated and that DR shall be the last one in a concatenated NSDU.

When receiving a Reset indication or a Network provider initiated disconnect indication DR and DC shall be flushed because they can be lost (because of TS1, TS2).

ER shall be kept because in some situations its loss may lead to undefined situations.

7.5 TRANSMISSION PROCEDURES.

The main issue is to find a mechanism which allows concatenation of one TPDU containing user data with one or more of the other TPDUs and segmentation of a data TSDU in TPDUs. Such mechanism must optimize [15] the use of the OSIRIDE Network (X.25 ITAPAC) so it must be able to fill the user data field of the N_DATA_request whenever it is possible.

For this reason the segmenting function is driven by the Network service machine (N_machine). When this machine is enabled to send TPDUs on a network connection it scans the TCDBs of the transport connections multiplexed on that network connection in

---

[15] They are not important from the throughput point of view, you save only lower layer header, but are important from the charging point of view as you are charged per packet or half-packet.

search of a TPDU containing user data (CR, CC, DT, DR, ED TPDU in the OSIRIDE first release), performing a cyclic scansion algorithm.
When it finds the TPDU it stores its type and length; such length is the real length in case of a CR, CC, ED TPDU and initially 0 in case of a DT TPDU; in fact in order to optimize the use of the transmission medium such length will be chosen (and the segmentation performed) after the end of the concatenation algorithm so as to fill, if possible, the NSDU.

The portion of the buffer containing the NSDU to be transmitted that can be utilized by the concatenation algorithm is NSDU LENGTH (128) - LENGTH OF TPDU (containing user data)
The concatenation of only one control TPDU per transport connection is allowed. All DC TPDUs are in garbage queue and we have decided to consider garbage queue as a transport connection queue so we send only one DC in an N-DATA request.
ER TPDU cannot be concatenated.

This transmission structure optmizes the use of the transmission medium even when a network signalled error occurs. In fact the transport retains the buffer containing the TSDU (T_DATA request) until it has been completely acknowledged.
It only puts a marker for every DT TPDU sent. This marker contains the DT TPDU number (YR_TU_NB), length and the pointer to the user data field. So when the resynchronization procedure is invoked the transport protocol machine will remove all the markers down to the one corresponding to the resynchronization point.

```
 PROCEDURE TRANSMIT
(* called by N_Machine after an enable trasmission or a SEND_TPDU *)
   Begin
     if transmission_flag = true then
        Begin
           locate the first TCDB for the scanning algorithm
           SEARCH (TCDB_Pointer,found,TPDU_length, TPDU_type,TX-GO)
(* if found equals false the only significant information given by
SEARCH is TX-GO flag *)

(* found equals true  means that a CR,CC,ED,DT has been found, TPDU_type
indicates the type of the TPDU *)

(* TX-GO is true if at least one TCDB contains a TPDU to be sent
different from not mandatory AK TPDU *)

(* the variable TPDU length equals 0 when the TPDU is a DT TPDU
(see segmenting algorithm ) and is the exact length of the
TPDU when it is a ED,CR or CC TPDU *)

(* the variable TPDU_type can indicate one of the following TPDUs:
DT or CR or CC or EDTPDU. *)

        IF TX-GO =true THEN
        BEGIN
           if (is Garbage Queue AND is ER TPDU) then
           BEGIN
              N_DATA_request
              transmit_flag = false
           END ELSE
           BEGIN
              CONCATENA (N_DATA_request buffer,found,TPDU_type,
                         TPDU_length,current_TCDB_pointer)
              N_DATA_request
              transmission_flag =false
           END;
        END;
     END;
END; (*of procedure transmit *)

PROCEDURE SEARCH
BEGIN
   initialize scanning pointer;
(* this information is contained in the input variable TCDB_pointer **
   REPEAT
   IF ((CR_flag) or (CC_flag) or (DR_flag) or
      ((( EX_flag = true ) and
      (ED_anchor ne NIL)) and ((T_status = Data Transfer) or
      (T_Status = Data Transfer T.D.))) or
      ((TSDU to be sent = true) and (T_Status = Data tranfer)) THEN
   BEGIN
      found := true;
      fill output parameter of the procedure;
   END ELSE
   position on next TCDB;
   UNTIL (found or
```

```
            (all the elements of the TCDB list have been scanned))
 END; (* of procedure SEARCH TCDB containing a (CR,CC,DT,ED) TPDU *)

 PROCEDURE CONCATENA (found, TPDU-type,TPDU-length,
                      current-TCDB-pointer)
(* if TPDU-type = DT then TPDU-length is equal 0 ;
  if found equals false the TPDU-type and TPDU-length are not
  significant *)
 BEGIN
    if found then
       available area := 128 - TPDU-length;
    repeat
      if sending request of RJ,EA,AK,DC then
      BEGIN
(* the test of the control-TPDU  flags is performed in the following
  order:RJ,EA,AK,DC. When a true flag is discovered the procedure
  creates the corresponding TPDU and   puts it in the buffer
  containing the concatenated   TPDUs to be sent in this N_DATA request
  and decrements available area size.
  The control TPDUs sending flags will be tested in the following
  order: DR,RJ,EA,AK,DC.
  At the first occurrence of a true flag this procedure puts the
  corresponding TPDU in the buffer containing the concatenated
  TPDUs in the N_DATA_request *)

   DR flag resets all other flags;

  RJ flag resets AK flag *)

        create TPDU;
        insert in N_DATA_request buffer;
        reset TPDU flag;
        available area := available area - IPDUlength;
        position on next TCDB;
     UNTIL (scanned all the TCDB or available area not significant)
                                                  (* too small *)
     if found then
     BEGIN
        restore TCDB pointer to the input value;

(* this information is contained in the input variable
  current TCDB pointer *)

        case TPDU type of
        CR,CC,DR: BEGIN
                    create TPDU;
                    reset TPDU flag;
                    insert in N_DATA_request buffer;

                  END;
        ED    :   BEGIN
                    take ED TPDU from anchor;
                    insert in NSDU;
                    IF class 2 THEN
                      release ED-TPDU from EX-QUEUE;
                  END;
```

```
          DT    : BEGIN
 (* input length is zero *)
                    if disponible area > k ("sufficiente") then
                    BEGIN
                       create DT TPDU as long as possible;
                       update table TSDU mark DT;
                       if end of TSDU then
                       BEGIN
                          set EOT in header;
                          reset TSDU to be sent;
                          if class = 2 then
                            signal ENABLE-TRANSMISSION;
                       END;
                       insert in NSDU
                    END;
                 END;
    END;
 END;(*of procedure CONCATENA*)
```

7.6 A MECHANISM FOR MANAGING REFERENCES IMPLEMENTING AUTOMATICALLY THE FROZEN REFERENCE MECHANISM.

If K is the maximum number of transport connection simultaneously open and the range of allocable references is N (N > S*K) we can use an allocation, deallocation mechanism which guarantees the freezing of the reference for a time T > S*network transit delay.

S gives the security of freezing, it must be greater than 2. This reference management alogorithm needs two lists:

1. list A which contains the references allocated to transport connections

2. list B which contains the free references.

7.6.1 Reference allocation

The procedure for allocating the reference is the following:

```
Procedure allocreference
Begin
  extract the first reference of list B
  insert such reference in list A
end
```

7.6.2 Reference deallocation

The procedure which frees the reference is the following:

```
Procedure freereference
Begin
   extract the reference from list A
   insert such reference as last element in list B
end
```

7.7 AN ALGORITHM FOR CLASS SELECTION.

Class and options are selected utilizing the quality of service parameters contained in the T_Conn request and some statistical information.

```
 PROCEDURE CHOICE_CLASS (Quality of Service parameter, class number)

(* Quality of service parameter are : Throughput (I->R), Throughput
  (R->I),Resilience and Expedited data transfer service.
(* Class number allowed values are : 2, 3 *)

 BEGIN
    IF expedited data transfer service = true then
       EXPEDITED OPTION
    ELSE
       NO EXPEDITED OPTION
    IF resilience = high THEN
      class number := 3
    ELSE
      IF "good network resilience" THEN
(*this information is calculated by a statistical analizer of X.25
Network performance *)
      BEGIN
         class number:=2;
         IF (one of the two throughput parameters = (high or very high)
             OR Expedited option) THEN
(*as usually required by a file transfer *)
           USE OF EXPLICIT FLOW CONTROL OPTION
         ELSE
           NO USE OF EXPLICIT FLOW CONTROL
      END ELSE
         class number:=3
 END;(* of procedure CHOICE_CLASS *)
```

7.8 AN ALGORITHM TO DETERMINE WHEN MULTIPLEXING HAS TO BE USED

In the OSIRIDE release one transport layer only the owner of the network connection can multiplex transport connections on such

network connection and the main application is FILE TRANSFER (very unbalanced traffic).
The algorithm described in the following procedure is tailored on such environment.

```
PROCEDURE MULTIPLEXING? (Quality of Service parameter,
                Source network address,Destination network address,Result)

(* Quality of service parameter are : Throughput (I->R), Throughput
  (R->I),Resilience and Expedited data transfer service.
(* Result is a boolean variable, Result = yes means that
  multiplexing has to be performed *)

BEGIN
   IF ((one throughput parameter equals high) and (the other equals
        (low or very low)) and "there are more than three transport
        connection usually connected between this network addresses")
(* This information is calculated on a statistical basis *)
      THEN
        MULTIPLEXING:=true
      ELSE
        MULTIPLEXING:=false
END; (* of Procedure PREPAREMULTIPLEXING *)
```

APPEND. A - FILE MANAGEMENT IN OSIRIDE TRANSPORT

Introduction

File management is a necessary functionality in all OSIRIDE layers. We are giving here an example of how this can be realized. However this may be implementation dependent so this appendix could be considered just as an example.

Reasons for using files in transport

Transport (the same is valid for session) needs access to files for various reasons among which at the moment we can mark the following ones:

- initialization of global data (maybe shared between layers) of the system: session users names, local transport addresses
- time out lenghts, maximum number of connections etc.
- management of addressing tables: level 4 addresses versus network addresses
- Error logging

Inserting the file system management into the transport (session) is extremely heavy both from a memory occupation point of view and from a performance point of view. What is needed is a simple and unified mechamism to manage files that allow transport (session) to continue execution in parallel with file access.

Solution to the problem

A solution to this problem could be the following: Create another process, external both to transport and session, that be able to manage files for them.

This task that we'll call file server will dialogue with transport (session) via normal interprocess communication mechanisms, receive commands from transport (session) and send back answers when needed. This file server could also dialogue with a configuration task from which it could receive write commands to update configuration files. The configurator task could have either an operator interface or (better) could be a session user so allowing to receive configuration commands from CME. In this way the transport (session), when needing access to files will issue a command (send) to file server and continue execution.

If the answer from file server is necessary to proceed processing transport (session) shall freeze the state of that particular transport (session) connection until receiving answer and go on processing the other transport (session) connections. The answers from file server will arrive to transport (session) in

the normal pipes associated to the connection. There are two other advantages in this solution:

- the files organization can be changed transparently in transport (session) and anyone is allowed.If some processing is necessary over data contained in file, for instance selection of network connection attributes, it may be performed into the file server so realizing a sort of coprocessing that leads to a better transport session) Performance.

Structure of the file server

File server may have the following structure:

```
Program file server;

  Procedure XX;
(*Example of a read file command with associate
processing*)
BEGIN
  Open file (XXXX);
  Read Record(command params);
  Process(data contained in record);
  Send(output of processing, to link indicated in command);
  Close file(XXXX);
END;

  PROCEDURE YY;
(*Example of an error logging command*)
BEGIN
  Open file (ERRLOG);
  Prepare ERRLOG message(command params, time, etc.);
  write record (ERRLOG message);
  Close file(ERRLOG);
END;

  BEGIN
  Receive(command);
  CASE command of
    X:   ZZ;
    Y:   YY;
    :::::::::
    Z:   ZZ;
END.
```

The process will suspend on receive if there are no pending requests, an incoming one will reactivate it.

APPEND. D - REFERENCES

[1] Information Processing Systems - Open Systems Interconnection - Basic Reference Model - ISO 7498.
[2] Information Processing Systems - Open Systems Interconnection - Transport Service Definition - Draft Proposal ISO/DP 8072.
[3] Information Processing Systems - Open Systems Interconnection - Transport Protocol Specification - Draft Proposal ISO/DP 8073.
[4] ONE/V2 Concepts and facilities - OLIVETTI report 3981650 H
[5] ONE/V2 Services and protocols - OLIVETTI report 3981730 H
[6] ONE/V2 Transport protocol - OLIVETTI report 3981690 M
[7] "Guidelines for the implementation specifications of the OSIRIDE Layers" - CNUCE report C83-5.
[8] "The OSIRIDE SESSION Layer" - Doc OSIRIDE/83/SES/02