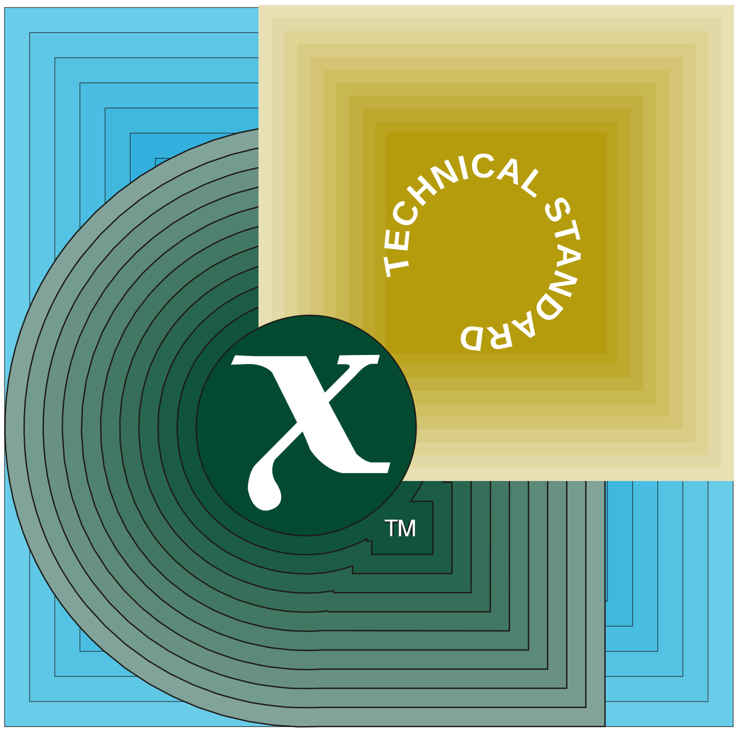
Technical Standard

Multiprotocol Transport Networking (XMPTN): Access Node



THE Open GROUP

[This page intentionally left blank]

X/Open CAE Specification

Multiprotocol Transport Networking (XMPTN): Access Node

X/Open Company Ltd.

X/Open CAE Specification

Multiprotocol Transport Networking (XMPTN): Access Node

ISBN: 1-85912-106-3
X/Open Document Number: C521

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.org

Contents

List of Examples

List of Figures

List of Tables

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done in any one of the following ways:

- anonymous ftp to ftp.xopen.org
- ftpmail (see below)
- reference to the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information using ftpmail, send a message to ftpmail@xopen.org with the following four lines in the body of the message:

```
open
cd pub/Corrigenda
get index
quit
```

This will return the index of publications for which Corrigenda exist. Use the same email address to request a copy of the full corrigendum information following the email instructions.

This Document

This document is an X/Open CAE Specification. It defines the XMPTN Access Node, through which the mixed protocol networking services provided by XMPTN are accessed by applications.

X/Open Multiprotocol Transport Networking (XMPTN) architecture supports mixed transport protocol networking, enabling application programs that were designed to operate over one transport protocol — such as SNA, NetBIOS, OSI or TCP/IP — to run over other transport networks. With XMPTN, existing applications can still run when the transport protocol for which they were written is exchanged for another one, and applications written for one transport can be introduced into networks which use other transports.

X/Open has published an XMPTN Architecture Guide which explains the role of access node in XMPTN. The associated XMPTN Address Mapper is defined in the separate X/Open specification, and the formats of message and data structures used by XMPTN are defined in the associated XMPTN Data Formats specification. X/Open offers a package of all four XMPTN documents, in Document Set T504.

This specification does not prescribe any specific implementation of MPTN.

Audience

This document is intended primarily for use by implementors of the MPTN Access Node functionality who wish to conform to the X/Open MPTN formats and protocols specification. It will also be of interest to diagnosticians who interpret these MPTN Access Node formats when analysing line flows, and to others who may wish to learn about the MPTN architecture from the data formats.

Network designers, network managers, or application program vendors who are interested in mixed protocol networking or network interconnection, as addressed by the MPTN architecture, are referred to the X/Open **Multiprotocol Transport Networking (MPTN) Architecture Guide** (see **Referenced Documents** on page xviii).

Structure

This specification is organised as follows:

- **Part 1:**
  - — Chapter 1 presents the terminology used in the architecture.
  - — Chapter 2 describes the protocol boundary structure for the XMPTN Access Node.
  - — Chapter 3 gives an overview of the protocol used.
- **Part 2:**
  - — Chapter 4 describes the processing specifications
  - — Chapter 5 describes the initialisation activity.
  - — Chapter 6 describes the connection-oriented protocols.
  - — Chapter 7 describes the connectionless protocols.
- **Part 3:**
  - — Chapter 8 describes AF/INET sockets transport user characteristics.
  - — Chapter 9 describes TCP/UDP transport provider characteristics.
  - — Chapter 10 describes SNA transport user characteristics.
  - — Chapter 11 describes SNA transport provider characteristics.
  - — Chapter 12 describes NetBEUI transport user characteristics.
  - — Chapter 13 describes NetBIOS transport provider characteristics.
  - — Chapter 14 describes SPX/IPX transport provider characteristics.
  - — Chapter 10 describes OSI transport user characteristics.
- **Part 4:**

  Appendices A to F provide supplementary information.

Trade Marks

Referenced Documents

The following documents are referenced in this Specification:

APPN Pacing
: Systems Network Architecture, Advanced Peer-to-Peer Networking (APPN), Architecture Reference; publicly available from IBM branch offices under the IBM reference SC30-3422.

ISO / IEC 8348
: ISO / IEC 8348:1987(F): Information Processing Systems — Data Communications — Network Service Definition.

LAN Technical Reference
: IBM LAN Technical Reference: IEEE 802.2, publicly available from IBM branch offices.

NetBIOS API
: IBM NetBIOS Application Program Interface (SC30-3587-00), publicly available from IBM branch offices.

SMB
: X / Open CAE Specification, October 1992, Protocols for X / Open PC Interworking: SMB, Version 2 (ISBN: 1-872630-45-6, C209).

XMPTN Address Mapper
: X / Open CAE Specification, Multiprotocol Transport Networking (MPTN): Address Mapper (ISBN: 1-85912-101-2, C520).

XMPTN Architecture Guide
: X / Open Guide, December 1995, Multiprotocol Transport Networking (XMPTN) Architecture (ISBN: 1-85912-116-0, G506).

XMPTN Data Formats
: X / Open CAE Specification, Multiprotocol Transport Networking (XMPTN): Data Formats (ISBN: 1-85912-111-X, C522).

X/Open CAE Specification

Part 1:
MPTN Access Node Overview

X/Open Company Ltd.

Chapter 1

Introduction

Multiprotocol Transport Networking (MPTN) architecture supports mixed protocol networking: running application programs that were designed to operate over one transport protocol — such as SNA, NetBIOS, OSI or TCP/IP — over additional transport networks.

This specification describes the architecture of an X/Open MPTN Access Node, which allows programs to access the mixed protocol networking services provided by MPTN.

Throughout this specification the term **XMPTN** is used to refer to the X/Open specification of MPTN.

1.1 Organisation

This document describes the MPTN access node in the following way:

- It defines the components of the MPTN access node.
- It defines the MPTN functions in terms of the actions of the access node components.

A useful primer on MPTN is the **MPTN Architecture Guide** (see **Referenced Documents** on page xviii).

1.2 Terminology

In MPTN terminology, the term *transport user* refers to any user of the MPTN components. It can be an application program written directly to MPTN, or application-support functions that support existing sets of applications. The term *transport provider* means a provider of communication service at the transport layer; to be more precise, it means providing the service at the transport/network layer, including subnetworking services. A transport provider uses one *transport protocol* to govern the exchange of information between nodes, thus providing a *transport network* of that type.

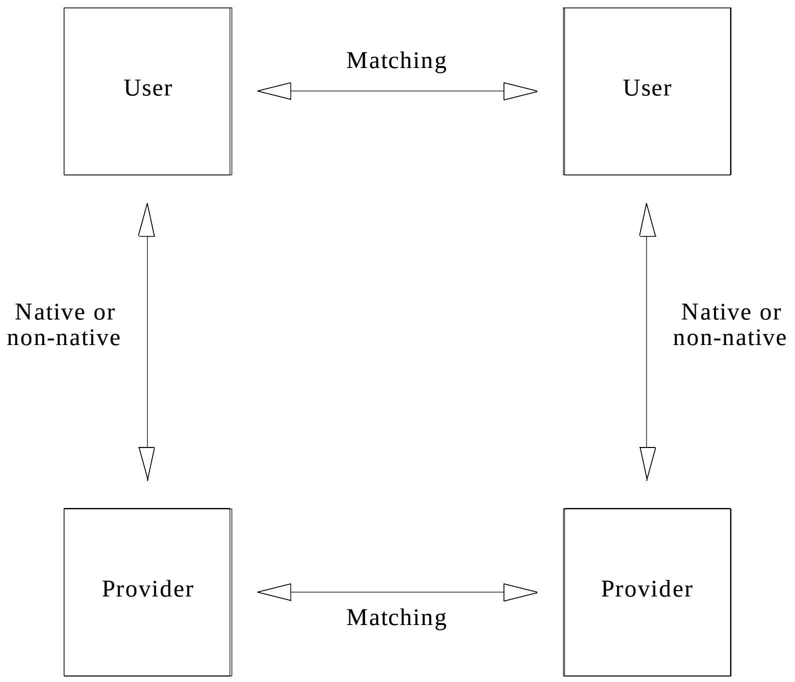


Figure 1-1 ‘‘Matching’’ and ‘‘Native’’

As illustrated in Figure 1-1, there are two sets of relationships that are important to this architecture: peer relationships, and relationships between transport users and transport providers. The term *matching* describes a horizontal relationship between peers. In the MPTN architecture, transport users must always match. To *match* means to have the same application support and to belong to the same application programming family. For example, two sockets programs match or two Common Programming Interface for Communications (CPI-C) programs match, but a sockets program and a CPI-C program do not.

The terms *native* and *non-native* describe a vertical relationship between a transport user and its corresponding transport provider. With respect to a particular transport user, a transport network that provides the address type and transport characteristics assumed in the transport-user design is native to that transport user. A transport provider that does not provide that address type and those transport characteristics is non-native. For example, TCP/IP is native for AF-INET sockets and non-native for SNA LUs. Application programs designed assuming a particular transport provider are native to that transport provider. At the same time, they are non-native to another transport provider. A *native node* is a node with no MPTN capability. For example, a node with SNA application programs running only over an SNA transport is a native node. MPTN addresses the need for transport users to run over non-native transport providers.

When discussing protocol, the terms *initiator* and *target* are used to refer to the transport user, transport provider, or access node that actively initiates the protocol sequence (initiator) or is the passive recipient of the initiation (target). For example, the transport user that starts a connection sequence is the initiator, and the transport user with whom the connection is sought is the target. Similarly, the transport user that wishes to close a connection is the initiator (of the termination) and the partner transport user at the other end of the connection is the target.

1.3 Access Node Components

This section describes a model to explain the interactions between a Transport User and a Transport Provider. A product is not required to follow this model in implementing XMPTN.

Figure 1-2 shows the components in an MPTN Access Node that allow transport users to access non-native transport providers.

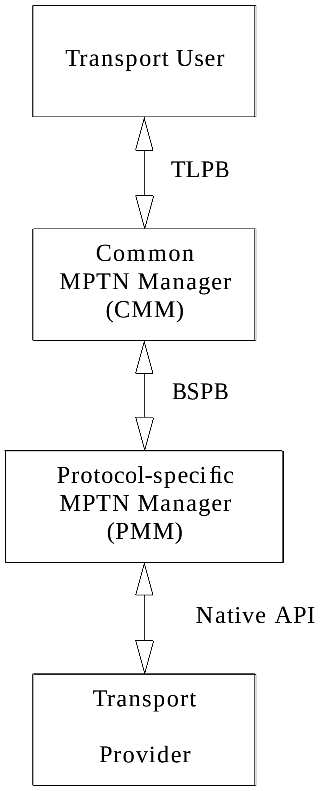

Transport User
TLPB
Common MPTN Manager (CMM)
BSPB
Protocol-specific MPTN Manager (PMM)
Native API
Transport Provider

Figure 1-2 MPTN Access Node: Components and Interfaces

The four components shown in this figure are described in the following four sections. The interfaces between components are then discussed.

1.3.1 Transport User

A transport user can be an application program that uses transport services directly, or it can be a service program that provides a programming interface to application programs. When the transport user provides another protocol-specific API, such as CPI-C, sockets or network basic extended user interface (NetBEUI), application programs running on these interfaces gain access to multiple transport protocols without change.

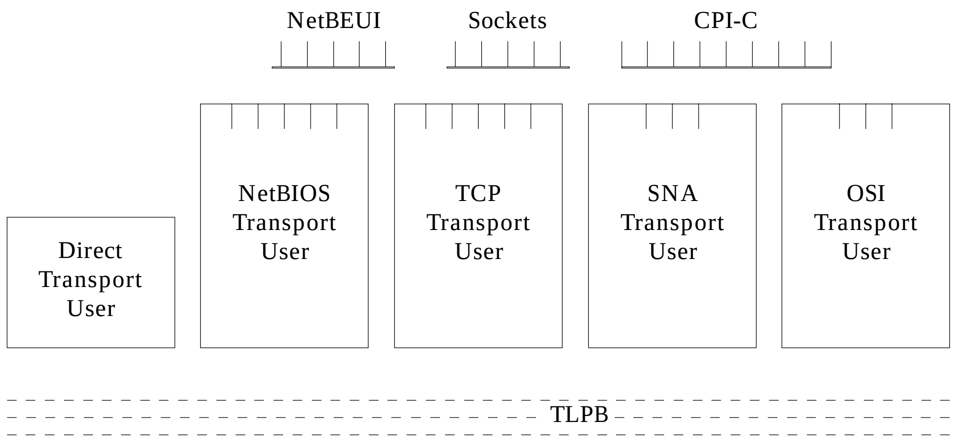


Figure 1-3 Transport User Types

Figure 1-3 illustrates a number of different transport users. The graphic symbol above some of the boxes represents an Application Programming Interface (API) to that transport user. Below the boxes is the Transport Layer Protocol Boundary (TLPB), which is the interface between the transport users and the Common MPTN Manager (CMM). The three different types are shown with different sized boxes to indicate differences in the amount of function involved:

- Those that represent a set of programs that use the TLPB directly. These are referred to as Direct Transport Users. This box is the smallest since it represents no additional function between the transport user and TLPB.
- Those that support user APIs with function very similar to the TLPB function. These boxes are middle-sized boxes, since they have the ability to do syntax mapping between the API and the TLPB. The examples shown are the NetBIOS Transport User and the TCP Transport User.
- Those that support user APIs that require additional protocol above and beyond that provided by the TLPB. These are the largest boxes, since they require protocol above the transport layer interface. For example, half-duplex LU 6.2 includes conversation state management and confirm and error processing over and above the transport protocols. OSI includes the functions from the session, presentation, and application layers. CPI-C is shown as the API supported by both OSI-TP and LU 6.2; others are available as well.

1.3.2 Common MPTN Manager

Referring again to Figure 1-2 on page 5, the common MPTN manager is the component that connects a transport user to a particular non-native transport provider, maps between different addresses, and determines which compensations are required.

The CMM may be involved in connection setup for native connections and with datagrams to the point of determining that the native transport provider is available and is the best match for the transport user's needs. No other MPTN services are needed for matching native users and providers.

An MPTN access node has one CMM, no matter what combinations of transport user types and transport provider types it supports. Thus, the common MPTN function can be reused for different protocol combinations.

1.3.3 Protocol-specific MPTN Manager (PMM)

The protocol-specific MPTN manager is the component that performs any functions needed by non-native users that depend on the underlying transport provider. For example, the PMM associated with a TCP transport provider may use the TCP domain name service (DNS) to resolve transport user addresses of any type to IP addresses. This DNS mechanism is not available for use by other transport providers.

An MPTN access node contains one PMM for each transport-provider type supported. Each PMM will perform the compensations required to support its underlying transport provider; however, it is the CMM that determines which compensations are required. These compensations may be available as a common library.

1.3.4 Transport Provider

The transport providers are existing software components that provide the transport-layer functions associated with a particular protocol stack. For example, the SNA transport provider used for non-native application programs is the full-duplex transport function, implemented with either a base full-duplex SNA logical unit type 6.2 (LU 6.2) conversation or a simulated full-duplex conversation running over two half-duplex LU 6.2 conversations. Another example is TCP/IP transport provider, supporting TCP for connection oriented reliable stream transport, and UDP for connectionless datagram service.

1.3.5 Interfaces

The TLPB and the BSPB, the protocol boundaries between the transport user and the CMM, and the CMM and the PMM, respectively, are described in Chapter 2. The Native API, shown between the PMM and the transport provider, is different for each transport protocol supported. Details of how native APIs are used can be found in Part 3 of this document.

1.4 Access Node Function

Of the four components described above, only CMM and PMM provide MPTN functions. This section describes their overall function. Following chapters detail the protocols used.

1.4.1 CMM Functions

Two types of functions are supported by the CMM: local functions, and those it performs in cooperation with other CMMs.

The local functions are:

- *transport user registration*

  In order to make itself accessible, a transport user registers its address with the CMM. The CMM uses this registration to support the routing of request to transport users.

- *protocol selection*

  When multiple transport providers are available, the CMM selects the order in which the providers are tried. This order can be system- or user-defined, or can be based on a cache of successful connections.

- *compensation determination*

  The CMM analyses the transport characteristics of the transport user and the transport provider in order to determine which compensations are required. The compensations used for a particular (user,provider) pair are computed from a universal set of compensations.

The functions performed in cooperation with other common MPTN managers are:

- *connection establishment and termination*

  Establishment and termination of end-to-end transport-user connections are the two most sophisticated functions of the CMM.

  In connection establishment, the initiating transport user indicates the desire for a connection with its partner by providing the partner's transport-user address and requested service mode to its local CMM. The initiating CMM employs MPTN address mapping to learn the partner's transport-provider address. The CMM then uses the connection establishment protocol of its selected transport provider and MPTN connection establishment protocol to establish an MPTN connection with appropriate service characteristics to the indicated target transport user. In the process, the CMM exchanges information about the characteristics of the end-to-end transport-user connection with its partner CMM.

  In the connection termination process, the CMMs must assure that the appropriate semantics are observed before closing the underlying transport connection.

- *non-native datagram routing*

  The source CMM adds an MPTN header (which includes the transport-user address of the source and destination) to the user datagram. The destination CMM uses this information to deliver the datagram to the destination transport user.

1.4.2 PMM Functions

The key functions of the PMM are:

- *protocol-specific address mapping functions*

  Two of the alternative address-mapping techniques defined in MPTN, algorithmic mapping and use of a protocol-specific directory, are protocol-specific. When either of these techniques is used, the PMM performs these functions when registration or resolution are requested.

- *compensation mechanisms*

  Each PMM receives information from the CMM as to which compensations are required for a particular transport user. The two PMMs at either end of the MPTN connection then insert and remove information from the flow of user data on the connection in order to cooperate to give the transport users the services they require. For most compensations, this information is minimal, usually no more than a single octet marker indicating the type of message. However, some compensations are more complicated, for example, expedited data and session outage notification.

Many transport users need the ability of sending to either themselves or another transport user on the same node. If the transport provider does not provide this ability, the PMM must provide an implementation specific way of supplying this loopback ability.

1.5 MPTN Addresses

It is a characteristic of a multiprotocol transport environment that the transport user and the transport provider use different protocols and have different address spaces for the representations of entities. Thus, an MPTN access node has the responsibility of mapping between the representation in the transport user address space and the representation in the transport provider space.

Within each address space there are *node addresses* and *local addresses*. The *node address* is the network address of an access node, and is unique within a given network. In order to identify a specific transport user within an access node, a *local address* is used. In some protocols, it is necessary to use both of these addresses to uniquely identify a transport user in a network.

Table 1-1 shows specific instances of these different addresses for some common protocols. Note that in some protocols both the node and local address are the same. In these cases, no local routing is done at the destination node since every transport user is independently addressed.

Address family	**Node address**	**Node + local address**
TCP	IP address	IP address.TCP_port
UDP	IP address	IP address.UDP_port
OSI	NSAP	NSAP.Tselector
NetBIOS	NetBIOS name	NetBIOS name
SNA (LU)	Netid.LUname	Netid.LUname

Table 1-1 Address Formats

Unless specifically noted otherwise, the term *address* in this document refers to an *MPTN-qualified transport address*, which provides the address type, node address, and local address. Refer to the **MPTN Data Formats** Specification (Section 1.4, MPTN Address Formats) and **MPTN Data Formats** Specification (Section 8.3, MPTN Addresses) for more information about MPTN addresses.

Chapter 2

Protocol Boundary Structure

2.1 Interfaces

This chapter provides a description of the means by which the components of an MPTN Access node communicate with one another. The interfaces described are the Transport Layer Protocol Boundary (TLPB) and the Below Specific Protocol Boundary (BSPB). These interfaces are part of the model as described in Section 1.3 on page 5, and as such are not required of any implementation.

The TLPB is the transport user's *logical* interface into the MPTN network. Specifically, the TLPB provides the transport user with the ability to access the functions provided by components of the MPTN Access Node.

This interface is called a protocol boundary, and *not* a programming interface, in order to emphasise the importance of semantic as opposed to syntactic equivalence. Products implementing MPTN are not required to have syntactic equivalence to this protocol boundary, but should have semantic equivalence to it in order to ease application portability.

The BSPB is the interface that allows communication within an MPTN Access Node between the CMM and any PMMs installed. It is strictly a convenient way to describe interactions internal to the access node.

2.2 Protocol Boundary Naming Conventions

Both protocol boundaries use a call/return interface, with specific function calls referred to as *verbs*. Each time a verb is invoked, there is a *caller* which is the component that initiates the call and a *callee*, the component that is the recipient of the call. In this chapter, the components in question will be either the transport user, the CMM, or the PMM.

Calls from the transport user to the CMM, or from the CMM to the PMM, are called *downcalls*. Conversely, calls from the CMM to the transport user, or from the PMM to the CMM, are called *upcalls*.

Calls between the transport user and the CMM have names of the form M_XXXXX, while calls involving the PMM have names of the form P_XXXXX, with the X's representing some descriptive function name. Downcalls have *_DC* appended to the name, while upcalls have *_UC* appended.

Figure 2-1 illustrates the relationships between functions in this upcall/downcall model.

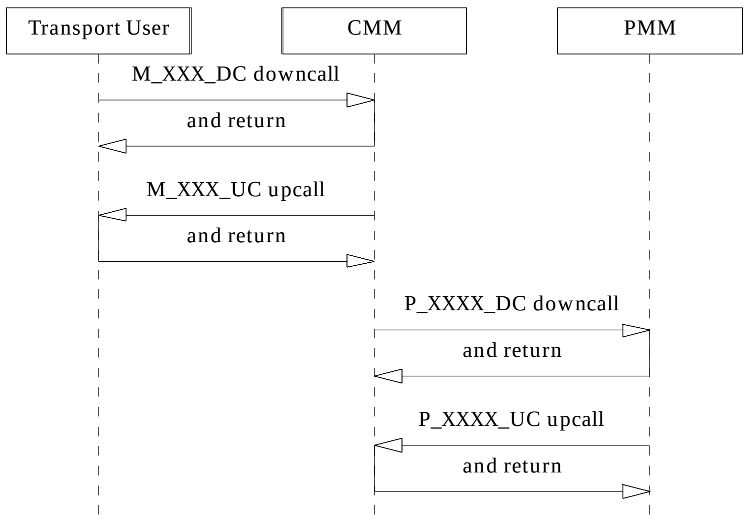


Figure 2-1 Upcall/Downcall Model

Two types of calls are defined for the call model, *request* calls and *result* calls. In request calls, the caller asks for some action to be taken by the called routine. In result calls the caller (who was the called routine in the request call) tells the called routine (who was the caller in the request call) whether or not the requested action occurred.

Generally, but not in all cases, a downcall is a request call, that is, it is initiated by the transport user to request a service. Upcalls which are asynchronous indications to the transport user (such as an incoming connection attempt, or receipt of a datagram) are also considered request calls.

All calls are *non-blocking*; that is, the request call will always return immediately, even if the requested task is still in progress. In this way, the caller is not prevented from performing other tasks while the requested task is in process. When a task cannot be completed immediately, the caller will receive an RC_PENDING return code. Upon completion of the requested task, a result call will be made to the caller of the request call. In general, this result call has the same name as the call it completes, with the suffix _DC replaced by _UC or vice versa. For example, an M_CONNECT_DC that returns RC_PENDING will be followed by an M_CONNECT_UC with the *status* parameter (and possibly the *sec_status* parameter) indicating the success or failure of the request.

Figure 2-2 is an example of a sequence of calls used to bind an address to a transport endpoint.

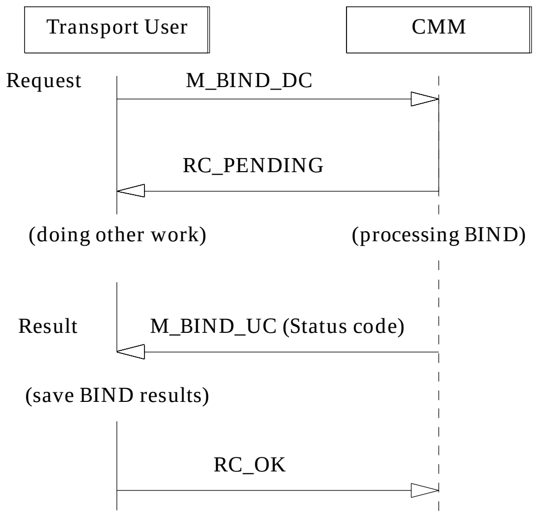


Figure 2-2 An Example Call Sequence

The TLPB verbs, grouped by function, are listed below, with a short description of the function of each. See Appendix B for details on the TLPB verbs.

- Initialisation

M_CREATE_DC	Create a transport endpoint.

- Addressing

M_BIND_xC	Associate an address to a transport endpoint.
M_CONFLICT_UC	Conflict found in usage of an address which the user indicated in an M_BIND_DC.
M_UNBIND_DC	Remove an address from a transport endpoint.

- Connection setup and termination

M_CONNECT_xC	Setup a connection to a remote address.
M_LISTEN_DC	Establish ability to receive connections from remote addresses.
M_ACCEPT_DC	Indicates acceptance/rejection of a connection request from a remote address.
M_ACCEPT_UC	Indicate arrival of a connection request from a remote address.
M_CLOSE_xC	Request closure of a connection.
M_CLOSED_UC	Indicates connection has been ended.

- Connection-oriented data transfer

M_SEND_DC	Transfer data to a remote address using a connection.
M_RECEIVE_UC	Indicates arrival of data from a remote address on a connection.

- Connectionless data transfer

M_RCVDG_INIT_DC	Establish ability of CMM to receive datagrams.
M_SEND_DG_xC	Transfer data to a remote address using datagrams.
M_RCV_DG_UC	Indicates arrival of a datagram from a remote address.

As an aid to understanding the TLPB in relation to the X/Open Transport Interface (XTI), an approximate mapping is shown in Table 2-1. The purpose here is to provide a reference by which someone familiar with XTI can get a feel for the approximate purpose of the TLPB verbs. Note that the TLPB uses an upcall/downcall model and the XTI uses a downcall/polling model.

TLPB Verb	XTI Verb
M_CREATE_DC	t_open
M_BIND_DC	t_bind
M_BIND_UC	
M_CONFLICT_UC	
M_UNBIND_DC	t_unbind
M_CONNECT_DC	t_connect
M_CONNECT_UC	t_rcvconnect
M_LISTEN_DC	t_listen
M_ACCEPT_DC	t_accept
M_ACCEPT_UC	
M_CLOSE_DC (orderly)	t_sndrel/t_close
M_CLOSE_DC (abortive)	t_snddis/t_close
M_CLOSE_UC	
M_CLOSED_UC	
M_SEND_DC	t_snd
M_RECEIVE_UC	t_rcv
M_SEND_DG_DC	t_sndudata
M_SEND_DG_UC	t_sndudata
M_RCVDG_INIT_DC	t_rcvudata
M_RCV_DG_UC	t_rcvudata

Table 2-1 TLPB to XTI Verb Mapping

The BSPB verbs, grouped by function, are listed below, with a short description of the function of each. See Appendix C for more details about the BSPB verbs. Note that these verbs are internal to the model of the access node described here; implementors are free to use their own model for the internal structure of the access node.

- Initialisation

P_CREATE_DC	Create a transport endpoint.

- Addressing

P_REGISTER_xC	Register the transport user with a directory service.
P_DEREGISTER_DC	Remove the transport user from a directory service.
P_CONFLICT_UC	Conflict found in usage of an address.

P_LOCATE_xC	Locate a non-native address.

- Connection setup and termination

P_CONNECT_xC	Setup a connection to a remote address.
P_ACCEPT_UC	Indicate arrival of a connection request from a non-native remote address.
P_CLOSE_xC	Request closure of a connection.
P_CLOSED_UC	Indicates that a connection has been closed.

- Connection-oriented data transfer

P_SEND_DC	Transfer data to a remote address using a connection.
P_RECEIVE_UC	Indicates arrival of data from a remote address on a connection.

- Connectionless data transfer (unicast and multicast)

P_SEND_DG_DC	Transfer data to a remote address using datagrams.
P_RCV_DG_UC	Indicates arrival of a datagram from a remote address.
P_JOIN_GROUP_xC	Join a transport provider multicast group.
P_QUIT_GROUP_DC	Quit a transport provider multicast group.

X/Open CAE Specification

Part 2:

MPTN Access Node Protocols

X/Open Company Ltd.

Chapter 3

Protocol Overview

This chapter provides a high-level overview of the protocol as an aid to understanding how the various components of an access node cooperate to provide a transport user with MPTN services. References to the more detailed protocol description in Part 2 of this document are included. Details on the verbs used are in Appendix B and Appendix C.

3.1 Initialisation

In order to use MPTN services, regardless of the type of service, a transport user must first make its address known to the network. It accomplishes this during the process of initialisation. Initialisation involves two steps:

1. *Creating* a transport endpoint for each instance of communication required.

   A transport endpoint serves to identify a particular transport user in an access node. To create the transport endpoint, the transport user issues an M_CREATE_DC, specifying the details of the type of communication requested.

2. *Registering* that transport user with the network.

   This is accomplished by issuing an M_BIND_DC.

Chapter 5 describes the initialisation process in more detail.

3.2 Establishing a Connection

If the transport user wants to setup a connection with a matching transport user, it must first know the partner's address. If the connection between the two transport users is native, then that address is all that is needed by the native transport mechanism to setup the connection. If the transport provider to be used is non-native, however, the MPTN components in the access node must find out how to get to that transport user's address via the non-native network by obtaining an address mapping, which gives the address of the transport provider that must be reached in order to find the transport user desired.

To initiate a connection, the transport user issues an M_CONNECT_DC with the matching transport user's address. The access node then attempts to obtain the address mapping for the matching transport user, using one of several methods. If the access node can find the matching transport user, it sends a request to establish a connection by:

1. Establishing an underlying transport provider connection.
2. Exchanging MPTN_Connect request and response as data on the underlying transport provider connection.

The MPTN_Connect contains the transport user addresses and information about the type of connection requested. See Section 6.1.5 on page 50.

In order to receive a connection attempt, a transport user must be listening for it. This is accomplished by issuing an M_LISTEN_DC. Once this verb has been issued, the transport user is prepared to receive any connection attempts addressed to it over the MPTN network.

When a connection attempt is directed to a transport user, the first thing that occurs is an underlying transport provider connection setup. Once this is established, an MPTN_Connect command flows as data across the underlying transport provider connection. Subsequently, a new transport endpoint is created for the connection, and the destination transport user will receive an M_ACCEPT_UC which includes information on the type of connection requested. The transport user either accepts or rejects this connection via the M_ACCEPT_DC. Acknowledgements of success/rejection will flow back to the matching transport user that requested the connection. See Section 6.2 on page 52.

3.2.1 Data Transfer

Data is transferred across the MPTN connection using the transport provider's native data transfer mechanism, with minimum embellishment. If differences exist between the transport services expected by the transport user and those provided by the transport provider, MPTN protocols compensate for the mismatch. The verbs used for data transfer are M_SEND_DC and M_RECEIVE_UC. See Section 6.3 on page 61.

In some cases, a transport user may want to send data in an expedited fashion. Expedited data is guaranteed to be delivered regardless of flow control. It's delivery is guaranteed to be no later than normal data sent after it. Some transport providers support expedited data natively. For those that don't, MPTN provides a mechanism to compensate for the lack of support which involves sending the data on both the connection and in datagrams. See Section 6.3.2 on page 63 for a description of this mechanism.

3.2.2 Connection Termination

If a transport user wishes to close a connection, it issues an M_CLOSE_DC, specifying the type of termination requested from four types:

- duplex orderly
- duplex abortive
- simplex orderly
- simplex abortive.

Simplex termination means that only the initiator's send capability is being closed (the target may continue to send data and the initiator receive it). Orderly termination assures that all data previously sent is delivered before the connection is terminated.

A transport user may receive an M_CLOSED_UC, which indicates that either the matching transport user has requested a connection termination or a network failure has occurred which caused the underlying transport provider connection to close.

Details about connection termination are found in Section 6.5 on page 71.

3.3 Datagrams

As in the connection-oriented case, the transport user that wishes to send a datagram must know the address of the matching transport user who is to be the recipient. By issuing an M_SEND_DG_DC, the transport user tells the access node to find the address mapping for the non-native recipient.

Once the address mapping is found, the access node sends the user's datagram in a MPTN_Datagram format using the transport provider's connectionless mechanism. If a transport provider does not provide a connectionless mechanism (for example, SNA), compensation is performed to provide the missing function.

On the receiving end, the transport user must be listening for datagrams in order to receive them. This is accomplished by issuing the M_RCVDG_INIT_DC. When the MPTN_Datagram arrives at the access node, the transport user will receive a M_RCV_DG_UC including the data from the received datagram.

For more information on connectionless MPTN service, see Chapter 7.

3.4 Multicast

The MPTN multicast function allows an MPTN transport user to originate a datagram to be sent to more than one destination. The destinations need not be in the same SPTN as the sender, and may use different transport provider protocols. The sender is not required to be a member of the group to which it is sending. Members of a transport user multicast group are limited to a single transport user NetID, but may span multiple transport provider NetIDs. The multicast capability provided has the normal datagram semantics, that is:

- delivery to any destination in the multicast group is not guaranteed
- data may arrive out of sequence at one or more destinations
- the sender knows nothing about the membership of the group
- multiple copies of the data may arrive at one or more destinations.

If the transport provider at some destination supports multicast as an inherent function, then MPTN uses this *inherent*[1] multicast capability; otherwise, MPTN will unicast copies of the message to all the intended recipients in the SPTN through a multicast server. Mixed mode environments are supported where some of the intended recipients reside on transport networks with inherent multicast capability whilst other recipients require the multiple unicast simulation of the multicast function.

To become a member of a group, an MPTN transport user issues an M_BIND_DC with an indication that the bind is to a group address. If the group does not exist, then the group is created implicitly, and registered with the Address Mapper (if present). Furthermore, if the transport provider does not support inherent multicast, a group entry is created within the Multicast Server, and the user is added to the membership list of the group.

In general, the mapping from a transport user group address to transport provider group address can be done in three ways:

- address mapper lookup
- protocol-specific directory lookup
- algorithmic translation.

However, because most multicast groups are transient entities, it is advisable to use a solution that permits dynamic addition and deletion of group addresses.

The Address Mapper handles dynamic registration of MPTN multicast groups. Since registration requests could come from multiple members of a multicast group, the Access Nodes and Address Mapper follow a procedure to ensure that each transport user group address maps to a single transport provider group address for each available transport provider.

If a Multicast Server is used in conjunction with an Address Mapper, it must register its own transport address with the Address Mapper so that multicast emulation is available. This is done via the MCS_CM_REGISTER_REQUEST command. Because the multicast server unicasts datagrams to the Access Node's CMM, the mapping of the CMM address to each transport provider address that uses a multicast server must be registered with the Address Mapper.

1. We need to distinguish between *inherent* and *native* multicasting. The former says that a protocol has the capability to multicast data. The latter says that the transport user and the transport provider protocols match and that the protocol has inherent multicast.

If addressing techniques other than the Address Mapper are used, then the facilities of multicasting may be limited. For example, with algorithmic mapping, it is hard to tell if a group exists or not. With protocol-specific directory services, if the directory cannot support dynamic additions and deletions to the database, then the multicasting is limited to pre-defined groups.

3.4.1 Broadcast

Broadcast is treated as a special case of multicast, that is, where every transport user of a particular type is a member of the group. Broadcasts are limited to a single NetID.

If broadcast is supported by a particular protocol, all transport users of that type will automatically be joined to the well-known group for that protocol.

Chapter 4

Processing Specifications

The messages that flow between MPTN nodes are described in the XMPTN Data Formats document. Each command format, as opposed to a header, carries with it a *processing specification* that tells the receiving node how to handle the command based on a number of criteria. This mechanism allows access nodes to use new functions when their partners support it and still be able to interoperate with back-level nodes.

Every MPTN command and optional field contains a specification of the appropriate action from a back-level node that does not understand the command or optional field type. All nodes must understand the processing specification so that they can take the correct action when they encounter unrecognised fields. If a node receives an MPTN command with an illegal command processing specification value, it will discard the command.

The processing specification is a one byte field in the common prefix that specifies:

- whether the command is a request or a response
- if a response, whether it is positive or negative
- the actions that should be taken by an MPTN node that does not recognise the command type
- if a negative response, the reason for the failure (rejected by the partner or not understood by the partner).

4.1 Bit Definitions

The bits in the processing specification byte are referenced by the names in Table 4-1. See also the **MPTN Data Formats** Specification (Section 8.1, Common Prefix for MPTN Command) and the **MPTN Data Formats** Specification (Section 8.2, Prefix for Optional Fields).

Table 4-1 Processing Specification Fields

Bit	Name	Explanation
[0]	MPTN_RSP	Whether the command is a request (0) or a response (1). Reserved in an optional field.
[1]	MPTN_RSP_NEGATIVE	Only set if MPTN_RSP also set; indicates whether the response is negative. This bit is set if the command is failed for any reason. If set, a diagnostics subfield is required unless the error is indicated by processing specification bits (bits 5,7 of this byte).
[2]	MPTN_DEST_REJECTED	Only set if MPTN_RSP also set, Set if the destination recognised the command and all its required subfields, but rejected it for some other reason. Reserved in an optional field.
[3}	Reserved.	Reserved for future use. See Section 4.1.3 on page 28 for the special definition of this term.
[4]	MPTN_EXT_REQUIRED	In a request, instructs a non-destination extended MPTN node to fail the command if it does not recognise the command tag or optional field tag. The command is turned into a negative response and returned to the source. In a response, if MPTN_EXT_UNRECOG is also set, indicates that a non-destination extended MPTN node failed the command because it did not recognise this command or field.
[5]	MPTN_EXT_UNRECOG	Set in either a request or a response. If MPTN_EXT_REQUIRED is also set, this bit indicates that the command was failed because this format was not recognised. If MPTN_EXT_REQUIRED is not also set, this field is informational, indicating that at least one non-destination extended MPTN node along the path failed to recognise a field or command that could be ignored.
[6]	MPTN_DEST_REQUIRED	In a request, instructs the destination to fail the command if it does not recognise the command tag or optional field tag. In a response, if MPTN_DEST_UNRECOG also set, indicates that the destination failed the command because it did not recognise this command or field. If this field is not set and the destination does not recognise the command or field, the command is discarded or the optional field is ignored.
[7]	MPTN_DEST_UNRECOG	Set in a response only; If MPTN_DEST_REQUIRED is also set, this bit indicates that the command was failed because this field was not recognised. Otherwise this bit is purely informational, indicating that the destination failed to recognise a field that could be ignored. The combination MPTN_DEST_UNRECOG and not MPTN_DEST_REQUIRED is never set in a command processing specification, since the destination discards the command in this case.

Note: ‘‘0’’ is the Most Significant Bit.

4.1.1 Failure Modes

Using the bits in the processing specification field, it is possible for a command to fail in several different ways:

- The destination recognised the command and all the optional fields marked MPTN_DEST_REQUIRED, but rejected it for some other reason.

  The MPTN_RSP, MPTN_RSP_NEGATIVE and MPTN_DEST_REJECTED bits are all set.

- The destination failed to recognise the command or an optional field marked MPTN_DEST_REQUIRED.

  The MPTN_RSP and MPTN_RSP_NEGATIVE bits are set in the command processing specification, and the MPTN_DEST_UNRECOG and MPTN_DEST_REQUIRED bits are set in the processing specification fields of the unrecognised format(s).

- A non-destination extended MPTN node failed to recognise the command or an optional field marked MPTN_EXT_REQUIRED.

  The MPTN_RSP and MPTN_RSP_NEGATIVE bits are set in the command processing specification, and the MPTN_EXT_UNRECOG and MPTN_EXT_REQUIRED bits are set in the processing specification fields of the unrecognised format(s).

- The command included an illegal command processing specification. Since the processing specifications are messed up, none of the bits can be relied on. Thus, it can not be determined if this is a request or response. The command is discarded.

4.1.2 Command Analysis

Figure 4-1 summarises the analysis of a format that can be made based on the command processing specification value.

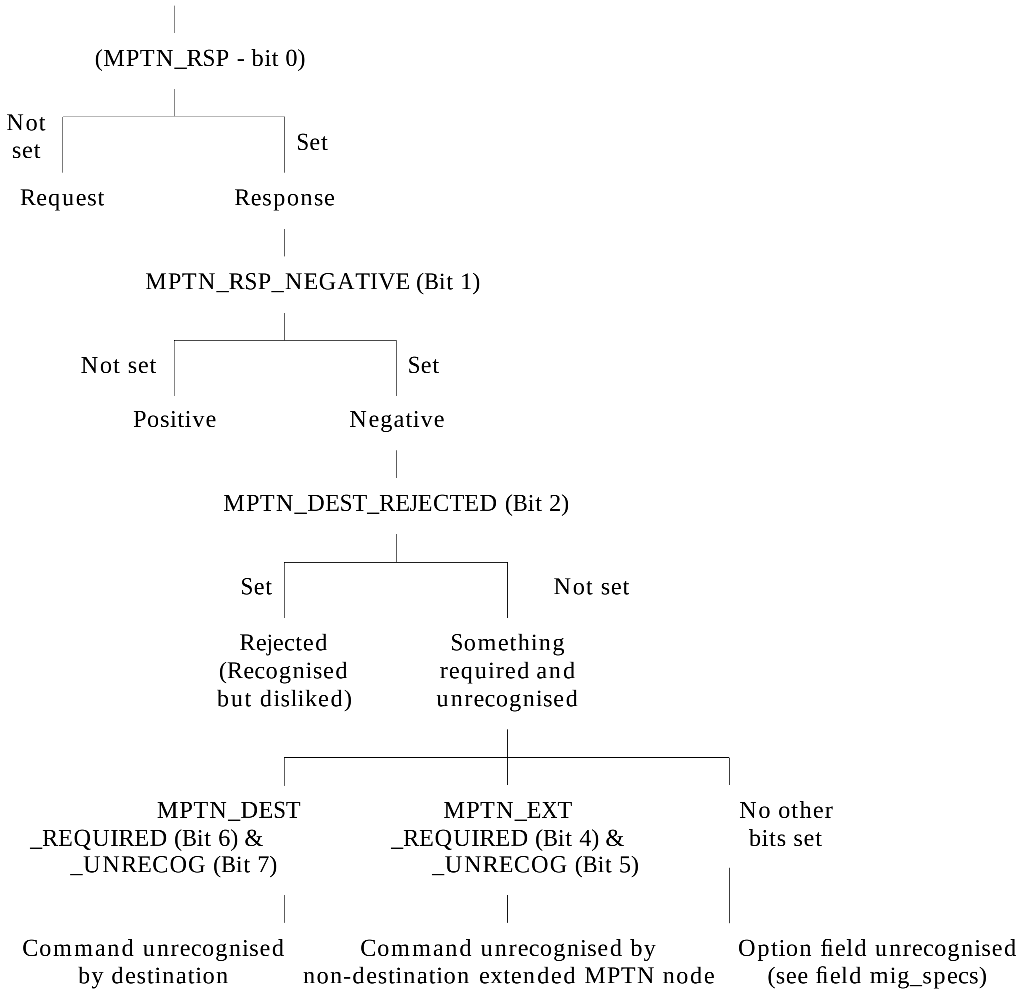

(MPTN_RSP - bit 0)
Not set
Set
Request
Response
MPTN_RSP_NEGATIVE (Bit 1)
Not set
Set
Positive
Negative
MPTN_DEST_REJECTED (Bit 2)
Set
Not set
Rejected (Recognised but disliked)
Something required and unrecognised
MPTN_DEST _REQUIRED (Bit 6) & _UNRECOG (Bit 7)
MPTN_EXT _REQUIRED (Bit 4) & _UNRECOG (Bit 5)
No other bits set
Command unrecognised by destination
Command unrecognised by non-destination extended MPTN node
Option field unrecognised (see field mig_specs)

Figure 4-1 Command Analysis Based on Processing Specification Field

4.1.3 Processing Rules

The following rules govern the behaviour of both up-level and back-level nodes in order to make orderly migration possible.

1. Every optional field and every command contains a one-byte processing specification field that specifies the actions that should be taken by either an access node or a non-destination extended MPTN node that does not recognise the command or field type. See Table 4-1 for the layout of this byte field.

2. A destination node that fails to recognise a command type or optional field tag uses the processing specification value to choose an action based on the rules specified in Table 4-2 on page 29.

3. Commands are processed before optional fields. If the command is valid, then the setting of the processing specification relies solely on the result of any optional field processing.
4. All optional fields should be processed, unless an error has already been encountered.
5. It is possible that future format changes may require multiple copies of the same optional field, while a back-level node may understand only a single instance of the field. If this happens, the receiver uses the first instance of the field and treats any additional copies of the field that it does not expect as unrecognised fields, that is, it should ignore or fail them as specified in their processing specification fields.
6. Certain required fields and optional fields (those having an overall length count) may be extended in the future by adding additional information to the end of the existing format. This may be done if and only if the success of the command does not depend on the user being able to process the additional information. For example, the user transport requirements field (see the **MPTN Data Formats** Specification (Section 3.2, MPTN_Connect)) may be extended with additional user characteristics that a back-level destination access node does not recognise and can ignore. In this case, a destination access node should not fail a command because the field is longer than expected. When the destination access node returns such a field in a reply, the following processing applies:
   — When the destination access node recognises the additional information it must echo it back.
   — When the destination access node does not recognise the additional information, it is required that the destination access node drop the part not recognised. In this case, it cannot be assumed that if the additional information is returned, it is understood.
7. Reserved fields are set to zero by the sender and ignored by the receiver. This allows future use of these bits without breaking a back-level node.

Table 4-2 shows the actions taken by a destination that does not recognise a field or command.

Command or Field	**Processing Specification**	**Action**
Command	not MPTN_DEST_REQUIRED	Discard command
Command	MPTN_DEST_REQUIRED	Set following bits in command processing spec: MPTN_RSP MPTN_RSP_NEGATIVE MPTN_DEST_UNRECOG Set Time To Live (TTL) field to initial value. Optionally add diagnostics field if the command is recognised. Return command to source.
Field	not MPTN_DEST_REQUIRED	Set following bit in field processing spec: MPTN_DEST_UNRECOG Proceed with rest of command processing.
Field	MPTN_DEST_REQUIRED	Set following bits in command processing spec: MPTN_RSP MPTN_RSP_NEGATIVE Set following bit in field mig_spec: MPTN_DEST_UNRECOG Set TTL field to initial value Optionally add diagnostics field if the command is recognised. Proceed with command processing.

Table 4-2 Processing Spec: Actions for Unrecognised Commands / Fields

Chapter 5

Initialisation

5.1 Transport Endpoints

Before a transport user can communicate with other matching transport users, it must make itself known and describe its transport requirements to the CMM. The information about the transport user is recorded in a control block, a data structure which serves as a *transport endpoint*. In order to be accessible to the network, a transport endpoint must have an address associated with it.

An MPTN control block is created when the transport user wishes to:

- initiate a connection request
- listen for a connection request
- send or receive datagrams.

In all cases, the transport user creates the control block by issuing an M_CREATE_DC. When accepting a connection request, additional data may be learned when the PMM issues the P_ACCEPT_UC (see Figure 6-6 on page 52).

When creating the control block, the CMM must have information related to the type of communication that the control block will be used for. This information includes the address family (see below), the transport characteristics required, and any restrictions on transport providers that apply. This information is maintained by the CMM throughout the life of the control block.

The M_CREATE_DC specifies the *address family* that the transport user will use with this control block. This refers to the format of the address that the transport user's application will register and expect. For example, if the transport user was running an FTP application, the format used would be that of an IP address.

In addition, the transport user has the ability to restrict which transport providers may be used with a particular control block. The transport user may select one particular transport provider, indicate a preference order for available transport providers, or indicate that any available transport provider will do. A transport user may prefer a particular transport provider based on service related criteria (cost, reliability, and so on) or on some policy based criteria (for example, use transport provider A only if you are using application B).

Details about the transport services requested are also provided by the transport user:

- *connection-oriented or connectionless*

  The type of transport service that must be provided. For example, a transport user supporting NetBIOS may choose to be registered with both connection-oriented and connectionless transport providers. Or, a sockets transport user supporting a stream socket will choose a connection-oriented service, but when supporting a datagram socket will choose connectionless service.

- For connections:

  — *stream or record*

The data type requested. If the request is for record data, then whether or not the passing of partial records is permitted must be specified.

— *termination types*

The type of termination procedure required by the transport user. The two characteristics of importance are whether it is *simplex* or *duplex*, and whether it is *abortive* or *orderly*. See Section 6.5 on page 71 for more details.

— *expedited data*

Whether or not the transport user needs to send expedited data. See Section 6.3.2 on page 63.

— *expedited marking*

Whether or not the transport user requires the ability to mark the position in the normal data where the expedited data is sent.

The relationship between data structures that provide the above information when a control block is created is illustrated in Figure 5-1. The *req* parameter in the M_CREATE_DC is a data structure of type M_INFO. The M_INFO structure contains within it the CONN_CHARS data structure. These structures are described in Appendix A.

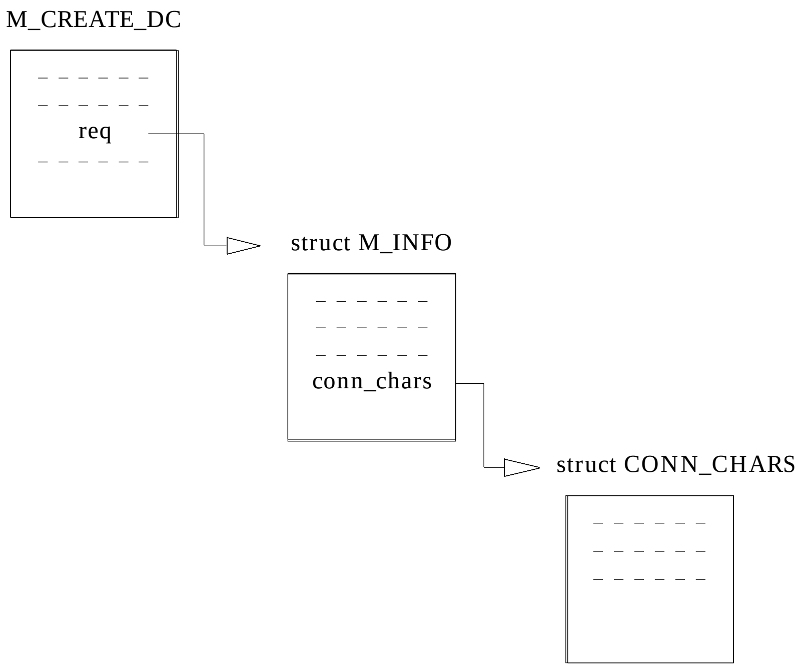


Figure 5-1 Relationship Between Data Structures in M_CREATE_DC

5.2 Address Binding and Registration

Once the transport user has created a control block, it must associate it with an address by which it is known to the network. This association is achieved when the transport user issues the M_BIND_DC with the following information:

- The transport user's address to be registered.
- An indication of whether or not the address should be registered with an external directory server or MPTN address mapper. Some transport users may wish to only initiate connections or datagrams, and as such would not require that their address be registered.
- Optional user data — this is data that is to be registered; that is, stored, along with the transport user address.

There are three specific address registration mechanisms:

- MPTN address mapper
- native directory service
- algorithmic mapping.

The PMM informs the CMM about which address mapping mechanism it supports on the P_INIT_UC. Note that in a network with access nodes supporting different address registration and resolution methods, only those nodes using the same mechanisms can dynamically locate one another. In this case, a node may support more than one mechanism to ensure connectivity.

5.2.1 MPTN Address Mapper

Figure 5-2 shows the case where an MPTN address mapper is used for registration. In this case, the CMM communicates with the address mapper and is responsible for registration. The responsibility lies with the CMM since this method is common to all transport providers.

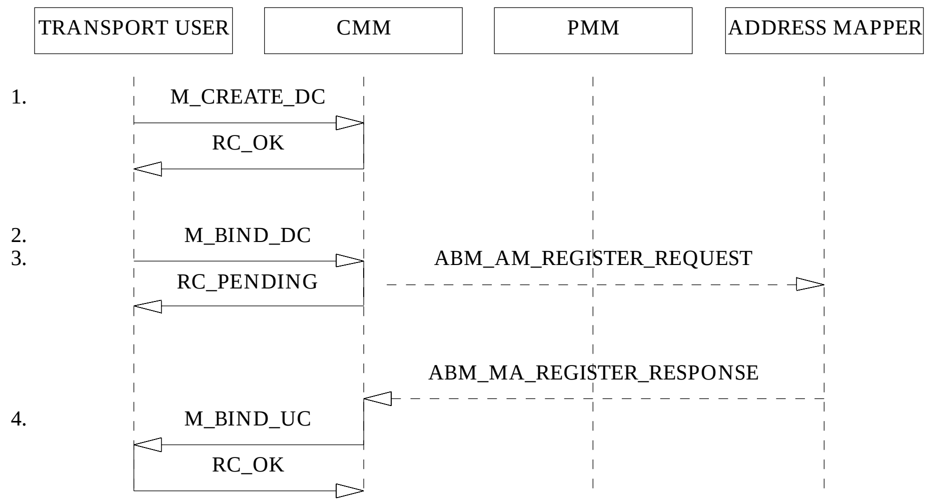


Figure 5-2 Initialisation, Registration to an MPTN Address Mapper

Annotations

2 To initiate registration, the transport user issues an M_BIND_DC to the CMM, including the transport user's address and an indication that external registration is desired. The call returns an RC_PENDING, since external flows must be generated.

3 The CMM then requests that the address mapper register the address of the transport user. Details of the CMM to address mapper interactions are available in the MPTN Address Mapper reference.

4 Upon completion of external registration attempt, an M_BIND_UC will be issued by the CMM indicating the status of the bind attempt.

5.2.2 Native Directory Service

In the native directory service case, the PMM is responsible for registering the transport user's address, since the procedure is protocol dependent.

Figure 5-3 shows the case where a native directory service is used for registration.

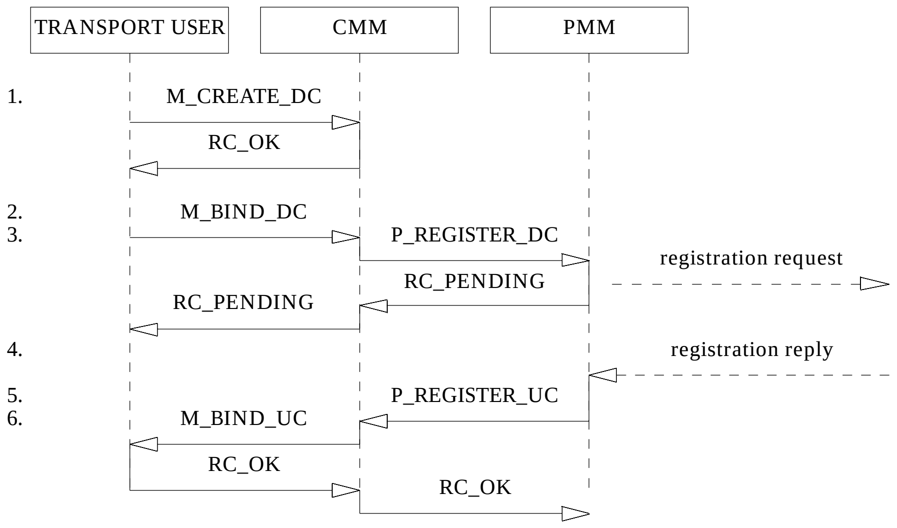


Figure 5-3 Initialisation, Registration to a Native Directory Service

Annotations

2 After the transport user issues the M_BIND_DC, the CMM issues a P_REGISTER_DC with the address of the registering transport user

3 If required, the PMM then initiates a protocol specific registration procedure. In some cases, as with the TCP/IP Domain Name Service, the external line flows (registration request/reply) will not be present since dynamic address registration is not supported and no *pending* is needed.

5 When the protocol specific registration is completed, the PMM issues a P_REGISTER_UC to the CMM indicating the status of the registration and, if the registration was successful, the identity of the transport provider used for registration.

6 Upon receiving the P_REGISTER_UC, the CMM issues an M_BIND_UC to the transport user, indicating the status of the bind attempt.

5.2.3 Algorithmic Mapping

In the algorithmic case, Figure 5-4, the mapping function resides in the PMM. Using this method, no dynamic registration with an address service occurs.

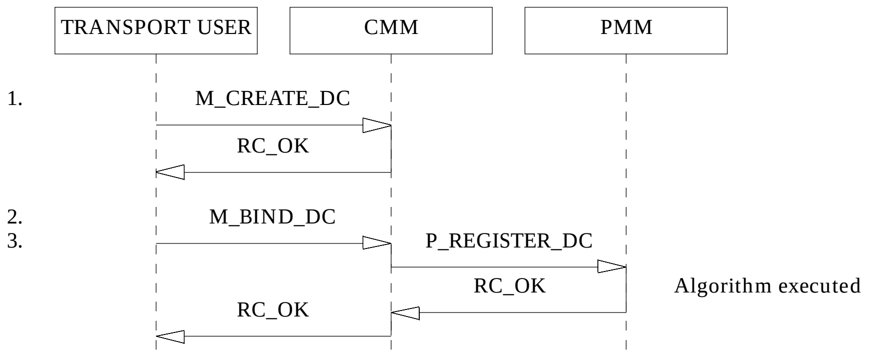


Figure 5-4 Initialisation, Algorithmic Mapping

Annotations

2-3 The transport user issues an M_BIND_DC to the CMM. The CMM passes the transport user address to the PMM via a P_REGISTER_DC command. PMM performs the algorithm and once the algorithm is done, the PMM has a mapping between a transport user and a transport provider address.

5.2.4 Joining a Multicast Group

This section describes the procedure for joining a group. It is assumed that if a join is issued and the group does not exist, the natural result will be that the group will be created with a single member, the transport user that issued the join. If the group exists, the transport user simply becomes a member of the group.

The general procedure for joining a multicast group starts with a request from the CMM to resolve the transport user group address to a transport provider group address for a particular transport provider type.

5.2.4.1 Address Binding using an Address Mapper

In the preferred case when an Address Mapper is being used, groups are created dynamically, and may be deleted dynamically. Thus, it is important to keep a reference count of how many active group members are in each group registered with the Address Mapper. This allows the clean up of group entries when they have no members.

An Access Node joins a group by issuing an M_BIND_DC. This generates an ABM_AM_REGISTER_REQUEST message to the Address Mapper. This request contains the transport user group address and a NULL value for the transport provider group address.

There are two situations at this stage, depending on whether the provider is capable of inherent multicasting or not.

- If the provider is multicast-capable:

  The Address Mapper checks to see if a Transport Provider group name has been registered for this Transport User group name. If it has, the corresponding Transport Provider group name is returned. If not, the Address Mapper will request that a Transport Provider group name be generated/created. Once the Transport Provider group address has been created, it is returned to the Access Node. See Figure 5-5 for details.

- If the provider is not multicast-capable:

  The return address from the mapping should be the Multicast Server's provider address. Then the CMM sends an MCS_AC_JOIN_REQUEST to the Multicast Server, supplying the transport user group address and unique CMM identifier. The unique CMM identifier is an MPTN qualified name that is unique to this node. If the group does not exist, then the MCS_AC_JOIN_REQUEST should be interpreted as both a create group and a join group request. Therefore, the Multicast Server will create a new entry for the group, if it did not previously exist.

  For each Transport User group, the Multicast Server maintains a list of members, that is, a list of unique CMM identifiers.

  See Figure 5-6 on page 38 and Figure 5-7 on page 39 for details.

5.2.4.2 Alternate Address Binding

If an alternate method of address binding is used, then we may not always get notification whether a group exists or not. For example, consider that a provider-specific directory is used to map a user group address to a provider group address. At address resolution time, the existence of a mapping does not guarantee that the group exists. Hence, when the PMM receives a P_JOIN_GROUP_DC with a provider group address, it is not clear whether the group has to be created or not. If this fails, then the group first has to be created.

The above dealt with the situation when the provider was capable of inherent multicast. If the user to provider group address mapping yields a Multicast Server address, then flows proceed as before. The only difference is that there are no flows between Access Node and Address Mapper.

5.2.4.3 Protocol Description

Case 1:

Case 1 is when a non-native inherent multicast is done.

The ABM_AM_REGISTER_REQUEST attempt from the Access Node to the Address Mapper yields a positive or negative response, indicating that the group either exists or that it does not, respectively. If it does exist, the transport provider group address is returned to the Access Node. The PMM uses this transport provider group address to perform the inherent multicast join operation.

A negative response indicates that the Address Mapper did not have a mapping to a Transport Provider group name, and was unable to create a Transport Provider group name.

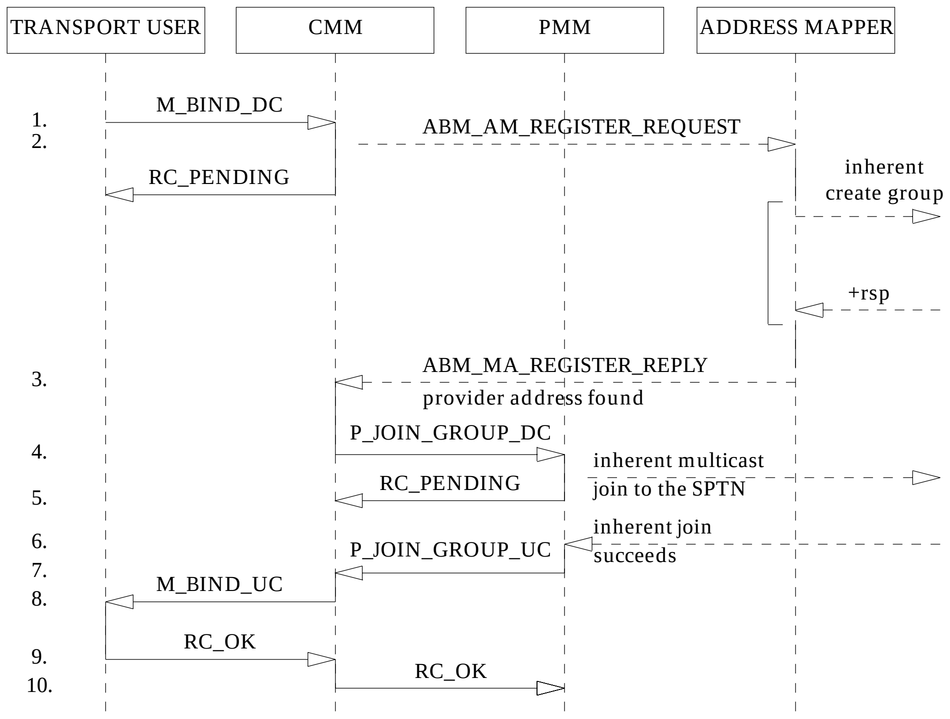


Figure 5-5 Join Flows for Inherent Multicasting: Inherent Join

Annotations

1 The Transport User issues an M_BIND_DC, providing the transport user group address.

2-3 The CMM determines the provider group address by issuing an ABM_AM_REGISTER_REQUEST, giving the transport user group address.

- If the Address Mapper has a provider group address registered for this user group address, it is returned on the response of the ABM_AM_REGISTER.
- If the Address Mapper does not have a provider group address registered for this user group address, the Address Mapper attempts to create a provider group address. If successful, the Address Mapper registers the mapping and returns the provider group address on the response.

The figure shows address resolution performed through an Address Mapper. If some other means (algorithmic, provider specific database lookup) is used, then the resolution would be through a P_REGISTER_DC from the CMM to the PMM (see Section 5.2.2 on page 34 and Section 5.2.3 on page 35 for details).

4 The CMM issues a P_JOIN_GROUP_DC with the transport provider group address. This is a request for the PMM to do an inherent join of the provider's group.

5 A return code of pending is returned to the CMM while the inherent join is being performed.

6-7 On receiving notification of the successful join to the group on the SPTN, the PMM makes a P_JOIN_GROUP_UC call to let the CMM know.

8-10 The CMM returns to the transport user indications of the successful bind, and the processing completes successfully.

Case 2

Case 2 is when the Multicast Server is used to join a group

The first step in this case still has to be to locate the transport provider group address for the multicast group. However, since the provider is not capable of multicasting, the address returned is always the provider address for the Multicast Server (if it is up). Whether the group exists or not, the Address Mapper will return the Multicast Server's transport provider group address. The CMM at the Access Node will then send an MCS_AC_JOIN_REQUEST message to the Multicast Server, where the unique CMM of the Access Node is added to the membership list of the group. If the group does not exist, the Multicast Server will create a membership list for this user group, add the unique CMM identifier for the Access Node to the membership list, and register the new user group address with the Address Mapper.

The Multicast Server keeps a reference count; that is, a count of the number of members in a group. When a new group is created, the Multicast Server registers the group with the Address Mapper and sets the reference count to 1. When a new member joins the group, the reference count is incremented, and, when a member quits the group the reference count is decremented. When the reference count becomes 0, the Multicast Server will deregister, via M_UNBIND_DC, the group name with the Address Mapper.

When a Transport Provider uses the Multicast Server to support multicast, the CMM in the Access Node must register its unique CMM identifier with the Address Mapper. This is done as part of initialisation of the Transport Provider. This needs to be done so that the Multicast Server is able to obtain transport provider addresses for each entry in the membership list.

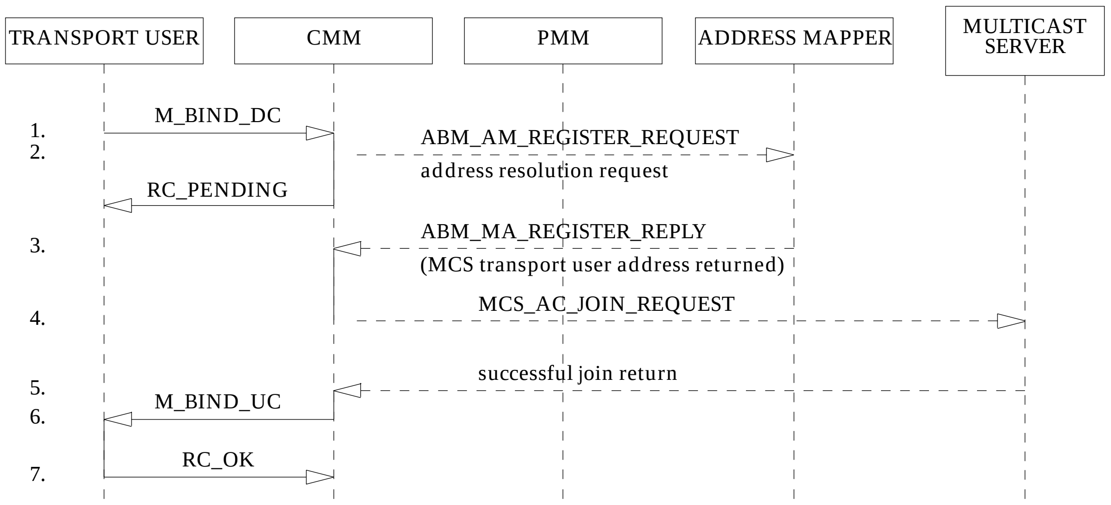


Figure 5-6 Join Flows for Multicast Server: Group Already Registered

Annotations

1 The Transport User issues an M_BIND_DC, providing the transport user group address.

2-3 The CMM first tries to determine the provider group address by issuing an ABM_AM_REGISTER_REQUEST, giving the transport user group address.

The figure shows address resolution performed through an Address Mapper. If some other means (algorithmic, provider specific database lookup) is used, then the mapping should always return the Multicast Server transport provider address.

4 The CMM issues an MCS_AC_JOIN_REQUEST to the Multicast Server. The Multicast Server adds the new member to the list it maintains for this transport user group, and increments its reference count.

5-7 On receiving notification of the successful join to the group at the Multicast Server, the CMM issues an M_BIND_UC to inform the Transport User of the successful join.

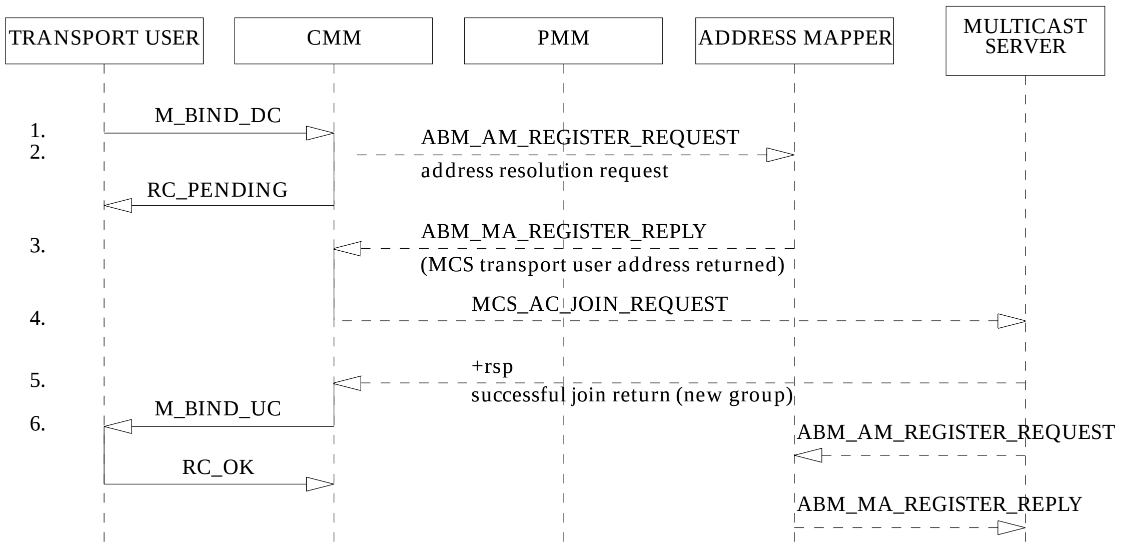


Figure 5-7 Join Group for Multicast Server: Group Not Previously Registered

Annotations

1 The Transport User issues an M_BIND_DC, providing the transport user group address.

2-3 The CMM first tries to determine the provider group address by issuing an ABM_AM_REGISTER_REQUEST, giving the transport user group address.

The figure shows address resolution performed through an Address Mapper. If some other means (algorithmic, provider specific database lookup) is used, then the mapping should always return the transport provider address of the Multicast Server.

4-5 The CMM sends an MCS_AC_JOIN_REQUEST to the Multicast Server. Since this is a new group, the Multicast Server creates a list of members for the group, adds this member to the list, sets the reference count to 1, and registers the new Transport User group with the Address Mapper.

6 On notification of a successful join, the CMM in the Access Node notifies the Transport User, via an M_BIND_UC, that the bind has completed successfully.

When the Multicast Server realises that this is a new group, it issues an M_CREATE_DC and M_BIND_DC to register the new group with the Address Mapper. This is done so locates for this Transport User group will succeed. This registration is done after the MCS_AC_JOIN_REQUEST to assure that the group exists and the first member has joined, before locates will succeed.

5.2.5 Registration Errors

If the address registration failed for some reason, the CMM will return a negative status code in the M_BIND_UC, or the return from the M_BIND_DC. Possible errors are:

- Duplicate address — the address given has already been registered.
- Directory service not available — either the MPTN address mapper or native directory server is not currently communicating with the access node.

The control block is not destroyed upon failure of a bind attempt. Rather, it remains available for subsequent bind attempts until an address is successfully bound or until an M_CLOSE_DC is issued by the transport user to destroy it.

5.2.6 Deregistration

If the transport user wishes to deregister an address from a control block, it may do so by using the M_UNBIND_DC. As in the registration case, the protocol flows depend on which type of address registration is in use, and the type of address (individual or group).

Figure 5-8 illustrates the case in which native directory is used, and the address is an individual (not a group name).

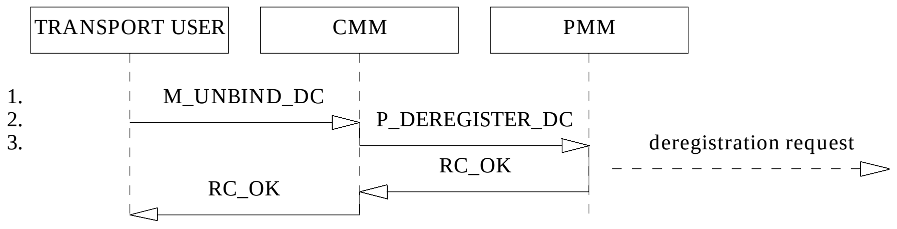


Figure 5-8 Deregistration to a Native Directory Service

Annotations

1-2 The transport user issues the M_UNBIND_DC to the CMM, which in turn issues a P_DEREGISTER_DC to the PMM, including the address of the transport user to be deregistered.

3 The PMM then initiates the native deregistration process of the transport provider.

The protocol flow for the case of an individual address with an address mapper used for registration is illustrated in Figure 5-9.

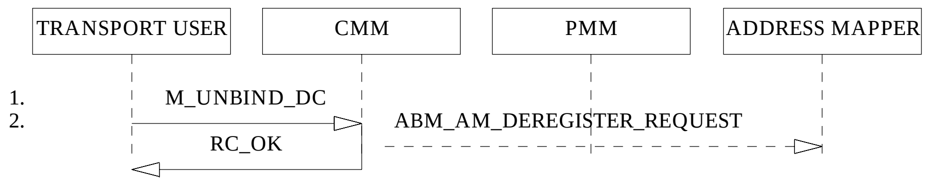


Figure 5-9 Deregistration to an MPTN Address Mapper

Annotations

1-2 The transport user simply issues an M_UNBIND_DC to the CMM, which then sends a deregistration request to the address mapper.

Deregistration of a group address is similar to deregistration of an individual address, the difference is that in addition to deregistering the address from the Address Mapper (or native directory), a *quit group* must also be performed. The way this *quit group* is performed depends on whether the underlying transport provider supports inherent multicast or uses a Multicast Server.

The following figures explain how deregistration of a group address works when an Address Mapper is used.

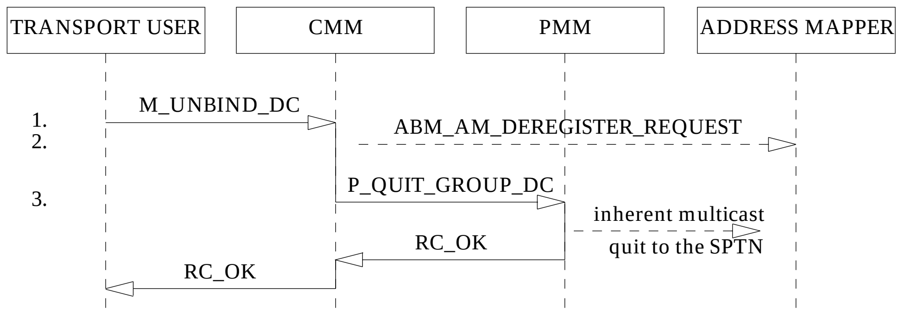


Figure 5-10 Deregistration of Group: Address Mapper & Inherent Multicast

Annotations

1-2 The transport user simply issues an M_UNBIND_DC to the CMM, which then sends a deregister request to the Address Mapper.

The Address Mapper decrements its reference count for this group, and if the reference count is 0, it also deletes the entry for the group.

3 CMM issues a P_QUIT_GROUP_DC instructing the PMM to do an inherent quit from the transport provider group.

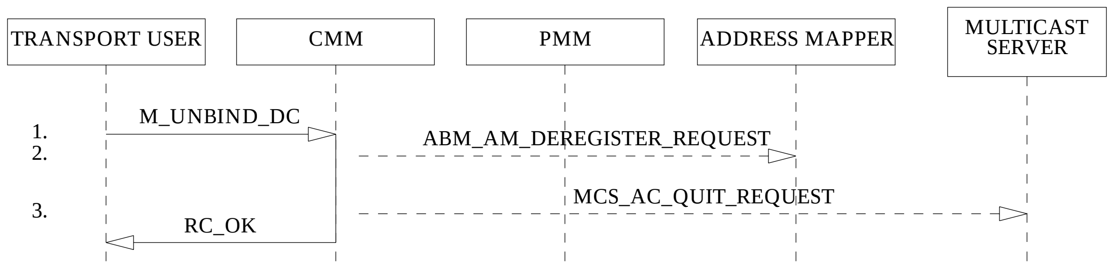


Figure 5-11 Deregistration of Group: Address Mapper & Multicast Server

Annotations

1-2 The transport user simply issues an M_UNBIND_DC to the CMM, which then sends a deregister request to the Address Mapper.

3 CMM issues a AC_QUIT_GROUP to the Multicast Server instructing the Multicast Server to delete this user from the group.

Chapter 6

Connection-oriented Protocols

MPTN supports full duplex connection-oriented service between matching transport users. A connection between a pair of transport users is associated with an MPTN control block in each MPTN node.

6.1 Connection Setup — Initiator

A transport user requests a connection by issuing an M_CONNECT_DC, specifying the following:

- The address of the matching transport user that is the target of the connection.
- The service mode of the connection that it requires; for example, a transport user might specify that it requires a specific level of security on a connection, or that the connection must provide certain delay characteristics.

Some transport users (such as an OSI or an SNA transport user) might need to send data used as part of the connection processing; it would be included as *connection data.*

The CMM is responsible for responding to transport user requests for connection establishment. To establish a connection, the following functions are performed:

- *selecting a transport provider*

  The initiating CMM is responsible for selecting the transport provider to be used on a connection. A *connection preference list* is a local mechanism that lists each installed transport provider type in preferential order. The CMM uses this list to sequentially try to resolve the target transport user's address. This list may be created based on any number of criteria such as cost, bandwidth, and so on. It may be modified by the transport user when the control block is created, by the protocol preference parameter on the M_CREATE_DC.

  There may be multiple transport provider addresses for each transport provider type. Within a transport provider type, the transport provider addresses appear in the order in which they should be tried. The CMM will select a transport provider type from the preference list, and will try each transport provider address for that type, in the order they appear, before moving on to the next transport provider type in the preference list.

  At the discretion of the local node, protocol selection can become considerably more sophisticated by comparing protocol mismatches and selecting the transport provider that provides the closest match.

- *providing transport provider caching*

  The list of transport providers may be cached, and the CMM may modify the list for use on subsequent connection attempts. For example, the cached list may be saved with the transport provider used on the previous connection at the top of the list.

- *providing address resolution*

  The CMM attempts to resolve the target transport user's address using one of the following methods:

  — *algorithmic*

The PMM generates a target transport provider address based on the target transport users address. In this case, only one address will be returned.

— *native directory*

The PMM uses the transport provider's native address resolution mechanism. In this case more than one address may be returned.

— *address mapper*

The CMM communicates with an MPTN address mapper to obtain the address mapping. In this case, more than one address may be returned. Note that when an address mapper is used, the transport provider selection and address resolution functions are merged somewhat to the extent that the address mapper supports multiple protocols.

- *providing address caching*

  The CMM may keep an address cache of destination addresses that have been previously resolved, in order to eliminate address resolution overhead on subsequent connection attempts.

There are two types of entries in this cache:

1. Normal cache entries that are kept as long as possible; that is, until the cache fills up and they are aged out.
2. Limited use entries. These are entries that are to be used only a certain number of times, then they are removed from the cache.

If the MPTN Connection fails and the address mapping was done via an address mapper, the CMM may have to send an ABM_AM_Not_Found_Request to the address mapper. Refer to the address mapper document for details.

6.1.1 Address Resolution

Depending on the type of address resolution implemented, either the CMM does the resolution itself (in the address mapper case), or the PMM does it (the native directory and algorithmic cases).

Figure 6-1 illustrates the address resolution process for the case where the access node is using a native directory.

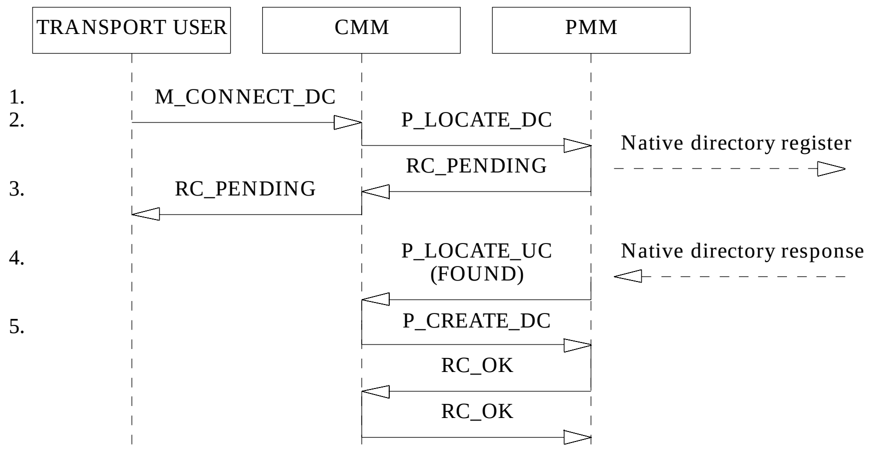


Figure 6-1 Address Resolution — Native Directory

Annotations

1-3 After the M_CONNECT_DC has been issued by the transport user, the CMM will issue a P_LOCATE_DC to the PMM (*2.*) to request that the PMM provide the mapping between the target transport user's address and its corresponding transport provider address. The information specified in the P_LOCATE_DC is:

- The target transport user's address as specified in the M_CONNECT_DC.
- The service mode as specified in the M_CONNECT_DC.
- A pointer to the MPTN control block, for identifying the connection request on the P_LOCATE_UC.

The return code for both downcalls will be RC_PENDING, indicating that the PMM is attempting to resolve the target transport user's address using the transport provider protocol specific mechanism (*3*).

4 Once the address has been found, the PMM issues a P_LOCATE_UC to the CMM containing the following information:

- A successful status code, indicating that the address has been resolved.
- The target transport user's address.
- A list of addresses of the target transport provider found. More than one address may be found depending on the transport provider. For example, a multi-homed IP node will return multiple addresses.
- An indication of which local transport provider responded to the locate.

5 At the arrival of the P_LOCATE_UC, and before returning from the P_LOCATE_UC, the CMM issues a P_CREATE_DC to the PMM to instruct the PMM to create a control block that will serve as the transport endpoint identifier for the native connection.

The CMM specifies the following in the P_CREATE_DC:

- The type of transport service requested by the transport provider, in this case connection-oriented.

- A pointer to the MPTN control block previously created by the CMM.
- Information about the transport user's requirements for this transport provider.

The case of address resolution using an algorithmic method is illustrated in Figure 6-2.

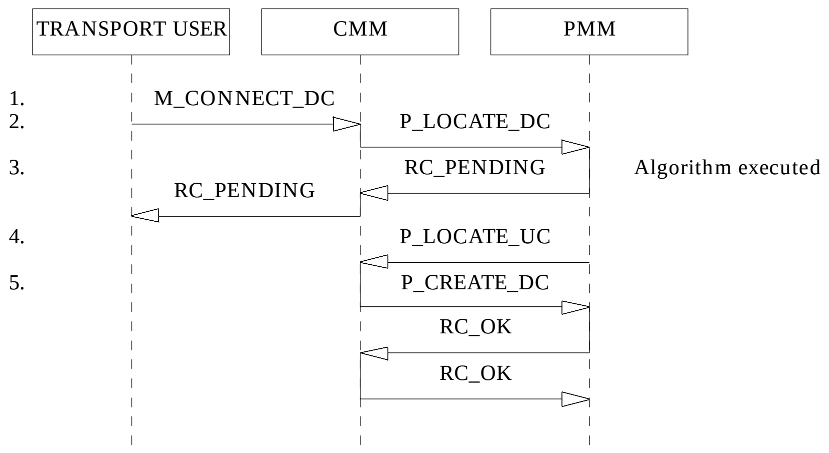


Figure 6-2 Address Resolution — Algorithmic

Annotations

1-4 The flows are similar to those in the native directory case above (Figure 6-1 on page 44), except that in this case, flows external to the access node do not occur. Instead, when the CMM issues the P_LOCATE_DC, the PMM executes the address mapping algorithm to produce the matching transport provider's address. That address is then included in the P_LOCATE_UC issued to the CMM.

The third case is that in which the address resolution is accomplished by using an MPTN address mapper, as shown in Figure 6-3. Here, as in the address registration case (see Section 5.2.1 on page 33), the CMM is responsible for communicating with the address mapper. Again, the address resolution process may be merged somewhat with the transport provider selection process when using the address mapper.

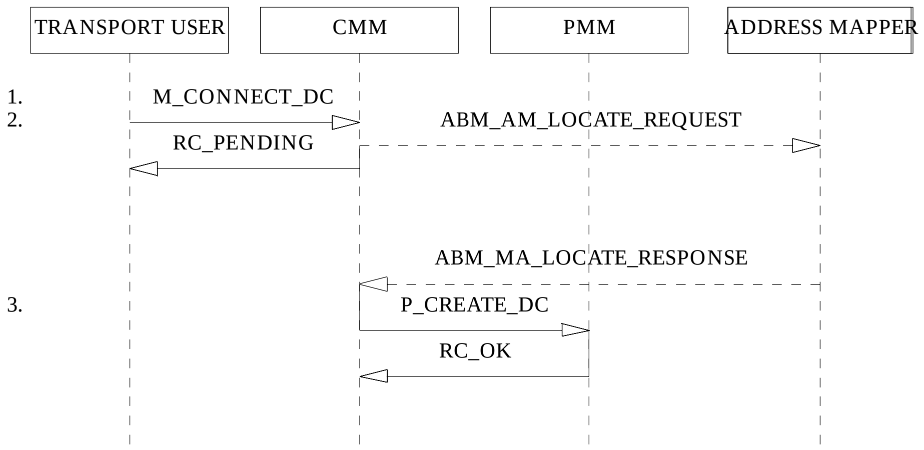


Figure 6-3 Address Resolution — MPTN Address Mapper

Annotations

1-2 After the M_CONNECT_DC is issued by the transport user the CMM uses the address mapper to resolve the address. CMM then selects which transport to use from the list of transport provider addresses returned by the Address Mapper. See the MPTN address mapper reference for further information.

3 Once the address has been resolved, the CMM issues a P_CREATE_DC to the PMM, as in the previous two cases.

6.1.2 Connection Establishment

Figure 6-4 illustrates the complete connection establishment flows for the connection initiator, regardless of which address resolution method is used.

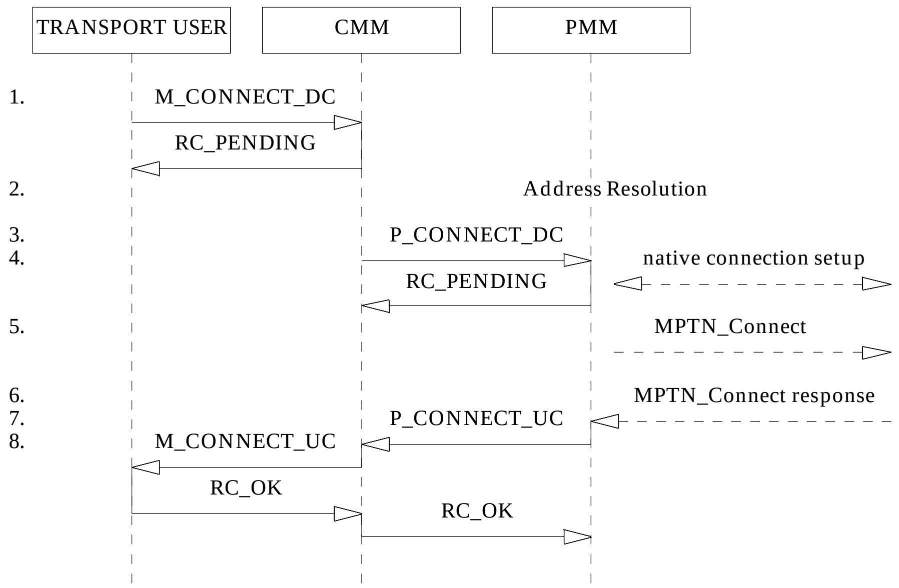


Figure 6-4 Normal Connection Setup, Address Not Cached — Initiator

Annotations

1-4 Once the address has been resolved (*2.*), the CMM issues a P_CONNECT_DC (*3.*) to the PMM with the following information:

- The addresses of both the initiating and target transport providers.
- The service mode requested for the connection.
- The MPTN compensation headers required for the connection.

  See Section 6.1.4 for more information.
- The MPTN_Connect command that is to be sent to the target PMM over the native connection.

  The format contains, among other things, the transport requirements of the requesting transport user:

  — The maximum lengths of record, expedited or termination data

  — the termination type supported

  — whether or not the transport user requires the position of expedited data flowing in the normal data stream to be marked.

  See the **MPTN Data Formats** Specification (Section 3.2, MPTN_Connect) for details of the format.
- If required, the connection correlator required for expedited data, which is a concatenation of the initiating transport user's address and a correlator unique to this

connection. This value is used to correlate between expedited data sent on the connection, and that sent in MPTN_DG_OOB_Data correlator.

PMM will establish a transport provider connection. This is the underlying transport connection that the MPTN Connection will be established on.

5-6 The MPTN_Connect and MPTN_Connect response are exchanged. This establishes the MPTN Connection between the transport users.

7-8 Once the MPTN Connection is established, the M_CONNECT_UC upcall is made to the transport user to inform it that the requested connect has been established.

The CMM has the ability to streamline the connection process on subsequent attempts by storing the address mapping of the target transport user address and which transport provider was used to connect to it. This is referred to as caching. Before looking at the next phase of connection processing (see Section 6.1.5 on page 50), we should examine the address cached case.

6.1.3 Target Address Cached

If the CMM uses an address cache, it may already have the identity of the transport provider and address mapping used to reach a particular target transport user. As such, it can skip the address resolution process.

This case is illustrated in Figure 6-5.

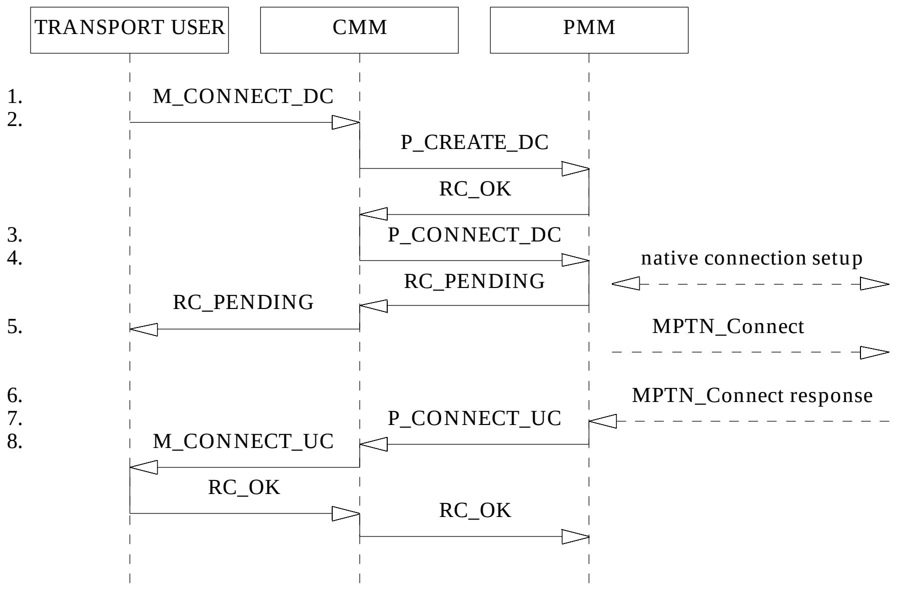


Figure 6-5 Normal Connection Setup, Address cached — Initiator

Annotations

3 In this case, the CMM immediately issues a P_CONNECT_DC to the PMM with the same information as in the address not cached case (Figure 6-4).

6.1.4 Selecting Compensations

When the P_CONNECT_DC was issued by the CMM, it contained the compensation headers required for the connection. These headers are selected by the CMM to account for mismatches in the requirements of the transport user and the native capabilities of the transport provider. The CMM knows the requirements of the initiating transport user from the information stored in the control block. It also knows the capabilities of the transport provider from installation information. Depending on the sophistication of the CMM, this information could be hard-coded, or it could be the result of some handshaking between the CMM and PMM at power-up time.

The *compensation package* for a specific (user, provider) pair is computed from a universal set of compensations, and need not be predefined for specific pairs of mismatched protocols. Therefore the addition of either a new transport protocol type or a new user type generally requires only the normal analysis and selection from the compensations previously defined.

The compensation selection for most functions is quite straight forward; either the function exists or not. The selection of expedited compensations requires the additional consideration of length, and whether or not the transport provider overwrites or concatenates additional expedited data. In the case of termination, consideration must be given to the fact that multiple types of termination may need to be supported and that transport users often have rules about which types are permitted to follow one another. These compensation issues are discussed more thoroughly in Section 6.3 on page 61.

6.1.5 Exchanging MPTN_Connect Commands

On receipt of P_CONNECT_DC, the PMM can initiate the native transport provider connection (*4.* in Figure 6-4 on page 48 and Figure 6-5 on page 49). If the transport provider uses a half duplex mechanism, such as LU 6.2 half duplex conversations, and the transport user requires a full duplex connection, the PMM will set up two half duplex connections.

The normal connection establishment mechanism for the transport provider protocol occurs, with the target transport address that is used being a combination of the target node address from the address resolution process (see Section 5.2 on page 33.) and the *well-known local address*. This well-known local address is a special local address for MPTN connections that is known by all access nodes supporting a particular transport provider. It provides a common mechanism by which any access node can request an MPTN connection from any other access node using the native transport provider protocol. It is similar to the TCP/IP concept of a well-known port.

The currently designated values for well-known local addresses are:

TCP	397 (decimal)
UDP	397 (decimal)
SNA	X’28 F0 F0 F1’
SPX	8795 (hexadecimal)
IPX	8796 (hexadecimal)

After the native connection has been established, the two nodes must exchange an MPTN_Connect command and response in order to set up an MPTN connection. The initiating node sends the MPTN_Connect command, including addressing information, connection characteristics, and any connection data required. See the **MPTN Data Formats** Specification (Chapter 3, Connection Establishment Format) and **MPTN Data Formats** Specification (Chapter 8, Formats Common to Multiple Messages) for details of the format.

After processing, the target node will send an MPTN_Connect response (*6*) to the initiating node. This response may include some changes to the connection characteristics requested by the target transport user. The negotiable connection characteristics are:

- maximum record length
- maximum expedited record length
- maximum length of connection termination data.

These three connection characteristics may be negotiated down. The characteristics sent in the MPTN_Connect response will be the ones used on the connection.

In the MPTN_Connect command and response, there is a *correlator suffix* field. This correlator suffix is assigned by the CMM, and when combined with the destination and source user addresses must be unique for all time. This correlator field is also used by the PMM in the MPTN_DG_OOB_Data and MPTN_Cntl_Datagram commands.

Once the positive response is received, the PMM issues a P_CONNECT_UC to the CMM, which includes:

- an indication that the MPTN Connect processing was successful
- the MPTN_Connect response command sent by the target node
- the address of the initiating transport provider.

Upon receiving the P_CONNECT_UC, the CMM changes its state to *connected*. At this point the CMM then issues an M_CONNECT_UC to the transport user, as the result call for the transport users original M_CONNECT_DC. The upcall contains the following:

- the status of the connection processing
- the connection reply data, if any
- the set of connection characteristics provided by the target transport user.

6.2 Connection Setup — Target

The connection setup procedures for the target of the connection is illustrated in Figure 6-6.

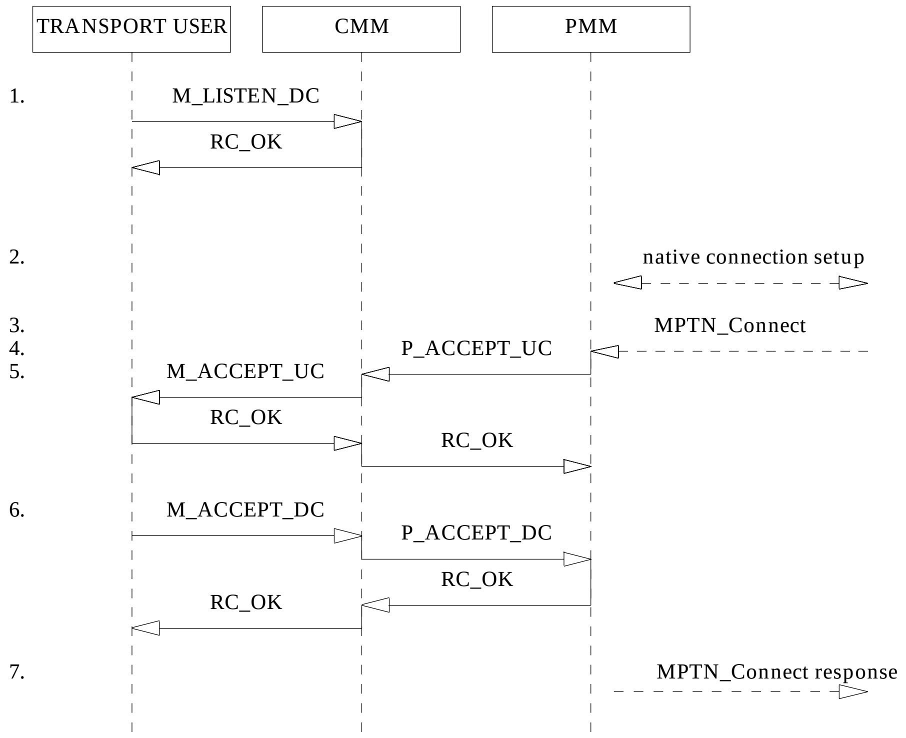


Figure 6-6 Normal Connection Setup — Target

Annotations

1 In order to accept incoming connection attempts from remote partners, the transport user must listen for connections by issuing an M_LISTEN_DC to the CMM. The transport user will specify in the verb whether it wants to listen for a single connection or multiple connections. If the transport user wishes to receive only a single connection attempt, then the listen will be canceled after the first connection attempt. If multiple connection attempts is chosen, then the listen will stay up until explicitly canceled by the transport user.

2-4 The PMM will receive a native connection attempt, and will participate in the setup. When the MPTN_Connect arrives from the initiating node, the PMM issues a P_ACCEPT_UC to the CMM with the following information:

- The MPTN_Connect command that arrived on the transport provider's well-known local address.
- The initiator (source) and target (destination) transport provider addresses.
- An indication of which transport provider is to be used on this connection.
- A PMM connection identifier. The connection identifier is created by the PMM, and the necessary information is passed between the PMM and CMM in the P_ACCEPT_UC.

5 The M_ACCEPT_UC is sent to the transport user by the CMM to indicate the arrival of the connection request. The following information is provided to the transport user:

- The connection data sent by the initiator, if any.
- The address of both the initiator (source) and target (destination) transport users.
- The service mode requested by the initiating transport user.
- The CMM connection identifier implicitly created for this connection.
- The connection characteristics requested for the connection by the initiating transport user. See Section A.2 on page 197 for details about these characteristics.

6 When the connection reply is generated, the transport user responds and either accepts or rejects the connection request with a corresponding M_ACCEPT_DC, providing the following information to the CMM:

- Status field indicating whether the transport user accepts or rejects the requested connection.
- The modified connection characteristics.

  The target transport user has the option of reducing the value of selected connection characteristics supplied by the initiating transport user.
- The connection reply data, if any.

The P_ACCEPT_DC returns the following information to the PMM:

- Status field indicating whether the transport user accepts or rejects the requested connection.
- The MPTN compensation headers required for the connection. See Section 6.1.4 on page 50 for details.
- The MPTN_Connect response command to be returned. See the **MPTN Data Formats** Specification (Section 3.2, MPTN_Connect) for details about how the response is structured.
- The connection correlator, which is used for expedited data transfers. See Section 6.3.2 on page 63.

7 PMM sends the MPTN_Connect response to the initiating transport user.

6.2.1 Compensation for Full Duplex Connections

The underlying transport provider connection is assumed to be a full duplex connection. If the connection that is established is half duplex, then compensation is done to simulate a full duplex connection. This compensation is accomplished by establishing twin-opposed half duplex connections (that is, one connection is used as the send pipe, the other is used as the receive pipe).

When the connection target PMM receives a half duplex connection request at its well-known MPTN address, the target node *knows* it must establish the second half duplex connection. The target node will receive the MPTN_CONNECT on the first half duplex connection, and will send the MPTN_CONNECT response on the second half duplex connection.

There is enough information in the MPTN_CONNECT and MPTN_CONNECT response for the initiator node to be able to associate the first and second half duplex connections. If the Transport User supports connect/connect, this is done by matching the destination user address and source user address on the MPTN_CONNECT, which goes out on one connection, with the same fields on an incoming MPTN_CONNECT response on a different connection. If the Transport User does not support connect/connect, this is done by matching the destination user address, source user address, and correlator suffix tuple on the MPTN_CONNECT, which goes out on one connection, with the same tuple on an incoming MPTN_CONNECT response on a different connection.

6.2.2 Connect/Connect

The following section describes how connect/connect is handled. Connect/connect is only applicable if parallel connections are not supported.

Connect/connect is when both Transport Users issue connect requests, and neither one has a listen posted. How this situation is handled depends on the user characteristic *connect/connect*. This user characteristic is passed to CMM on M_CREATE_DC. It is part of the M_INFO structure (see Section A.3 on page 198 for details).

If the transport user does not support connect/connect, a listen must be posted before a connection can be established. If the transport user does support connect/connect, the following figures and text explain how the connection is established.

There are two flavours of connect/connect. The first is where the CMM gets a incoming MPTN Connection request with no listen having been issued. CMM holds the request for a period, and during the period the transport user issues an M_CONNECT_DC. This is referred to as a Dual Connect. Figure 6-7 on page 56 and Figure 6-8 on page 57 describe a standard and optimised protocol.

The second flavour is where both CMMs issue MPTN connection requests such that the requests cross. This is referred to as a Crossing Connect. Figure 6-9 on page 59 describes this protocol.

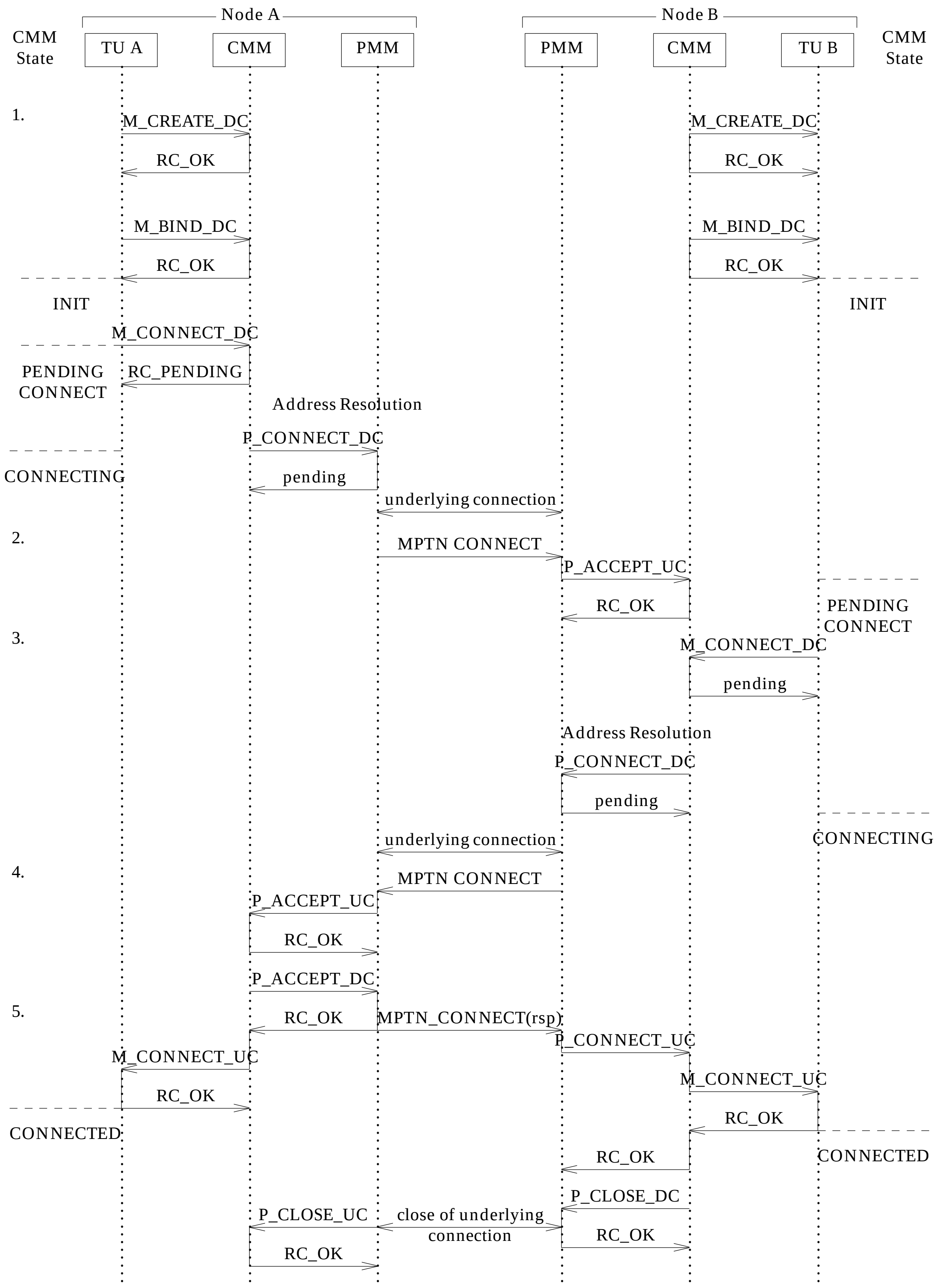


Figure 6-7 Dual Connect

Referring to Figure 6-7:

Annotations

1 Both sides issue commands to create communication end-points, and to bind a transport user address to that end-point. When the end-point is created and the M_BIND_DC has been processed, the CMM goes into *INIT* state.

The user characteristic *connect/connect*, which is part of the M_INFO structure passed on M_CREATE_DC tells whether the transport user supports connect/connect. It is assumed here that the transport user supports connect/connect.

When CMM in Node A receives the M_CONNECT_DC, the CMM goes into *PENDING CONNECT* state.

When CMM in Node A issues the P_CONNECT_DC, the CMM goes into *CONNECTING* state.

2 When the MPTN_CONNECT is received at Node B, and the P_ACCEPT_UC is issued to CMM, the requested partner has a connection end-point, but no listen has yet been issued. If the transport user supports connect/connect (as in this case) the connection request will be held on to, giving the local transport user a chance to issue an M_LISTEN_DC or M_CONNECT_DC.

The CMM in Node B puts the transport users connection endpoint into *PENDING CONNECT* state.

3 The Transport User B issues an M_CONNECT_DC while the CMM is in the *PENDING CONNECT* state. An optimisation which is described in Figure 6-8 on page 57 could be done at this point, but it is not shown in this example.

CMM knows at this point that it is in a connect/connect scenario. In fact, it knows this is a dual connect scenario. CMM resolves the destination address, builds the MPTN_CONNECT, and issues the P_CONNECT_DC. When building the MPTN_CONNECT, the CMM must take the negotiable user/transport characteristics into account from the incoming MPTN_CONNECT and adjust them properly.

4 When the MPTN_CONNECT is received at Node A, the P_ACCEPT_UC is issued to CMM, CMM treats this as if it were the MPTN_CONNECT(rsp) for the previous request but does not issue a M_CONNECT_UC yet.

5 CMM also treats this MPTN_CONNECT as a request for an MPTN connection and builds and send an MPTN_CONNECT(rsp) to Node B.

At this point, two MPTN connections are up, although the transport users have not been notified. An algorithm is executed by both CMMs to determine which MPTN connection is closed, and which is kept up.

This algorithm consist of putting the two IP addresses in host order and performing an unsigned compare. If the addresses are the same, the port numbers are compared. The low address loses and must bring down the underlying connection that it initiated.

If the transport provider is performing FDX Simulation, the second connection will not be brought down. It is used to complete the FDX Simulation compensation.

The transport users are then informed about the connection that will be kept.

6 The extra MPTN_Connection is taken down.

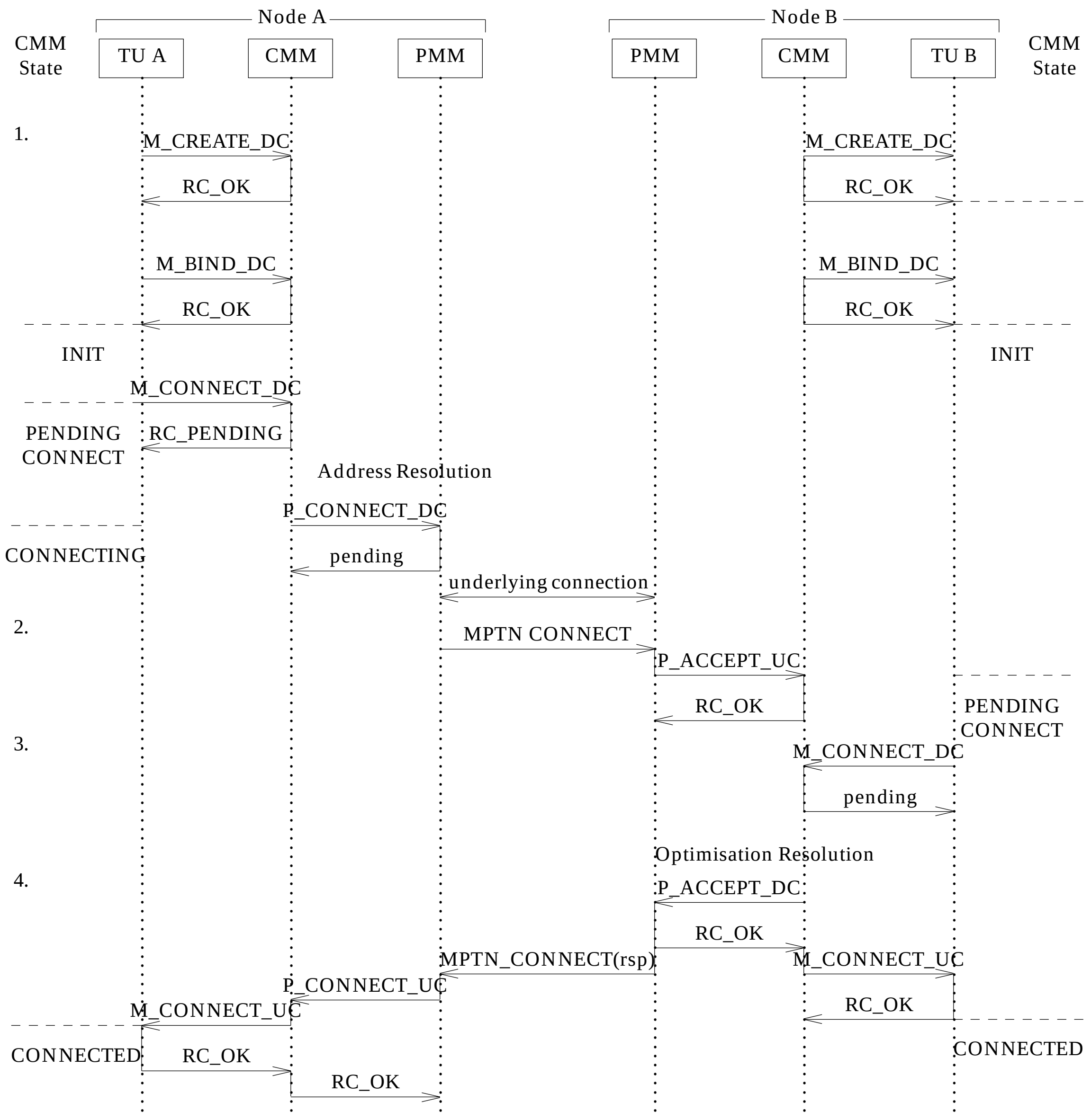


Figure 6-8 Dual Connect: Optimised Version

Referring to Figure 6-8:

Annotations

1 Both sides issue commands to create communication end-points, and to bind a transport user address to that end-point. When the end-point is created and the M_BIND_DC has been processed, the CMM goes into *INIT* state.

The user characteristic *connect/connect*, which is part of the M_INFO structure passed on M_CREATE_DC tells whether the transport user support connect/connect. It is assumed here that the transport user supports connect/connect.

When CMM in Node A receives the M_CONNECT_DC, CMM goes into *PENDING CONNECT* state.

When CMM in Node A issues the P_CONNECT_DC, the CMM goes into *CONNECTING* state.

2 When the MPTN_CONNECT is received at Node B and the P_ACCEPT_UC is issued to CMM, the requested partner has a connection end-point, but no listen has yet been issued. If the transport user supports connect/connect the connection request will be held on to give the local transport user a chance to issue an M_LISTEN_DC or M_CONNECT_DC.

The CMM in Node B puts the transport users connection endpoint into *PENDING CONNECT* state.

3 The Transport User B issues an M_CONNECT_DC while in the *PENDING CONNECT* state. This figure shows an optimisation of the protocol described in Figure 6-7 on page 56.

CMM knows at this point that it is in a connect/connect scenario. In fact, it knows it is in a dual connect scenario. CMM treats the M_CONNECT_DC as if it were an M_LISTEN_DC. CMM can resolve the destination address by getting the associated transport user address from it's connection endpoint control block.

From Node A's point of view, this appears to be a non-connect/connect scenario. Node A, however, must be able to support the connect/connect protocol as shown in Figure 6-7 on page 56.

4 CMM builds the MPTN_CONNECT(rsp) and sends it to the partner via an P_ACCEPT_DC.

Note: The CMM at Node A responds back to the transport user with an M_CONNECT_UC.

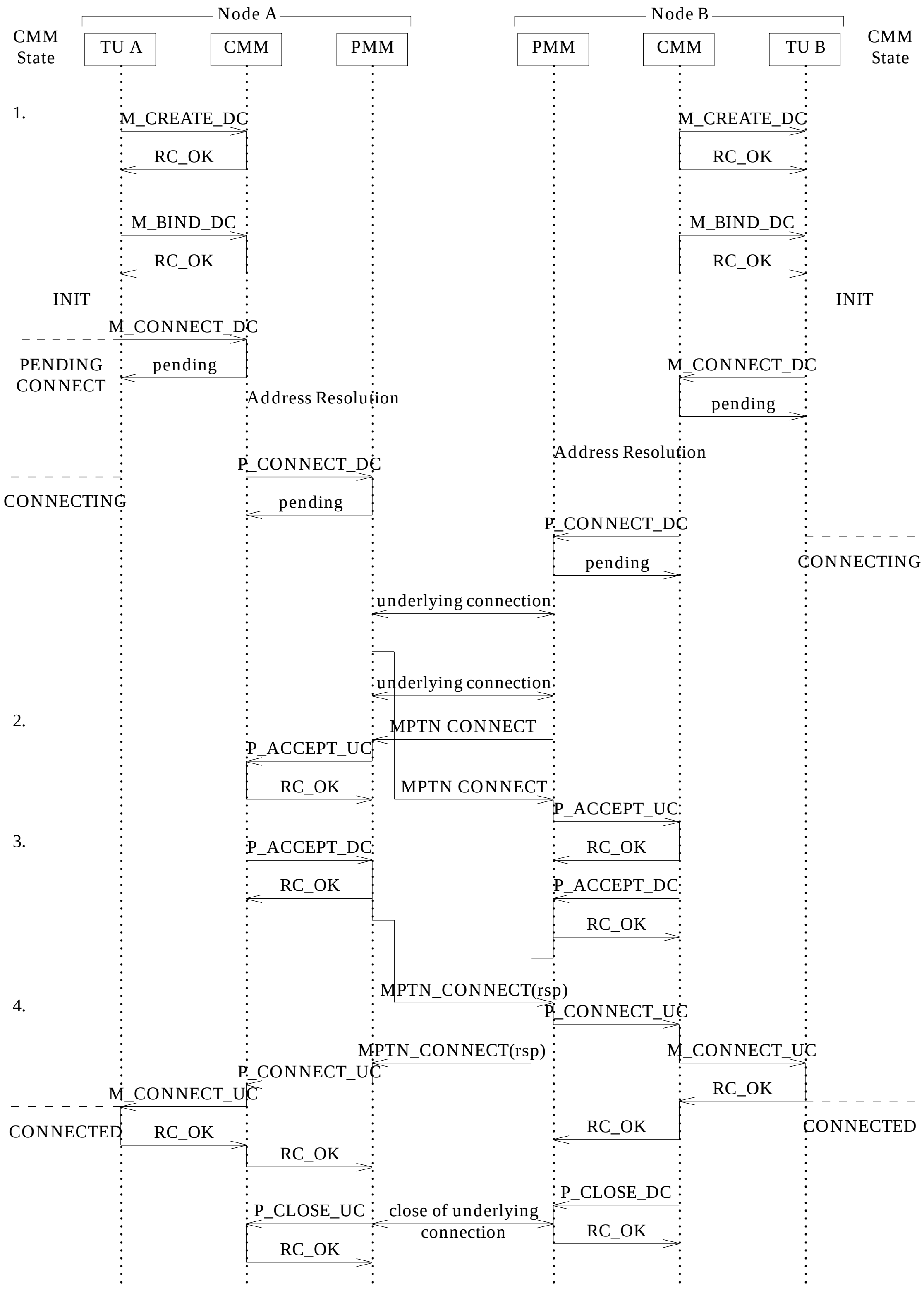


Figure 6-9 Dual Connect: Crossing Connections

Referring to Figure 6-9:

Annotations

1 Both sides issue commands to create communication end-points, and to bind a transport user address to that end-point. When the end-point is created and the M_BIND_DC has been processed, the CMM goes into *INIT* state.

The user characteristic *connect/connect*, which is part of the M_INFO structure passed on M_CREATE_DC tells whether the transport user support connect/connect. It is assumed here that the transport user supports connect/connect.

In this case, both CMMs are in *INIT* state when they receive the M_CONNECT_DC. Both CMMs change to *PENDING CONNECT* state, resolve the destination address mapping, build the MPTN_CONNECT and issue a P_CONNECT_DC.

At this point, the CMMs change to *CONNECTING* state.

2 Each CMM will receive an P_ACCEPT_UC while in *CONNECTING* state (normally, CMM would receive a P_CONNECT_UC), thus the CMM knows it is in a connect/connect scenario, in fact, it knows that it is in a crossing connect scenario.

CMM does not issue a M_ACCEPT_UC in this case.

3 CMM will build an MPTN_CONNECT(rsp) and issue a P_ACCEPT_DC. The MPTN_CONNECT(rsp) is sent to the partner.

4 When the MPTN_CONNECT(rsp) is received by CMM, there are essentially two MPTN connections that have been established, however the transport users only wanted one. Both CMMs will execute an algorithm to determine which MPTN connection will be kept and which will be taken down. Both sides can issue the M_CONNECT_UC to the transport user and the extra MPTN connection is taken down.

This algorithm consists of putting the two IP addresses in host order and performing an unsigned compare. If the addresses are the same, the port numbers are compared. The low address loses and must bring down the underlying connection that it initiated.

If the transport provider is performing FDX Simulation, the second connection will not be brought down. It is used to complete the FDX Simulation compensation.

6.3 Data Transfer

The M_SEND_DC and P_SEND_DC send data on a single connection. Guaranteed, error free delivery of in sequence (first in first out) data is provided. The P_RECEIVE_UC and M_RECEIVE_UC deliver data received from a connection partner. The receive upcalls are delivered serially for a given connection to ensure data is received in the order it was sent.

Either stream or record data is supported. The initiating transport user indicates the type of service it requires when the control block is created.

Non-expedited data that is passed on multiple send verbs is delivered to the destination transport user in the same order it was passed by the initiating transport user to its CMM. Achieving this requires three elements:

- The data from multiple M_SEND_DC verbs must be passed to the transport protocol in the same order in which the verbs were issued.
- The transport protocol must maintain data sequencing (a normal characteristic of transport connections).
- The destination transport user must be able to distinguish the order in which data was sent.

If the transport user indicates in the M_CREATE_DC (in the M_INFO data structure) that partial records are supported, then a record may be passed to the CMM using multiple consecutive M_SEND_DC verbs, one for each record segment. However, note that the two matching transport users need not use the same value of *partial_records*.

When the transport user sends a partial record, it indicates this to the CMM via the *incomplete_record* parameter of the M_SEND_DC. Similarly, if partial records are supported, data may be delivered to the transport user on multiple M_RECEIVE_UC verbs. These boundaries are identified to the receiving transport user using the *incomplete_record* parameter of the M_RECEIVE.

If support for partial records is not indicated in the M_CREATE_DC, then records must be passed on a single verb crossing (M_SEND_DC or M_RECEIVE_UC).

The data from a single M_SEND_DC may be transmitted by the transport provider as a single packet or a series of smaller packets, depending on the size of the transmit buffers available. These alternatives are transparent to the initiating transport user. The segments may be received at the target node using either a single or multiple M_RECEIVE_UCs, depending on whether or not partial records are supported by the target transport user.

The PMM has the responsibility to add the appropriate compensations to the data stream, as specified by the CMM. If any compensation headers are needed, all records sent on the connection must have a header.

The elements of the compensation package for a connection are as follows:

- *Record boundaries*

  Transport users may want to preserve record boundaries when transmitting data. If the transport user requires the preservation of record boundaries and the transport provider supports stream transport, a header and record length are added to mark the record boundaries.

- *Record segmentation*

  The maximum size of a record varies across different transport protocols. If the transport user needs to be able to send longer records than the transport provider supports, then the records are segmented and a header is added to each record indicating the segmentation.

- *Expedited data*

  Some transport users, for example, TCP/IP or SNA, require the ability to send expedited data. Some protocols do not provide this ability, for example, NetBIOS. If a mismatch occurs, a header is added to each record to be expedited. In addition, an MPTN_DG_OOB_Data may also be sent. See Section 6.3.2 on page 63.

- *Correlation of expedited and normal data*

  Some TCP/IP users take advantage of that protocol's ability to maintain information about the position of the last byte of normal data when expedited data was sent. Most other transport protocols do not provide this capability. This can be accomplished since each of the expedited records is marked with a header.

- *Maximum size of expedited data*

  Different protocols allow different amounts of expedited data to be sent at one time. TCP/IP allows only 1 byte, OSI allows up to 16 bytes, and SNA allows up to 86 bytes of expedited data to be sent. If the transport user needs to send a larger record of expedited data at one time than the transport provider allows, then the records are segmented by the PMM and each segment is marked with a header.

- *Session Outage Notification (SON)*

  Some protocols require immediate notification if a session outage occurs (for example, partner's node dies or intermediate route dies). Other protocol allow a long time to elapse before the session outage will be detected. When a transport user requires immediate SON, and the transport provider does not provide that function, an MPTN compensation is performed. This compensation is described in Section 6.4 on page 68.

The semantics of M_SEND are as follows:

- A positive return code is an indication that the data will be reliably delivered to the destination (unless the connection fails); it does not imply that the destination has already received the data.
- If the PMM is requested to flush the data (via P_SEND_DC), it is transmitted as soon as possible subject to network flow control; otherwise the PMM may hold data until a minimum number of bytes are buffered to be sent or a timer expires.

6.3.1 Sending Data

Figure 6-10 shows the data transfer flows for the initiator of a normal data transfer.

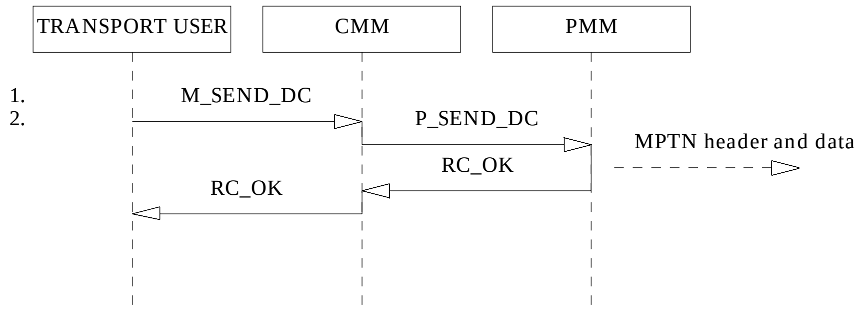


Figure 6-10 Data Transfer — Initiator

Annotations

1 For the send, the M_SEND_DC is issued to the CMM by the transport user with the following information:

- The data to be sent.
- An indication of whether the data being passed is a complete record (or last segment) or an incomplete record.
- An indication as to whether or not the data to be sent is expedited data.

  If this connection was created with a datatype of record, then expedited data must be passed only on record boundaries. Thus, the transport user may not send expedited data if the previous M_SEND issued by the transport user sent partial records.
- If required, a request to the CMM to send the current data, and any data queued in the PMM, as fast as possible.

2 Upon receiving the M_SEND_DC, the CMM issues a P_SEND_DC to the PMM containing the same parameters as the M_SEND_DC. The PMM then creates the appropriate MPTN Header format and sends it and the data to the matching node over the MPTN connection. Upon sending the MPTN Header, the return values are sent for the P_SEND_DC, and the M_SEND_DC.

If the size of the record the transport user sends to CMM (and CMM forwards to PMM) is greater than the transport providers maximum record size, the PMM needs to segment the data.

6.3.2 Sending Expedited Data

Some transport users require the ability to send expedited data that fulfill two requirements:

- The expedited data must arrive before data sent later on the connection.
- The expedited data must be able to bypass normal data queues, which may be congested.

As an example, both OSI and SNA transport users have the ability to send expedited data even if the receiver is not receiving normal data.

In order to provide for these requirements, an MPTN compensation is provided that sends data on the normal connection (to meet requirement 1 above), and, if needed, as a datagram (to meet requirement 2). The receiver processes the version that arrives first, and discards the other.

Three cases are defined in order to analyse the need for sending expedited data:

- The transport provider supports expedited data, and the length of the expedited data field is sufficient to support the transport user's requirement. No compensation is required.
- The transport provider supports expedited data, but the length of the expedited data field is not sufficient to support the transport user's needs. The transport provider's treatment of subsequent expedited messages then needs to be considered:

  — Subsequent expedited sends cause the preceding expedited data to be lost if it is not received fast enough. In this case, a header with a length field is added before the expedited data. As much of the composite packet is sent as expedited data as allowed by the underlying transport (for example, one in TCP/IP), and the rest is sent in the normal data stream, preceded by an MPTN header. The receiving PMM receives urgent notification about the header, and flushes the normal data stream to receive the rest of the expedited data.

— Subsequent expedited sends do not cause preceding expedited data to be lost, but assures in-order, loss-free reception. In this case, a header with a length field is added before the expedited data. The composite packet is segmented into packets whose sizes are no greater than the maximum amount of expedited data allowed by the underlying transport, and each segment is sent as expedited data. Segments after the first one do not contain headers.

- The transport provider doesn't support expedited data, and it cannot guarantee that congestion problems will not delay the normal data flow. The expedited data header is added and the format is sent in the normal data flow. The expedited data requires an acknowledgement, and if it is not received within an appropriate time, then an out-of-band datagram (that is, MPTN_DG_OOB_Data) is also sent. See Figure 6-11.

As shown in Figure 6-11, the expedited send initially proceeds in the same manner as a normal send, except that both the M_SEND_DC and the P_SEND_DC are marked as expedited.

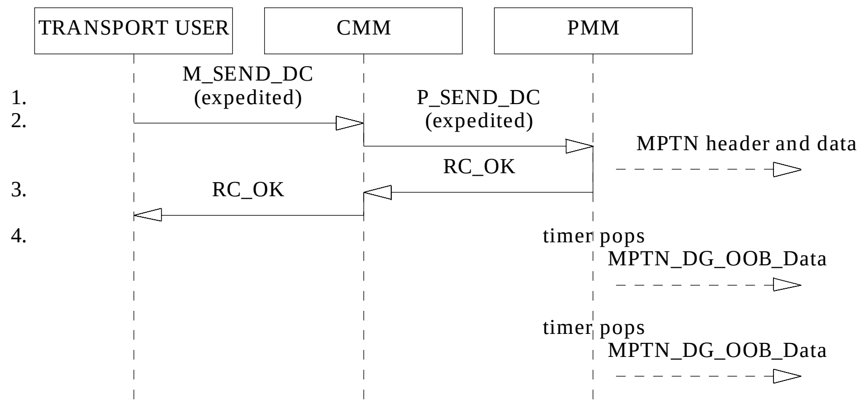


Figure 6-11 Expedited Data Transfer — Initiator

Annotations

1-2 Both the M_Send_DC and P_Send_DC are marked as containing expedited data.

3 Initially, the data is sent in a normal data stream with an MPTN Header, and an acknowledgement timer is set. If an MPTN_Header is received indicating acknowledgement of the expedited MPTN Header, then no other action is taken and the send resembles a normal send.

4 However, if the timer expires, the PMM will then send the MPTN_OOB_DG_Data format using the transport provider's normal datagram service. Conventional retry logic is applied to the MPTN_DG_OOB_Data format if no acknowledgement is received. This process will continue until either:

- An acknowledgement is received either as a datagram or on the connection.

  Note: There are a number of scenarios in which the initiator of the datagram will receive multiple acknowledgements.

- The retry limit is exceeded, at which point the matching transport user is considered to be inoperative and the connection is closed.

In addition to the data being sent, the MPTN_DG_OOB_Data format includes a connection correlator used to identify the associated connection, and a connection sequence number to associate the in-line data and the datagrams that repeat it. The connection correlator consists of two parts: the correlator address, which is the address of the initiating transport user, and the correlator suffix, which is a value unique to the transport user. The two parts are concatenated to form the connection correlator. See the **MPTN Data Formats** Specification (Section 5.3, MPTN_DG_OOB_Data). for more information.

There are two ways to correlate the MPTN_DG_OOB_Data to the associated connection:

- Using the connection correlator field only.
- Using the connection correlator field and the optional receiver connection alias field.

The choice is up to the initiator of the format.

The connection correlator is recognised by both sides without any extra protocol. The receiver connection alias is a local identifier that can be used by the destination for quick identification of the connection.

In the first MPTN_DG_OOB_Data sent for the connection, the initiator provides its own local identifier (in the sender connection alias field) and the connection correlator. The target locates the connection using the connection correlator and stores the sender connection alias so that it knows the value to be returned in the receiver connection alias field in the future.

Whenever an initiator builds an MPTN_DG_OOB_Data format, it should include a receiver connection alias, if available. Inclusion of the receiver connection alias is recommended for performance reasons.

The initiator continues to include its sender connection alias until it receives an indication from the target that it has been received, either in a positive response to an MPTN_DG_OOB_Data sent, or in an MPTN_DG_OOB_Data command from the partner with the receiver connection alias containing its local id.

In order to correlate the activity on the connection with the out of band data, connection sequence numbers are used. Each MPTN_DG_OOB_Data sent on behalf of a particular connection is assigned a sequence number which identifies it as an event in a particular direction, with each direction having independent sequence numbers. These sequence numbers do not appear in the expedited data sent on the connection because the connection guarantees in-order reception; datagrams do not.

The connection sequence numbers start at zero and increment after each expedited message is sent, whether that message is on the connection or is an MPTN_DG_OOB_Data (as in the case of DA_TERM). The sequence numbers will wrap (reset to zero) at 65,535. A retransmission of a previous expedited message is not counted as a separate event.

6.3.3 Receiving Data

Figure 6-12 shows the data transfer flows for the target of a normal data transfer.

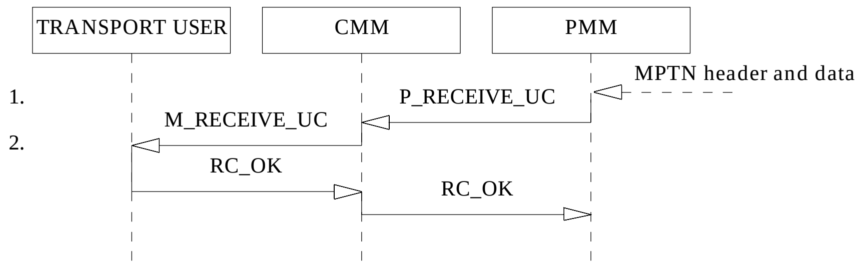


Figure 6-12 Data Transfer — Target

Annotations

1 For the receive, when the MPTN Header format is received, the PMM issues a P_RECEIVE_UC to the CMM with the following information:

- The status of the receive.
- The data received.
- If the data is marked expedited, a marker to correlate the position of the expedited data in the normal data field.
- If record data is being passed as partial records, an indication of whether or not this data completes a record.

2 Upon receiving the P_RECEIVE_UC, the CMM issues an M_RECEIVE_UC to the transport user with the same information as in the P_RECEIVE_UC.

6.3.4 Receiving Expedited Data

The target of the expedited data transfer will respond to each instance of the expedited data received with an appropriate acknowledgement, either an MPTN Header marked as an expedited acknowledgement or an MPTN_DG_OOB_Data response.

In the case of the MPTN_DG_OOB_Data response, the connection sequence number is included in the MPTN_DG_OOB_Data reply, and indicates to the initiator to stop sending retransmissions.

6.3.5 Pacing

There is no pacing protocol in the MPTN components. The transport user would typically implement some form of pacing, which may or may not be exercised in the non-native path depending on the point in the transport user from where the TLPB calls are made. The transport provider in turn will typically provide some form of pacing, which will be exercised by the PMM[2].

2. The NetBIOS stack is an exception to this, as both a transport user and provider.

The MPTN components have been designed to tie together the pacing in the transport user and the transport provider. The following is one possible technique for connecting these two together. The link between the two pacing mechanisms is implemented through a buffer sharing mechanism that is used during data transfer. This is summarised below.

- When the PMM receives data, it reads the length prefix to discover how much data follows. Then, it asks the transport user for a buffer. If a buffer is successfully allocated (it may be refused as described below), the data is read into this buffer.
- When the PMM is asked to send data with a P_SEND downcall and the data is completely sent, it returns RC_OK, which tells the transport user that it can free the buffer. If the entire data cannot be sent, the PMM queues the data and returns RC_PENDING as described above. When the data is completely sent, it must free the buffer.

The transport user is free to refuse a buffer requested if its own pacing mechanisms tell it not to accept any more incoming data. This will cause the PMM to refuse to receive the data from the transport provider stack, which will cause data to be queued inside the stack until it reaches internal queue length limits. This back pressure will cause the transport provider stack's pacing mechanism to be triggered and ultimately, data flow will stop.

Since the PMM depends on the transport user to give it buffers during inbound data processing, it must handle the situation where the transport user has no buffers to give. When the PMM does not get the transport user buffer that it requested, it must *stop receiving data* on the native transport.

In such a scheme, a mechanism must exist for the transport user to inform the PMM when a buffer is available, after the request for a buffer was refused.

6.4 MPTN Keepalive Protocol

This section describes why certain combinations of transport users and providers need to perform a node-wise keepalive protocol. The protocol is then described.

6.4.1 The Problem of Out-of-sync Connections

This problem occurs when the transport provider protocol does not provide *session outage notification* (SON) in a timely manner. For example, the problem can occur with SNA transport users over TCP/IP transport provider because even though TCP/IP provides a per connection KEEPALIVE facility, the typical values of TCP keepalive timers (the period of inactivity after which liveness checking is initiated) are much larger than SNA's SON requirements. Sockets transport users over SNA transport provider would not face this problem because socket applications do not require *immediate* SON capability. The rest of the discussion assumes that the transport user requires immediate SON capability and the transport provider does not provide that function, thus an MPTN Keepalive compensation is performed by the transport provider's PMM.

When two Access Nodes have a number of MPTN connections between them (ignoring connections that are coming up or are being taken down), one of two things can happen to cause them to have different views of which connections they have with each other:

- One of the access nodes crashes.

  This does not allow any MPTN flows (for example, connection termination) to reach the partner node to allow the MPTN connections to be taken down. None of the transport provider connections are taken down either. If there is no activity on any of the sessions, the two sides will have an inconsistent view of which MPTN connections (and therefore, which transport user sessions) they have with each other until the transport provider's KEEPALIVE mechanism kicks in. Even if there is some activity on one or more of the sessions, it may take a considerable amount of time for the transport provider to detect that a connection is broken.

- An intermediate router used by the transport provider connections between these two Access Nodes goes down for a while.

  During that period, each of the two access nodes can independently close one or more MPTN connections that they have with the partner. However, because of the dead router, neither the MPTN flows nor any transport provider connection termination flows can reach the partner. This would result in the two nodes having an inconsistent view of the connections they have with each other. This is illustrated in the flow diagram in Figure 6-13.

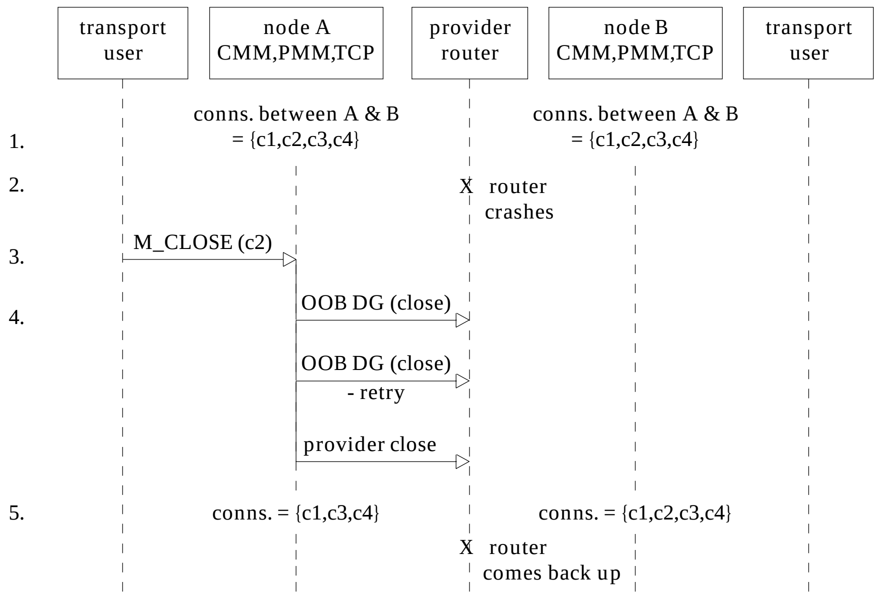


Figure 6-13 Inconsistent Views Caused by Router Crash

Annotations

1 At a certain point in (global) time, both nodes have a consistent view of what connections they have with each other, c1, c2, and so on, are MPTN connection IDs.

2 An intermediate transport provider router goes down.

3 The transport user on node A closes one of the connections.

4 The MPTN components attempt to close the transport user (that is, MPTN) connection, and then the transport provider connection. None of the packets reach node B.

5 The control block corresponding to connection c2 is deleted in node A. Nodes A and B now have inconsistent views of the connections they have with each other.

When two active Access Nodes have inconsistent views of their connections to each other, a new transport user connection setup request initiated by one node (which believes that the number of connections is within limits) may be rejected by the partner node (which believes that connection limits have already been reached).

The above situation is a direct result of the fact that while the router was down, the transport provider's keepalive mechanism did not get triggered. The assumption here is that the router went down and came back up before the transport providers keepalive timer expired. Since some transport provider's keepalive timers are typically set to an hour or more (for example, TCP), this is not an unrealistic scenario.

If the transport provider's keepalive timer were user settable, this problem could be avoided by setting the timer to a value that is smaller than the time required for a router or an end node to be restarted after a crash (for example, 2 minutes). However, most implementations do not allow this timer to be changed by the user. The alternate solution is to implement an MPTN keepalive protocol. The protocol is described in the next section.

6.4.2 A Keepalive Protocol Using Datagrams

The MPTN keepalive protocol is executed by two nodes that have one or more MPTN connections to each other. The protocol is performed if at least one of the transport user connections between the two nodes requires SON[3].

The keepalive protocol involves sending a keepalive datagram (DG) if an inactivity timer expires, and waiting for a response. The format of the keepalive DG is described in the **MPTN Data Formats** Specification (Section 5.4, MPTN_DG_KEEPALIVE_Hdr).

When the *first* MPTN connection requiring SON between a given source and destination address is set up, an inactivity timer is started for that partner (that is, node). The value of the timer should be user settable.

When any data is received from the partner, such as a connection request, data on an existing connection or any kind of a datagram (out-of-band datagram, control datagram, user datagram or keepalive datagram), the timer is reset. However, if no data is received from that partner before the above timer expires, the MPTN keepalive protocol is initiated with that partner by sending it a keepalive DG. Keepalive DGs are sent to the MPTN well-known datagram address.

Once the keepalive protocol is initiated, a keepalive DG is sent to the partner once every X seconds, where X is a value that can be configured by the user. At most 5 keepalive datagrams are sent. When a partner receives a keepalive DG, it is expected to send a keepalive DG RSP packet back to the sender. If the partner does not respond after 5 retries by sending a keepalive DG RSP or some other data, the partner is declared to be dead and all MPTN connections to that partner that requested SON are cleaned up.

The keepalive timer between a pair of addresses is initiated when the first connection requiring SON between that pair of addresses is *fully established*. Keepalive processing is stopped for a partner when the *last* connection to that partner requiring SON is being taken down. The architecture leaves open the option of using some other mechanism to deal with boundary conditions involving node or network outage when connections are being brought up or being taken down.

To cover the case where a node goes down and comes back up before the KEEPALIVE protocol is able to detect the outage, an optional node initialisation ID field is included as part of the MPTN_Connect, MPTN_Datagram, MPTN_Cntrl_Datagram and MPTN_Syntax_Mapper_Signal_Datagram. This field contains a value that uniquely identifies the instance of the sending node; for example, a time stamp corresponding to the time the node was initialised. The receiving node keeps this value, and compares it with future incoming node initialisation IDs from the same partner. If the values are different, the partner has gone down and recovered. Thus, all sessions with the partner that require SON should be cleaned up.

3. This is a transport user characteristic.

6.5 Connection Termination

When an MPTN connection is terminated, the transport user may optionally send termination data on the connection. The manner in which an MPTN connection is terminated is based on the following two characteristics:

- Whether one or both pipes of the full duplex connection are closed:

  — *simplex close*

  Only the sending pipe is closed and the end that performs the simplex close can still receive data.

  — *duplex close*

  Both the sending and receiving pipes are closed by the end that performs the duplex close, and therefore it can neither send nor receive any more data. When the termination request reaches the partner, it too disallows any sends and receives on its end of the connection.

- Whether any unsent or unreceived data is discarded when the termination indication is received by the sending and receiving ends, respectively:

  — *orderly close*

  The termination request, along with any termination data, is put on the send queue. If there is other transport user data in front of the termination request, the termination request (and termination data, if any) is only sent out after the user data. Since this flow is typically subject to pacing, the termination request may be held up in the queue for an indefinitely long period of time.

  — *abortive close*

  When an abortive close request is made by the transport user, any unsent data on the send queue for the connection is discarded. When the termination request is received by the partner, it too discards any data in its receive queue for the connection that has not yet been read by the application.

  There is another aspect of abortive close which distinguishes it from orderly close semantics. Abortive close semantics specifies that the transmission of the termination request (and the termination data, if any) is not controlled by pacing on the connection. For the purpose of providing compensation for termination when running over a non-native transport protocol, different transport users may need this ‘‘expedited’’ semantics of the termination request with different degrees of rigor. For example, adhering to the expedited semantics is critical for SNA but not for sockets transport users. See Section 6.5.3 on page 79 for an explanation of how expedited termination is handled.

  One aspect of abortive close that seems to vary from one protocol to another is whether such a termination request is acknowledged. This distinction is ignored in the MPTN termination framework.

Note: The termination request and any optional termination data must be sent in the same way.

Based on the above characterisation, a termination request can be one of four types — *simplex orderly*, *simplex abortive*, *duplex orderly* and *duplex abortive*.

Many transport users support multiple termination types. When a transport user closes an MPTN connection using a certain termination type, it can follow that by closing the connection again using a more severe termination type. The rules for repeating termination requests on a connection are:

- A simplex close can be followed by a duplex close.
- An orderly close can be followed by an abortive close.
- Both of the above can occur together, that is, a simplex orderly close can be followed by a duplex abortive close.

Figure 6-14 shows the allowable sequence of close requests that a transport user can make on a connection.

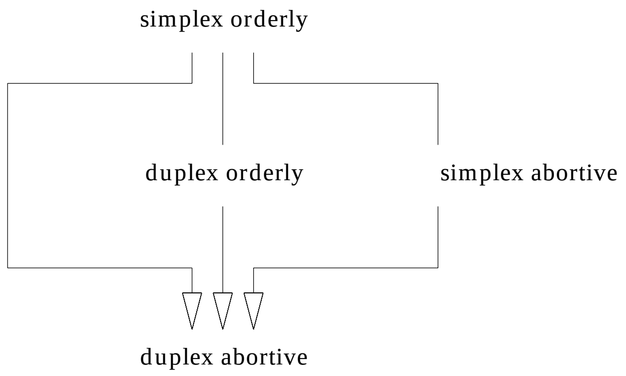


Figure 6-14 Allowable Sequence of Close Requests

For the case where there is an exact match between the combination of termination types needed and the use of termination data, no compensations are required.

6.5.1 Transport User Requires Duplex Termination

In order to describe the termination protocol, we will first examine the various possibilities for the external flows before we discuss the detailed internal flows. The external termination flows discussed below apply to the internal flows given in Figure 6-15, Figure 6-16 and Figure 6-17. The external termination flows occur as follows:

- First, an MPTN Header is sent containing the proper termination identifier (see the **MPTN Data Formats** Specification (Chapter 4, MPTN Headers) for a list of identifiers). If the transport user requires termination data, it will be sent along with this header.

  Note: If the user and provider capabilities match and no termination data is used, there may be no MPTN header — just the duplex close for the underlying transport connection.

- Then, if the transport provider termination protocol is duplex, a protocol specific duplex termination flow is sent, and the transport provider connection is considered terminated.

  Either the target or source may initiate the transport provider termination sequence, depending on the termination type supported by the provider and whether termination data is sent by the transport user. In some cases there may be an MPTN acknowledgement flow.

- On the other hand, if the transport provider termination protocol is simplex, then a protocol specific simplex termination flow is sent, and a return flow is expected. When the return flow is received, the connection is considered terminated.

Either the target or source may initiate the transport provider termination sequence, depending on the termination type supported by the provider and whether termination data is sent by the transport user. In some cases there may be an MPTN acknowledgement flow.

Figure 6-15 illustrates the case of duplex orderly termination for the transport user that initiates the termination.

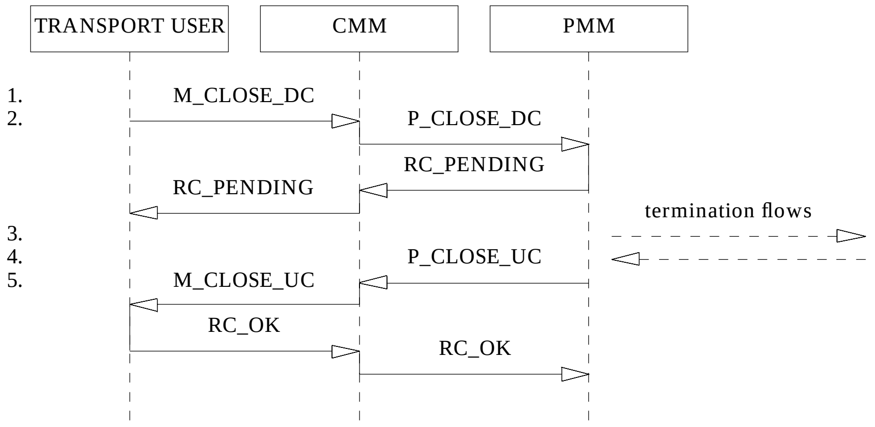


Figure 6-15 General Duplex Orderly Termination — Initiator

Annotations

1 The initiating transport user issues an M_CLOSE_DC, indicating the type of termination requested and any transport user specific termination data required.

2 Upon receiving the M_CLOSE_DC the CMM issues a P_CLOSE_DC to the PMM, with the same parameters as the M_CLOSE_DC.

3-4 Once the external termination flows have completed, the PMM issues a P_CLOSE_UC to the CMM.

5 Upon receiving the P_CLOSE_UC, the CMM issues an M_CLOSE_UC to the transport user.

Figure 6-16 illustrates the case of duplex abortive termination for the initiating transport user.

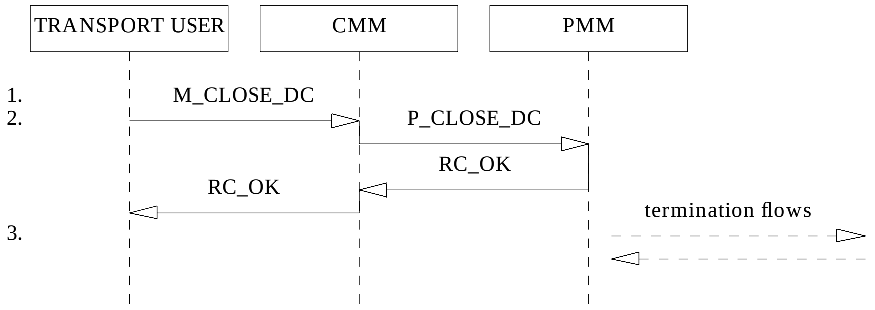


Figure 6-16 General Duplex Abortive Termination — Initiator

Annotations

1-3 The M_CLOSE_DC and P_CLOSE_DC are issued in the same way as in the duplex orderly case, except for the termination type indicated. The difference is that when the PMM returns from the P_CLOSE_DC, it may post either a positive return code or a pending return code. If a positive return code is posted, the CMM considers the connection closed and PMM will discard any queued data. If a pending return code is posted, the flow is similar to the previous *duplex orderly* flow (see Figure 6-15).

Figure 6-17 illustrates the target side of the duplex termination for both the abortive and orderly cases.

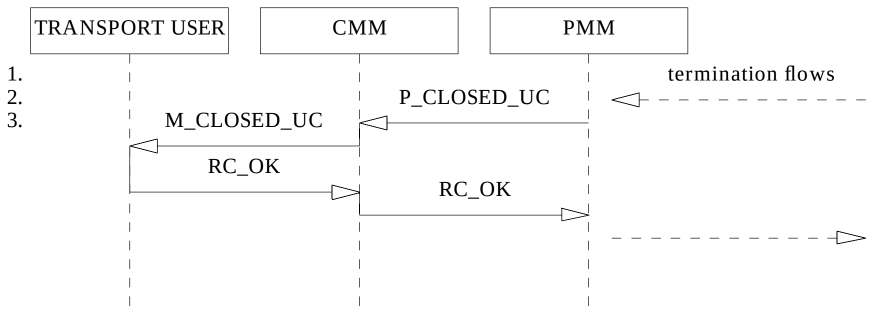


Figure 6-17 General Duplex Termination — Target

Annotations

1-2 When the termination flows are received from the initiating transport user, the PMM issues a P_CLOSED_UC to the CMM including any termination data sent by the initiator, and an indication of the type of termination requested.

3 After the CMM receives the P_CLOSED_UC, it issues an M_CLOSED_UC to the transport user, containing the information from the P_CLOSED_UC.

At this point, if the transport provider termination protocol is simplex, then the native target to initiator side of the connection is still up. The PMM then issues a native simplex termination to complete the termination of the native connection.

6.5.2 Transport User Requires Simplex Termination

As in the previous section, we will first examine the various possibilities for the external flows before we discuss the detailed internal flows. The external termination flows discussed below apply to the internal flows given in Figure 6-18, Figure 6-19, Figure 6-20 and Figure 6-21.

The external termination flows occur as follows:

- If the transport provider supports simplex termination and the transport user provides no termination data, then a protocol specific simplex termination flow is sent to the target. A protocol specific simplex termination flow is expected back from the target to close the other direction.
- If the transport provider supports simplex termination and the transport user provides termination data, then the termination data would be sent in an MPTN header flow preceding the exchange of protocol specific simplex termination flows, in either direction.

- If the transport provider supports duplex termination, then an MPTN header is sent with a simplex close indicator. Termination data is also sent, if required by the transport user. When the target transport user closes it's outgoing simplex connection, an MPTN header followed by a transport provider protocol specific duplex termination flow, is sent.

Figure 6-18 illustrates the case of simplex abortive termination for the initiating transport user.

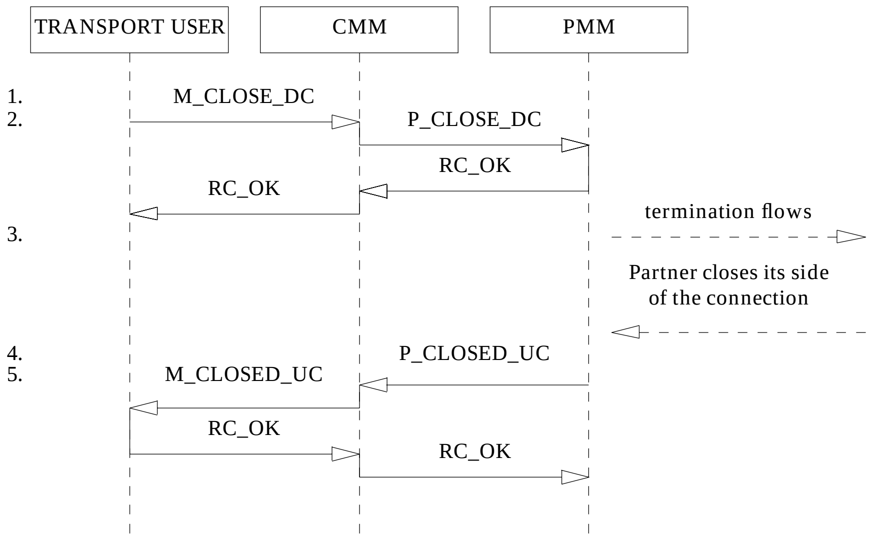


Figure 6-18 General Simplex Abortive Termination — Initiator

Annotations

1 The transport user issues an M_CLOSE_DC indicating the type of termination requested and any transport user specific termination data required.

2-3 Upon receiving the M_CLOSE_DC, the CMM issues a P_CLOSE_DC to the PMM, with the same information as in the M_CLOSE_DC.

PMM will return an RC_OK to CMM and issues a termination flow to terminate the outgoing connection.

4 When the partner closes its outgoing simplex connection and the PMM receives the command, it issues a P_CLOSED_UC to the CMM with an indication of the status of the termination. Upon receiving the P_CLOSED_UC, the CMM issues an M_CLOSED_UC to the transport user.

Figure 6-19 illustrates the case of simplex orderly termination by the initiating transport user. The issuing of the M_CLOSE_DC and P_CLOSE_DC proceeds in the same manner as in the previous simplex abortive case, with the exception of the close type indicating simplex orderly.

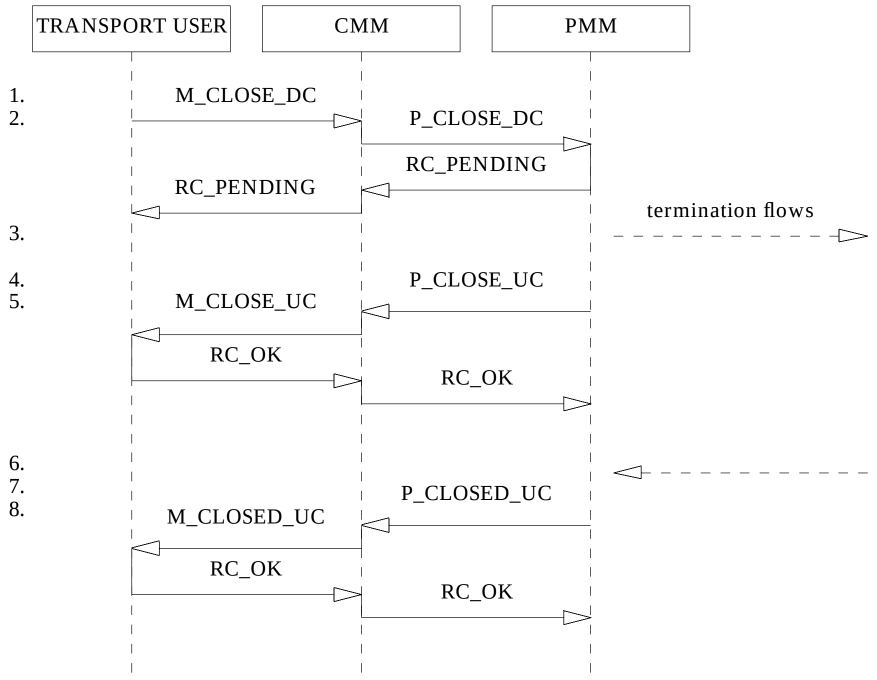


Figure 6-19 General Simplex Orderly Termination — Initiating Transport User

Annotations

4 After sending the termination flows, the PMM issues a P_CLOSE_UC to the CMM, indicating the status of the P_CLOSE_DC processing.

5 The CMM then issues an M_CLOSE_UC to the transport user, indicating the status of the M_CLOSE_DC processing.

6-8 When the partner closes its outgoing simplex connection and the PMM receives the command, it issues a P_CLOSED_UC to the CMM including the close type and any transport user specific termination data required.

Upon receiving the P_CLOSED_UC, the CMM issues an M_CLOSED_UC to the transport user, with the same information as the P_CLOSED_UC.

The case of simplex abortive termination for the target of the termination is illustrated in Figure 6-20.

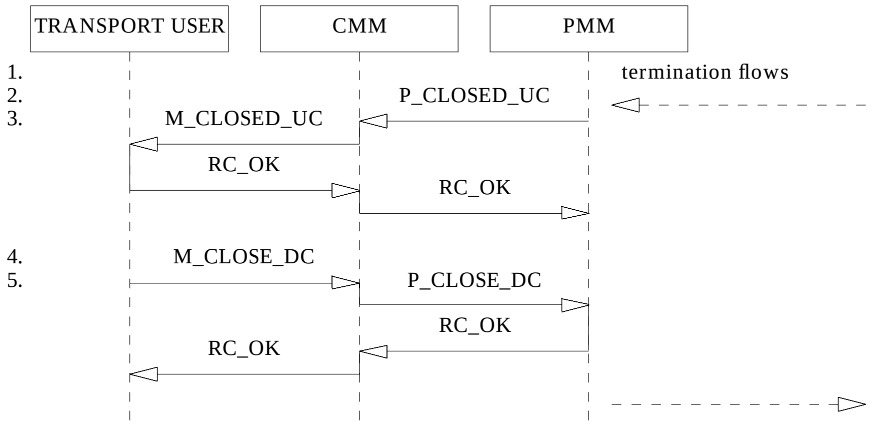


Figure 6-20 General Simplex Abortive Termination — Target

Annotations

2 Upon receiving the termination flow from the initiator, the PMM issues a P_CLOSED_UC to the CMM indicating the termination data, if any, sent by the transport user and an indicator of the type of termination requested. The CMM then issues an M_CLOSED_UC to the transport user with the same information as the P_CLOSED_UC. From the target's perspective, the initiator to target direction of the full-duplex connection is considered closed.

4 When it wishes to close its outgoing connection, the transport user then issues an M_CLOSE_DC to the CMM. Upon receiving the M_CLOSE_DC, the CMM issues a P_CLOSE_DC.

Figure 6-21 illustrates the case of simplex orderly termination for the target of the termination.

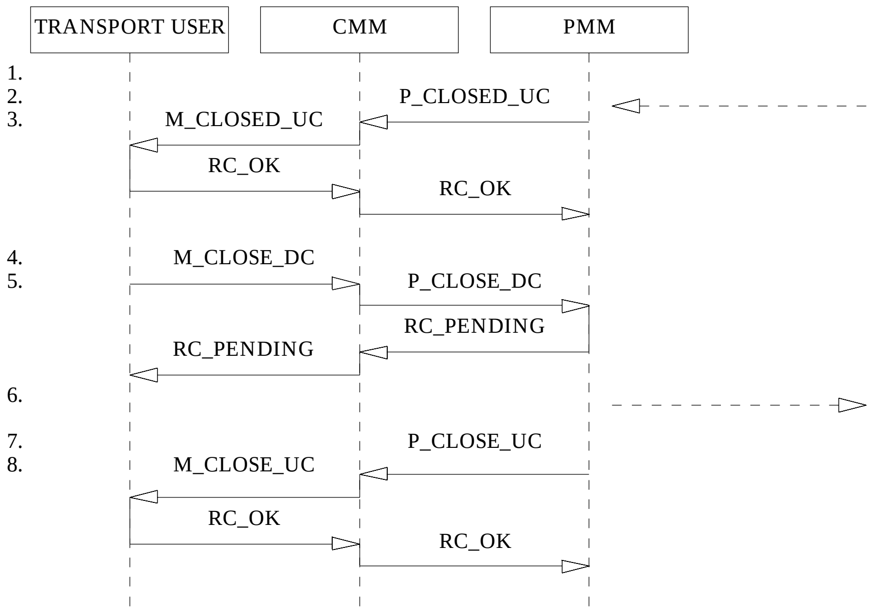


Figure 6-21 General Simplex Orderly Termination — Target

Annotations

5 The protocol for this case is the same as stated above for the simplex abortive target case up to identifier (*5.*). In the simplex orderly case, however, both the M_CLOSE_DC and the P_CLOSE_DC return RC_PENDING.

7-8 Once the external termination flows to the initiator are done, the PMM issues a P_CLOSE_UC to the CMM. The CMM then issues an M_CLOSE_UC to the transport user.

6.5.3 Expedited Termination

In cases where the transport user requires an abortive termination not controlled by pacing and the transport provider has no native mechanism to support it, MPTN provides an expedited termination using out-of-band datagrams, similar to that used for expedited data transfer. See Section 6.3.2 on page 63 for details of how the MPTN_DG_OOB_Data format is handled. Figure 6-22 shows the protocol flow for an expedited termination.

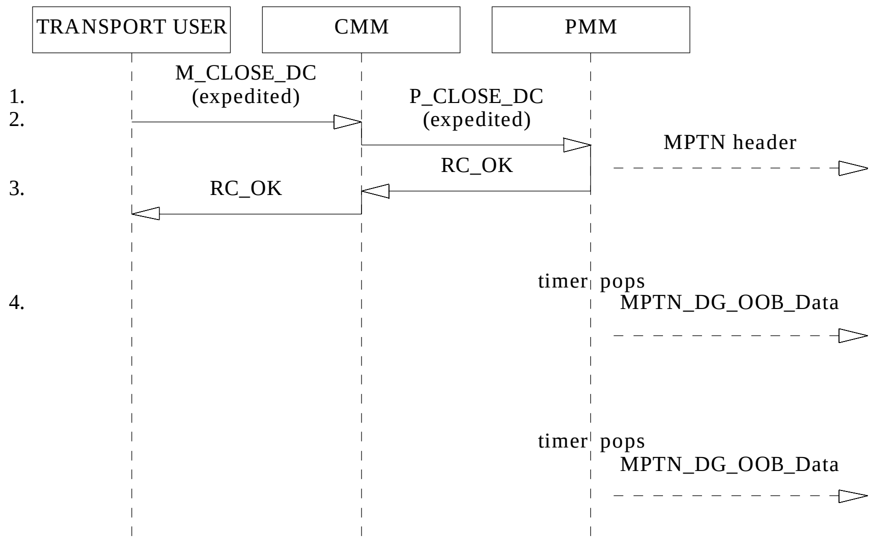


Figure 6-22 Expedited Data Transfer — Initiator

Chapter 7

Connectionless Protocols

MPTN supports the sending and receiving of datagrams over various transport providers. The TLPB provides a non-blocking datagram interface for use by transport users.

As in the connection-oriented case, there are three choices for address registration and resolution, algorithmic, native directory, and MPTN address mapper. Addresses may be cached by the CMM to reduce resolution overhead. Please refer to Chapter 5 for information about initialisation.

Datagram segmentation is supported by using a segment and offset combination. The segment specification field in the MPTN_Datagram format (see the **MPTN Data Formats** Specification (Section 5.2, MPTN_Datagram)) appears in each segment and supplies the total length of the unsegmented datagram and the position of the beginning of the segment relative to the beginning of the unsegmented datagram. See Section 7.1.3 on page 85.

7.1 Sending Datagrams

7.1.1 Target Address Cached

Figure 7-1 illustrates the protocol for sending a datagram in the case of an address already cached by the CMM.

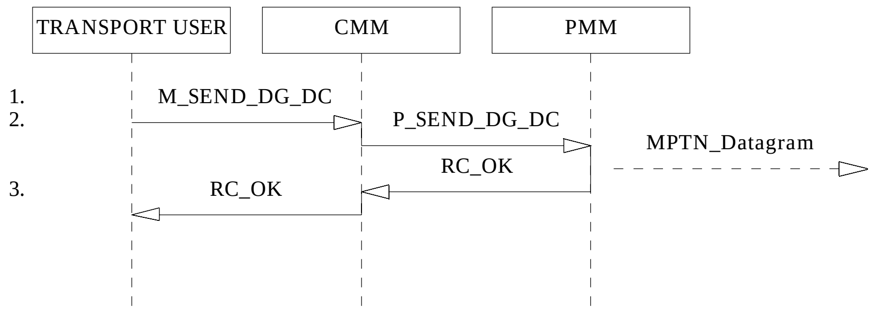


Figure 7-1 Datagram Send: Address Cached

Annotations

1 The transport user issues an M_SEND_DG_DC with the following information:

- The address of the target transport user.
- The service mode requested for the datagram.
- The data to be sent in the datagram.
- An indication of whether to discard previously cached routing information.
- An indication of whether the datagram is a complete or partial datagram segment.

2 Upon receiving the M_SEND_DG_DC, unless the indicator was set to discard previously cached routing information, the CMM finds the cached address mapping for the matching transport user address and issues a P_SEND_DG_DC to the PMM with the same information as in the M_SEND_DG_DC.

The PMM then creates the MPTN_Datagram format, using the header and user data information from the P_SEND_DG_DC, and sends it on the selected transport provider datagram service.

7.1.2 Target Address Not Cached

If the address mapping for the target transport user is not known, then an address mapping function must be used. Figure 7-2, Figure 7-3 and Figure 7-4 illustrate the three variations of the case where address resolution is required: algorithmic, native directory, and address mapper.

In the case of algorithmic address resolution for sending datagrams, the protocol flow is shown in Figure 7-2.

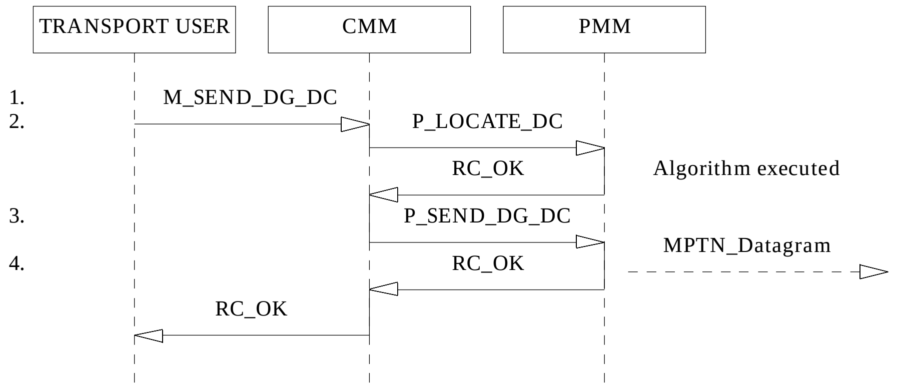


Figure 7-2 Datagram Send: Address Not Cached, Algorithmic Mapping

Annotations

1-4 The transport user issues an M_SEND_DG_DC to the CMM, which in turn issues a P_LOCATE_DC to the PMM so that the PMM can generate the address mapping of the target. This mapping is passed back to the CMM in the return from the P_LOCATE. The CMM then issues a P_SEND_DG_DC to the PMM, which will in turn create the MPTN_Datagram format and send it to the target via the transport provider's native datagram mechanism.

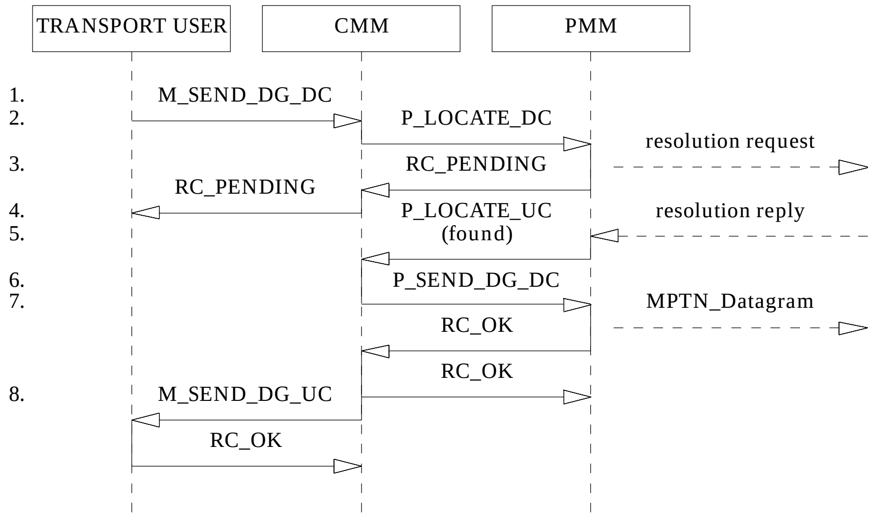


Figure 7-3 Datagram Send: Address Not Cached, Native Directory Service

Annotations

1 In the native directory case, the transport user issues an M_SEND_DG_DC to the CMM.

2-3 Upon receiving the downcall, the CMM issues a P_LOCATE_DC to the PMM requesting an address mapping. The PMM will then proceed to use the transport provider protocol specific mechanism for address resolution.

4 Once the transport provider protocol specific address resolution is completed, the PMM issues a P_LOCATE_UC to the CMM with the following information:

- The status of the address resolution attempt.
- The target transport user's address.
- A list of addresses of the target transport providers found.
- An indication of which transport provider responded to the locate.

6-8 The CMM then issues a P_SEND_DG_DC to the PMM. The PMM then sends the MPTN_Datagram on the selected transport provider datagram service.

Upon return to the P_SEND_DG_DC, the CMM issues an M_SEND_DG_UC to the transport user indicating the status of the datagram send.

The MPTN address mapper case is shown in Figure 7-4.

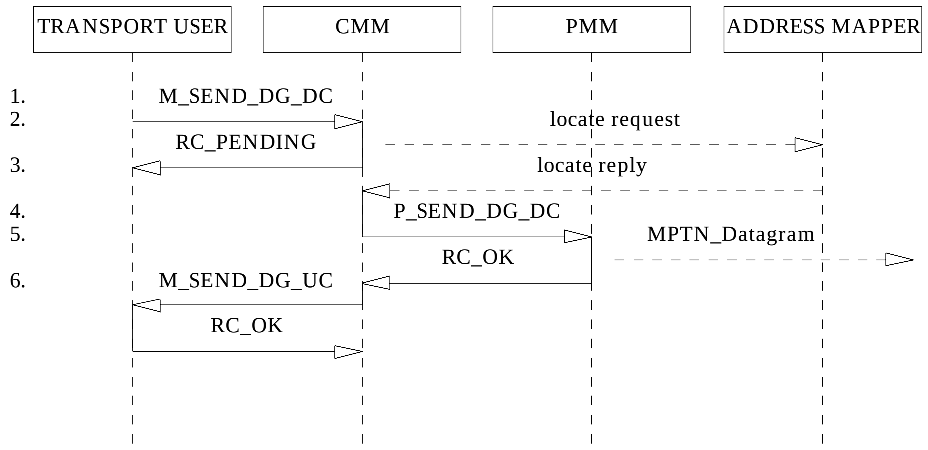


Figure 7-4 Datagram Send: Address Not Cached, MPTN Address Mapper

Annotations

1 The transport user issues an M_SEND_DG_DC.

2 The CMM returns an RC_PENDING and then uses MPTN address mapper flows to get the address mapping required. These flows are described in more detail in the MPTN address mapper reference.

4 Once the mapping has been received from the address mapper, the CMM issues a P_SEND_DG_DC.

5 The PMM then creates an MPTN_Datagram format using the header and user data from the P_SEND_DG_DC, and sends it out on the selected transport provider datagram service.

6 Upon the return to the P_SEND_DG_DC, the CMM issues a M_SEND_DG_UC to the transport user indicating the status of the datagram send.

A datagram send may fail if the address resolution fails. Possible scenarios include:

- Address service (native directory or address mapper) unavailable.
- Address mapping not found.

In this case the CMM issues an M_SEND_DG_UC indicating a negative status for the datagram send.

7.1.3 Datagram Segmentation

Connectionless service does not guarantee that the characteristics of the transport protocol remain the same from one datagram to the next between the two stations. MPTN provides the sender of a datagram with information about the maximum datagram receive capability of the target station, in order to increase the likelihood that the datagram will be accepted.

The CMM of the datagram sender indicates its maximum datagram receive capability in the *Maximum Datagram Size* parameter of MPTN_Datagram, MPTN_DG_OOB_Data, and MPTN_Cntrl_Datagram. The CMM receiver saves and uses that information for future transmissions to that transport user. If large datagrams need to be sent, then segmentation may be required.

MPTN provides for datagram segmentation in two ways:

- Segmentation by the transport user.

  If the transport user requires the ability to segment datagrams, it can do so as follows.

  The transport user first indicates that datagram segmentation is supported when creating the control block (see Section A.3 on page 198). The flow for a datagram segment looks just like a normal datagram send flow (see Figure 7-1 on page 82; assume for the sake of simplicity that the address is cached). When sending the first datagram segment, the transport user indicates on the M_SEND_DG_DC the total length of the datagram and that this is the first segment. Subsequent M_SEND_DG_DCs will indicate that the datagram is a segment and that it is a middle segment (there may be multiple middles), or the last segment. This causes the CMM to keep track of datagrams segments based on the cumulative length, until all have been received. When building the MPTN_Datagram command for each segment, the CMM uses the segment specification field (see the **MPTN Data Formats** Specification (Section 5.2, MPTN_Datagram)) to describe the total length of the unsegmented datagram and the offset position for each segment.

- Segmentation by the CMM. When the CMM is requested to send a datagram, and the CMM has recently received a datagram from that target transport provider containing prov_receive_maxdgm, the CMM uses that value as the maximum datagram send size.

  If the transport provider's maximum supported datagram size is smaller than the transport user's maximum datagram size (with the addition of the MPTN datagram header), then datagrams will be segmented by the CMM without explicit intervention by the transport user. The protocol flow for this case is illustrated in Figure 7-5.

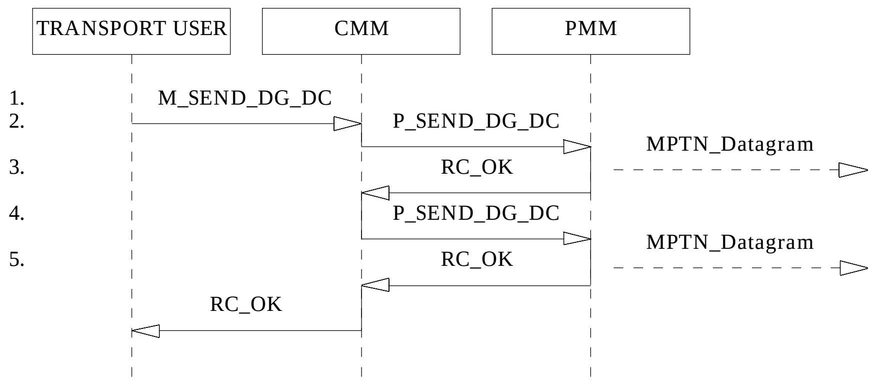


Figure 7-5 Datagram Segmentation: CMM Performs the Segmentation

Annotations

1 The determination of whether or not to segment is made by the CMM on each M_SEND_DG_DC issued.

2-5 When creating the MPTN_Datagrams, the CMM uses the same mechanism as described above, but makes decisions about how to assign offset values based on the maximum datagram length supported by the transport provider. Each datagram segment is then passed to the PMM in separate P_SEND_DG_DCs.

7.1.4 Multicast

When a datagram is sent to a group address (that is, multicast), the datagram is sent over each underlying transport provider in the Access Node. For each transport provider, this involves:

- Resolving the transport users group address to the corresponding transport provider group address for this provider.
- Distributing the multicast over that transport provider. If the provider supports an inherent multicast ability, the inherent multicast function is used to distribute the datagram over this transport provider (see Figure 7-6). Otherwise, the Multicast Server is used to distribute the datagram over this transport provider (see Figure 7-7).

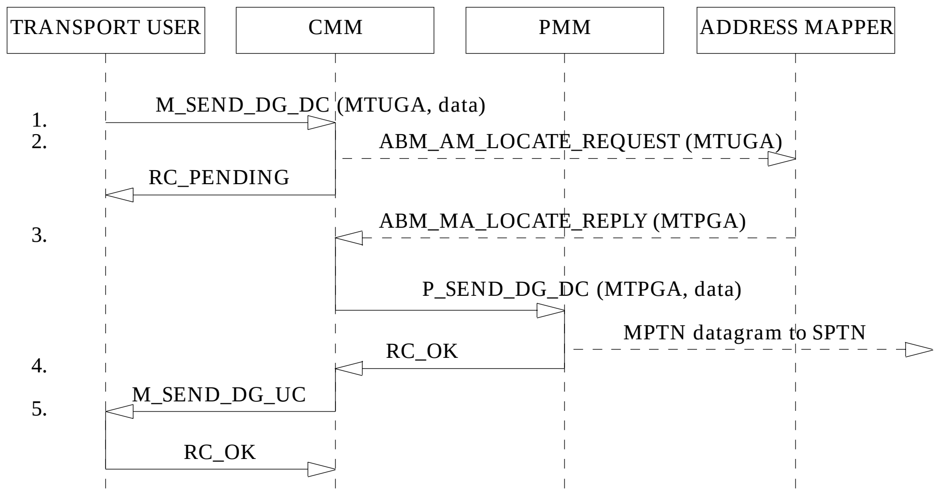

TRANSPORT USER
CMM
PMM
ADDRESS MAPPER
1.
2.
3.
4.
5.
M_SEND_DG_DC (MTUGA, data)
ABM_AM_LOCATE_REQUEST (MTUGA)
RC_PENDING
ABM_MA_LOCATE_REPLY (MTPGA)
P_SEND_DG_DC (MTPGA, data)
MPTN datagram to SPTN
RC_OK
M_SEND_DG_UC
RC_OK

Figure 7-6 Send Flows with Inherent Multicasting

Annotations

1 The Transport User issues an M_SEND_DG_DC, providing the multicast transport user group address (MTUGA) and the data. If the mapping is done through an Address Mapper, then RC_PENDING is returned to the User.

2 The CMM sends an ABM_AM_LOCATE message to the Address Mapper to determine the transport group provider address. If other mapping mapping strategies are used, then the corresponding flows are described in Figure 7-2 on page 82 and Figure 7-3 on page 83.

3 When the reply to the locate comes back, the CMM calls the PMM with data and the multicast transport provider group address (MTPGA).

4 When the PMM sends out the datagram, control returns to the CMM.

5 The CMM calls the Transport User with an M_SEND_DG_UC, completing the send.

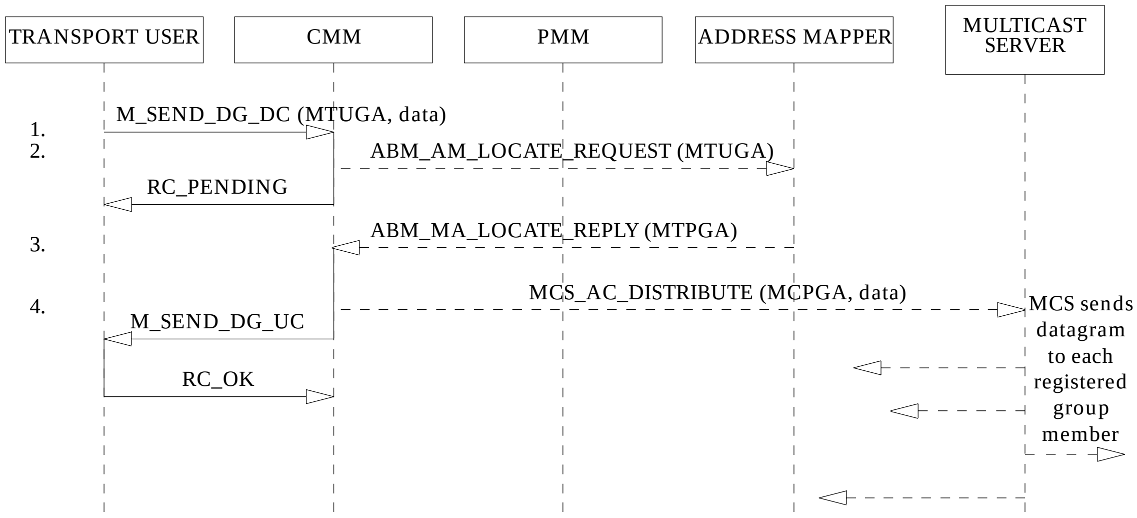

TRANSPORT USER
CMM
PMM
ADDRESS MAPPER
MULTICAST SERVER
1.
2.
3.
4.
M_SEND_DG_DC (MTUGA, data)
ABM_AM_LOCATE_REQUEST (MTUGA)
RC_PENDING
ABM_MA_LOCATE_REPLY (MTPGA)
MCS_AC_DISTRIBUTE (MCPGA, data)
M_SEND_DG_UC
RC_OK
MCS sends datagram to each registered group member

Figure 7-7 Send Flows with Multicast Server

Annotations

1 The Transport User issues an M_SEND_DG_DC, providing the multicast transport user group address (MTUGA) and the data. If the mapping is done through an Address Mapper, then RC_PENDING is returned to the User.

2 The CMM sends an ABM_AM_LOCATE message to the Address Mapper to determine the transport group provider address. If other mapping mapping strategies are used, then the corresponding flows are described in Figure 7-2 on page 82 and Figure 7-3 on page 83.

3 The reply to the locate is received by the CMM.

4 The CMM sends the datagram to the multicast server via the MCS_AC_DISTRIBUTE. The multicast server then sends a unicast datagram to each member of the group. Once the CMM sends the datagram to the multicast server, it calls the Transport User with an M_SENDDG_UC, completing the send.

7.2 Receiving Datagrams

Figure 7-8 illustrates the protocol for receiving datagrams.

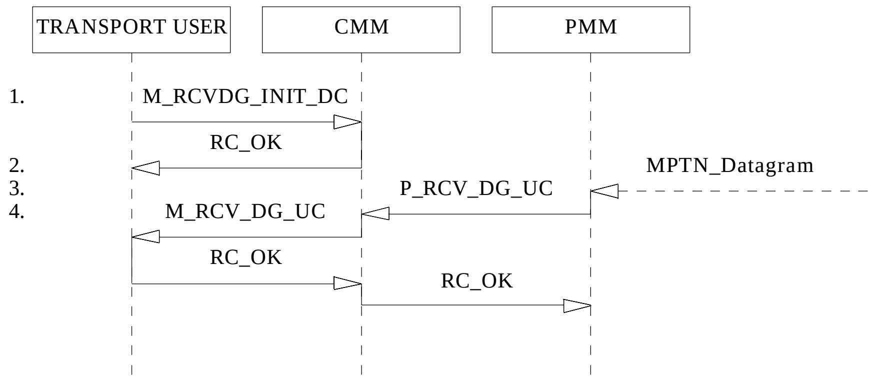


Figure 7-8 Receive datagram

Annotations

1-2 A transport user wishing to receive datagrams issues an M_RCVDG_INIT_DC to the CMM with the following information:

- An indication of whether or not the transport user wishes to receive broadcast datagrams. Datagrams will always be accepted for the transport user's unique address or its group address (if applicable).
- An indication of whether or not the transport user wishes to receive multiple datagrams. If multiple datagrams are desired, all datagrams will be accepted until the control block is destroyed using an M_CLOSE_DC. In the case where multiple datagrams are not desired, the CMM will stop listening for datagrams after one has been received.

The CMM then returns an RC_OK to the transport user to indicate that it is listening for datagrams.

3 When a datagram arrives, the PMM issues a P_RCV_DG_UC to the CMM with the following information:

- An indication of whether this is a datagram segment.
- The parsed datagram command structure. See the **MPTN Data Formats** Specification (Section 5.2, MPTN_Datagram).
- The data received.
- An indication of which transport provider the datagram was received on, in order for the CMM to be able to cache the transport provider id, if caching is supported.

4 Prior to issuing the M_RCV_DG_UC, the CMM retains the prov_receive_maxdgm if it is present in that datagram. This provides additional information for the size of future datagrams to be sent to that destination.

When a datagram is received, the receiving CMM searches the cache for the sender's transport user/transport provider address pair. If an entry is found, and the datagram contains prov_receive_maxdgm, then that value is saved in the prov_receive_maxdgm parameter of that cache entry. If the datagram does not have the prov_receive_maxdgm, then the entry's prov_receive_maxdgm is not changed.

If no entry for the address pair is found, then an entry is created. If the datagram contains prov_receive_maxdgm, then that value is saved in that cache entry. If the datagram does not contain prov_receive_maxdgm, the PMM's prov_protocol_maxdgm is saved as the prov_receive_maxdgm.

Upon receiving the upcall from the PMM, the CMM issues an M_RCV_DG_UC to the transport user with the following information:

- The address of the transport user sending the datagram, as well as the address of the local transport user.
- The data received.
- An indication of whether or not the datagram received was from a broadcast.

A datagram that has been segmented may be reassembled by either the CMM (or at an implementation's discretion, the PMM) or the transport user. If the transport user stated on the M_CREATE_DC that it supported datagram segmentation, reassembly is done by the transport user, otherwise the CMM (or the PMM) does the reassembly. Receiving a datagram that has been segmented occurs as follows:

- The transport user performs the reassembly. In this case, the datagram receive protocol flow looks just like the unsegmented datagram case (Figure 7-8). For each segment received, the M_RCV_DG_UC contains the total unsegmented datagram length and the segment offset needed by the transport user to reconstruct the datagram.
- The CMM (or PMM) performs the reassembly. Figure 7-9 shows the CMM case.

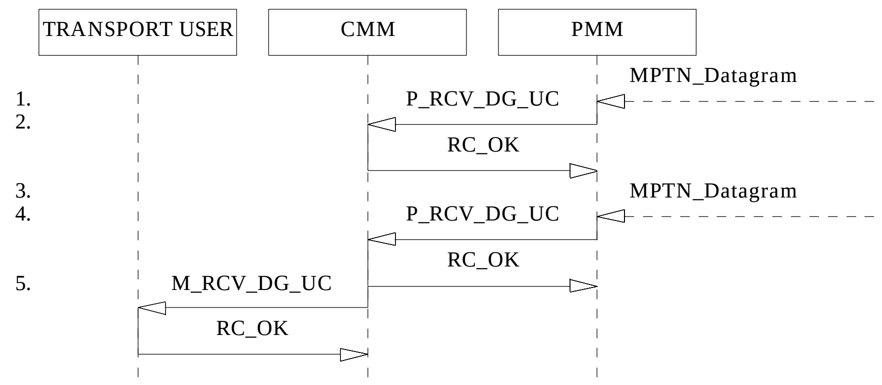


Figure 7-9 Receive Segmented Datagram: CMM Performs the Re-assembly

Annotations

1-2 The MPTN_Datagram format contains a segment specification (see the **MPTN Data Formats** Specification (Section 5.2, MPTN_Datagram)), which contains:

- The length of the entire unsegmented datagram.
- An offset from the beginning of the first datagram.

The PMM issues a P_RCV_DG_UC, passing the parsed datagram header to the CMM. The target CMM stores the segments based on the offset value in the header.

3-5 Once all segments have been received, the CMM issues an M_RCV_DG_UC to the transport user, as if it had been received as one unsegmented datagram.

Note: The segmentation and reassembly functions do not have to be performed by the same components in the initiating and target nodes. For example, if the initiating transport user does the segmentation, it is permissible for either the transport user or the CMM in the target node to perform the reassembly.

7.3 MPTN Control Datagrams

Some transport users support applications that depend on lower level protocol specific information being made available to the application. With MPTN, the piece of the transport protocol stack that would normally generate that protocol specific information may not be available. For MPTN to be able to support this type of application, a special MPTN control datagram is needed. Two examples are given below.

1. The NetBIOS interface supports a STATUS command which allows the retrieval of status and configuration information from another node. Much of the information contained in the response to this command, for example, link level statistics, does not apply to the MPTN environment and could be given default values at the source node. No MPTN flows would be involved if this were the case. Certain applications, however, rely on the *local name table* information which NetBIOS returns in the response. In order to allow an application to retrieve the name table from a remote node a new MPTN flow is required.

2. With raw sockets, the receiving application depends on the IP header being passed up to the application. With MPTN, the sockets application does not actually run over IP, thus IP is not available to generate the IP header. The MPTN_Cntl_Datagram is used to transport this type of data.

Control flow datagrams are distinguished by the fact that they carry transport user control information that was not generated by the actual user application.

MPTN_Cntrl_Datagram is used for carrying transport user control datagrams between two access nodes. The structure of this datagram type is identical to that of the MPTN_Datagram structure except for the value of command_type in the common_prefix. (See the **MPTN Data Formats** Specification (Section 8.1, Common Prefix for MPTN Command)).

The contents of the data field of a control datagram are defined for a specific transport user. Thus a NetBIOS transport user will define the data field of a control datagram as carrying a NetBIOS STATUS command; while a sockets transport user will define the data field of a control datagram as carrying raw sockets. Other transport users that make use of control datagrams will define their own data format.

7.4 MPTN Syntax Mapper Signal Datagram

Some transport protocols depend on functions of the lower layer protocol to trigger upper layer functions. With MPTN, the piece of the transport protocol stack that would normally generate that protocol specific information may not be available.

For MPTN to be able to support this type of application, a special MPTN syntax mapper signal datagram is needed. An example is given below:

> Native SNA will bring down its dependent LU control sessions, the SSCP-PU and SSCP-LU sessions, when the SNA link is deactivated. When these dependent LU control session are nonnative, there is no SNA link to be deactivated. The MPTN_Syntax_Mapper_Signal_Datagram(LINK_DEACT) command is used to request the upstream access node to simulate an SNA link deactivation and to bring down the dependent LU control sessions.

Syntax Mapper Signal datagrams are distinguished by the fact that they carry control information that flows between syntax mappers. The payload of the datagram is a new signal (that is, it is not part of the transport users protocol) but is required to replace something that is part of the lower layer protocol.

MPTN_Syntax_Mapper_Signal_Datagram is used for carrying signal/control datagrams between two syntax mappers. The structure of this datagram type is identical to that of the MPTN_Datagram structure except for the new value of command_type in the common_prefix.

The contents of the data field of a control datagram are defined for a specific transport user.

X/Open CAE Specification

Part 3:

MPTN Transport User/Provider Characteristics

X/Open Company Ltd.

Chapter 8

AF_INET Sockets Transport User Characteristics

8.1 Introduction

For an MPTN AF_INET (INET address family) Sockets Transport User, this chapter defines the architecture for the interface between the AF_INET sockets transport user and the MPTN architecture, at the Transport Layer Protocol Boundary (TLPB). Specifically, it defines the information that flows at that interface for the AF_INET sockets transport user.

The component that provides that information is the Sockets-MPTN Syntax Mapper.

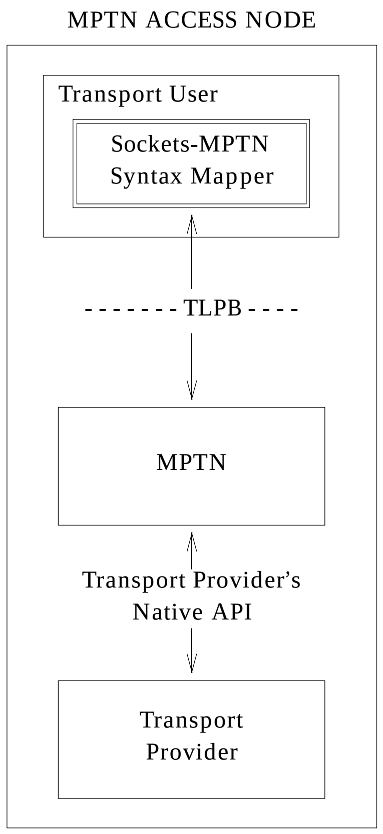

MPTN ACCESS NODE
Transport User
Sockets-MPTN Syntax Mapper
- - - - - - - TLPB - - - -
MPTN
Transport Provider's Native API
Transport Provider

Figure 8-1 The Sockets-MPTN Syntax Mapper, from an MPTN Perspective

8.2 Components of the Sockets Transport User

Figure 8-2 depicts the Berkeley implementation of the sockets API modified to interface to MPTN components.

The Berkeley TCP/IP code is structured into protocol-independent and protocol-dependent pieces. The protocol-independent piece is the box labeled *sockets code*. The interface at the top of this box is the sockets API. The sockets code performs protocol-independent processing for sockets API requests and then forwards the request (if necessary) to the actual protocol supporting the socket. The boundary between the sockets code and the protocol code is called the *protocol switch* or *pr_usrreq* boundary.

Enabling MPTN for the sockets API involves inserting the sockets-MPTN syntax mapper just under the protocol switch boundary. The syntax mapper determines whether a particular socket is associated with a native or non-native partner. If native, it simply forwards the request to the original TCP/UDP pr_usrreq function. If non-native, then the syntax mapper must map the request into appropriate calls to the TLPB.

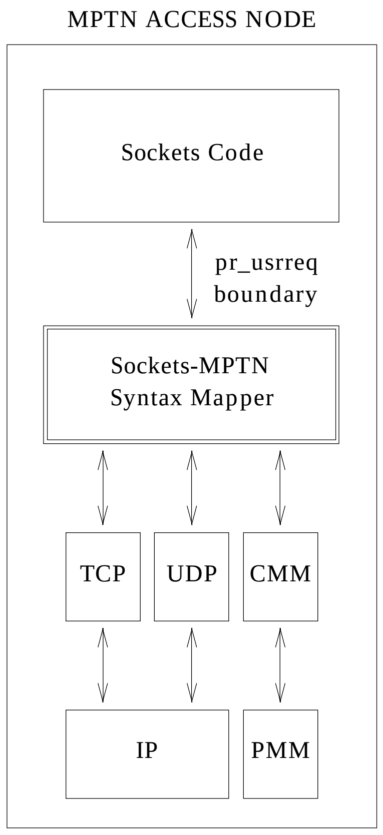


Figure 8-2 The Sockets-MPTN Syntax Mapper, from a BSD Sockets Perspective

Note: Some sockets calls, such as **getsockname**, **getpeername** and **select**, have no corresponding TLPB calls. The syntax mapper and sockets code together must maintain whatever state information is needed to handle calls such as these.

8.3 Design Considerations

The major topics that must be considered when designing a sockets Transport User product are:

- connection establishment / datagram receipt
- flow control and buffer management
- addresses
- sockets API considerations.

Each of these is discussed in following sections.

Throughout this chapter, references to AF_INET Sockets operations are highlighted in **bold** type.

8.3.1 Connection Establishment/Datagram Receipt

Initiating a non-native connection and sending a datagram to a non-native destination is straightforward; the appropriate M_CREATE, M_BIND, M_CONNECT, M_SEND, etc. downcalls are simply made to the CMM.

Receiving inbound non-native connection requests and datagrams is slightly more complicated. The CMM must be notified of the existence of the socket when it becomes *visible*. Sockets become visible when a **bind** call is issued, or when any call requiring a local address is issued without a prior explicit call to **bind** (such calls involve an implicit bind). At this time, the CMM must be notified so that incoming non-native data destined for it is properly received. The syntax mapper may choose to delay any CMM calls for a socket until it becomes visible, and then issue all the calls (for example, M_CREATE, M_BIND, M_LISTEN) during processing of a single socket call such as **listen**.

Once the CMM has been notified of the existence of the socket, it may forward incoming connect requests to the transport user. Note that the socket might not be in an appropriate state to respond to the request. Specifically, for example, the socket backlog may be zero, so the syntax mapper must hold the incoming connect request and provide an appropriate response, should the socket state change during the time the inbound connection remains active.

8.3.2 Flow Control and Buffer Management

Sockets have send and receive buffers associated with them. The size of these buffers can be adjusted by the sockets application (within implementation limits). These buffers are used to control the rate of data flow.

Sending Data

When data is sent on a stream socket, it is first placed in the socket's send buffer. Then the transport provider is notified that the data is to be sent. The data remains in the socket send buffer until the transport provider has completed the send operation. If the underlying transport has blocked the send operation, then the data will remain in the socket send buffer until the send operation unblocks. The sockets application can continue to send data until it fills up the send buffer. At that time, the sockets code will either block the application which attempts to send more data (if the socket is blocking) or return **ewouldblock** (if the socket is non-blocking).

Sockets applications may send *out of band* data by specifying the MSG_OOB flag on the call to send data. Such out of band data is passed to the CMM as *expedited* data. Note that an application may issue a multi-byte **send** request with the MSG_OOB flag. However, with the BSD sockets API, only the last byte is *marked* as out of band. The syntax mapper should pass the leading bytes to the CMM as normal data and flag the last byte as expedited.

When data is sent on a datagram socket, it is not placed in the socket's send buffer. Rather, it is immediately sent. A datagram can be dropped if there are insufficient buffers to complete send or receive processing for it.

Receiving Data

When data arrives for a stream socket, it is placed in the socket's receive buffer. If this buffer becomes full, no more data will be received from the transport provider until the sockets application has read some of the buffered data. At this time the sockets-MPTN syntax mapper provides buffers (if they have been requested) to the transport provider so that it may continue receiving data.

When data arrives for a datagram socket, it is placed in the socket's receive buffer only if there is sufficient space for the entire datagram. If there is insufficient space, the datagram is discarded by the syntax mapper. The syntax mapper never refuses to provide buffers to the transport provider for incoming datagrams.

8.3.3 Addresses

MPTN Parameter	Contents
MPTN Qualifier	X'02'
Address Mode	X'01' - individual
Node Address	4 byte IP address in network order
Local Address	2 byte port address in network order

Table 8-1 Address Values

Special Address Consideration for Address 0.0.0.0

IP conventionally uses address 0.0.0.0 to send loopback messages; that is, to send messages to this node. The syntax mapper needs to convert the IP address 0.0.0.0 to the local IP address for this transport user.

Dynamic Address Verification Required?

Sockets does not require dynamic address verification. Sockets addresses are centrally administered to guarantee uniqueness.

8.3.4 Sockets API Considerations

Sockets IP options relate directly to the IP transport provider, being used primarily for network testing or debugging. They are not applicable to the MPTN environment because the IP layer is not involved when going over non-native transport providers. Because raw sockets requires the sending and receiving of the IP header, IP options are, in fact, carried in the control datagram, though they have no effect.

The *security* IP option is supported by the MPTN Service Mode.

For **sock_stream (TCP)**, Type of Service cannot be changed dynamically per packet.

8.4 Characteristics of Connections

8.4.1 User Transport Requirements

This section lists the communication characteristics of connections that the sockets transport user needs. Not all of these characteristics necessarily need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 8-6 on page 106.

MPTN Parameter	**Contents**
Maximum record length	0 bytes
Maximum expedited record length	1 byte
Maximum connection termination data length	0 bytes
Close types	Simplex orderly, duplex orderly, duplex abortive
Expedited marking	Required

Table 8-2 User Transport Requirements

This information corresponds to the CONN_CHARS data structure.

8.4.2 Service Mode

Not applicable.

8.5 Characteristics of Datagrams

This section lists the communication characteristics of datagrams that the sockets transport user needs. Not all of these characteristics will need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 8-6 on page 106.

The syntax mapper employs two types of datagrams:

- user datagrams
- control datagrams.

8.5.1 User Datagrams

The syntax mapper provides the following information for the user datagrams it wishes to send.

User Datagrams	Contents
Destination Address	See Section 8.3.3 on page 100.
Source Address	See Section 8.3.3 on page 100.
User datagram	The user datagram.

Table 8-3 User Datagrams

8.5.2 Control Datagrams

In native environments, some transport users depend on lower level protocol-specific information being made available to the application. For example, in the raw sockets environment the receiving application depends on the IP header being passed up to the application. This is the case, for example, for the PING application.

However, in non-native environments promoted by MPTN, the sockets application does not actually run over IP, so IP is not available to generate the IP header. The MPTN control datagram is used to transport this type of data.

It is the job of the MPTN components to ensure that the IP header is part of the control datagram. This can be done either by utilising the IP stack to generate this header, or by generating the IP header itself. It is the job of the syntax mapper to generate the header.

Control datagrams carry additional information along with any data that the user application provided. For the sockets case, this additional information is the IP header that must accompany the data from raw sockets. (Some raw sockets may build the IP header themselves. It is up to the syntax mapper to build the IP header when necessary.)

Following is an example of ping over SNA, showing the header information added by each component.

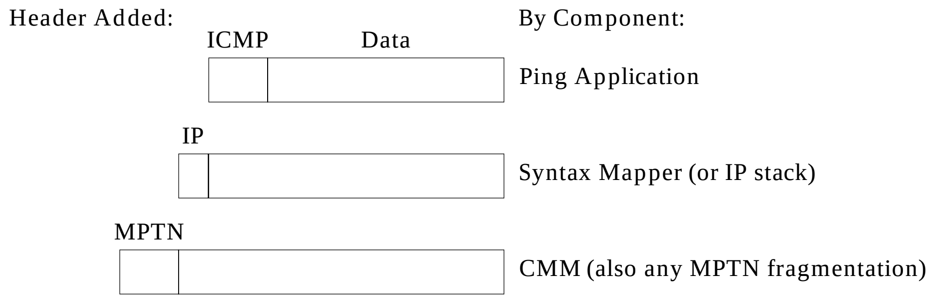

Header Added:
By Component:
ICMP
Data
Ping Application
IP
Syntax Mapper (or IP stack)
MPTN
CMM (also any MPTN fragmentation)

Figure 8-3 Ping Header Information per Component

The syntax mapper provides the following information for the control datagrams it wishes to send.

Type of Data	Contents
Destination Address	See Section 8.3.3 on page 100.
Source Address	See Section 8.3.3 on page 100.
Control datagram	The control datagram.

Table 8-4 Control Datagrams

8.6 Characteristics of Data

This section lists the characteristics of data that the sockets transport user needs. Not all of these characteristics will need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 8-6 on page 106.

Type of Data	Contents
Service type	ST_CO (connection-oriented) for SOCK_STREAM, or ST_CL (connectionless) for SOCK_DGRAM
Data type	Only applicable for service type = ST_CO, for which it is set to STREAM.
Maximum Length of Connection Data	0 bytes
Maximum Length of Connection Reply Data	0 bytes
Maximum Length of Data on a Datagram	64k – 9
Maximum Length of Transport User's address	6 bytes: 4-byte IP address + 2-byte port number
partial_records	No
Connection characteristics	See Section 8.4.1 on page 101.
Session outage notification	No
In line	Yes, Socket OOB data is not MPTN expedited, since it can not bypass normal data queues (it is Urgent, not Expedited). Thus, it must be sent in line even if the underlying transport provider has native expedited support.
Non-Queued Response sent as expedited	No
dg_seq#_recognized	No
tuser_data_recognized	No
direct_tlpb_type	0 (not direct TLPB user)
Connect-connect	Yes
Parallel connections	No
Not listening notification	No

Table 8-5 Types of Data

This information corresponds to the M_INFO data structure.

8.7 Compensations

The following table identifies the compensations required for support of an AF_INET sockets transport user on specific transport providers[4]. The native transport provider, TCP / IP, is not shown.

Compensation Value	Shorthand Name: Compensation Description	SNA	NetBIOS	IPX/SPX
X'00'	NOCOMP: On a record-oriented transport network, a message with no associated compensation. On a stream-oriented transport network, a record boundary marker (when no other compensation is needed).	Yes	Yes	Yes
X'01'	EXP: Expedited message.	Yes	Yes	Yes
X'03'	EXP_ACK: Expedited message acknowledgement.			
X'10'	DA_TERM: Duplex-abortive termination.			
X'12'	DA_TERM_ACK: Duplex-abortive termination acknowledgement.			
X'14'	DO_TERM: Duplex-orderly termination.	Yes	Yes	Yes
X'16'	DO_TERM_ACK: Duplex-orderly termination acknowledgement.			
X'18'	SA_TERM: Simplex-abortive termination.			
X'1A'	SA_TERM_ACK: Simplex-abortive termination acknowledgement.			
X'1C'	SO_TERM: Simplex-orderly termination.	Yes	Yes	Yes
X'1E'	SO_TERM_ACK: Simplex-orderly termination acknowledgement.			
X'20'	RECSEG: Segmented message.			

4. The compensation for the OSI transport provider will be added as work proceeds.

Compensation Value	Shorthand Name: Compensation Description	SNA	NetBIOS	IPX/SPX
X’21’	EXPSEG: Segmented expedited message.			
X’83	DG_OOB_DATA: MPTN_DG_OOB_Data message needed for expedited data.			
X’84’	DG_KEEPALIVE: MPTN_DG_KEEPALIVE_Hdr message needed for session outage notification.			

Table 8-6 Compensations Required to Support Sockets Transport Users

Chapter 9

TCP/UDP Transport Provider Characteristics

9.1 Introduction

For an MPTN TCP / UDP Transport Provider, this chapter defines the semantic rules that govern the way the TCP / UDP Transport Provider acts when it is invoked by the transport user. The semantics of the B-Specific Protocol Boundary (BSPB) are defined in terms of the actions taken by the TCP / UDP Transport Provider PMM when its BSPB entry point is called by the CMM or the transport user.

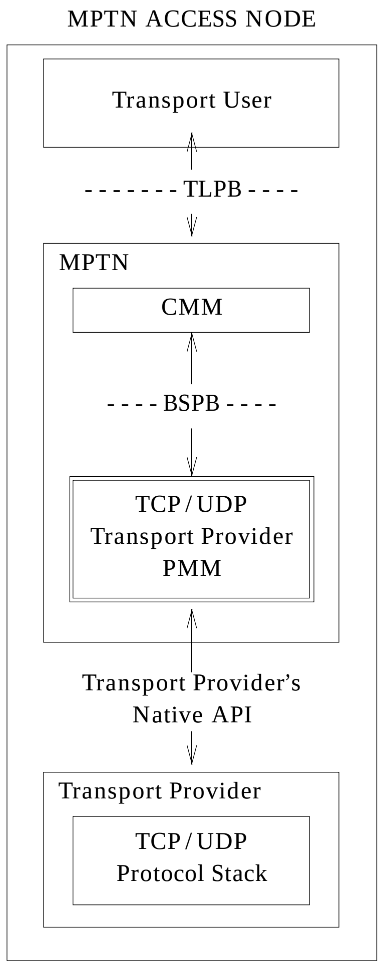


Figure 9-1 The TCP / UDP Transport Provider, from an MPTN Perspective

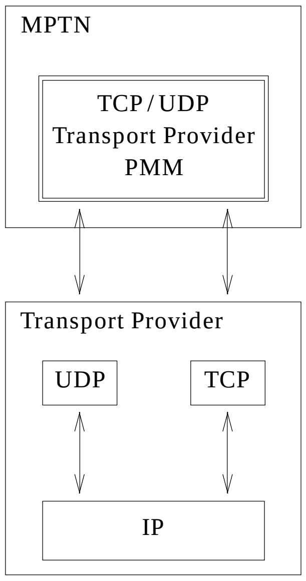


Figure 9-2 The TCP / UDP Transport Provider, from a TCP / UDP Perspective

9.2 Design Considerations

The major topics that should be considered when designing a TCP / UDP Transport Provider product are:

- transport provider characteristics
- naming and address mapping
- use of well-known port
- sending expedited data.

Each of these is discussed in a following section.

Throughout this chapter, references to TCP / UDP operations are highlighted in **bold** type.

9.2.1 Transport Provider Characteristics

This section lists the characteristics that the TCP / UDP transport provider supports. The compensations needed depend on the characteristics required by the transport user.

Note: In some cases (particularly for termination), compensations may be needed even when the characteristics of the transport user match those of the transport provider.

Transport Provider Characteristic	**Value Assigned by TCP PMM**
prov_tsdu	**MAX_TCP_SEND_SIZE**
Note: may vary from one platform to another.	
prov_etsdu	**0**
Note: urgent data mechanism not used.	
prov_send_maxdgm	**SO_SNDBUF**
Note: The value of the socket option for the implementation of sockets that is being used. If sockets is not being used, then use a mapping between that protocol and sockets.	
prov_receive_maxdgm	**SO_RCVBUF**
Note: The value of the socket option for the implementation of sockets that is being used. If sockets is not being used, then use a mapping between that protocol and sockets.	
prov_protocol_maxdgm	**576**
Note: may vary from one platform to another.	
addr_service	domain name server
Note: may vary by implementation.	
connect_supported	**TRUE**
stream_supported	**TRUE**
record_supported	**FALSE**
expedited_marking	**FALSE**
datagram_supported	**TRUE**
multicast_supported	**FALSE**
simplex_abortive	**FALSE**
simplex_orderly	**TRUE**
duplex_abortive	**TRUE**
duplex_orderly	**FALSE**
session_outage	**FALSE**

Table 9-1 TCP Transport Provider Characteristics

This information corresponds to the PROV_INFO data structure.

9.2.2 Naming and Address Mapping

When a transport user requests that a connection be set up non-natively, and the CMM chooses to use the TCP transport provider, the address of the transport user to which the connection must be set up has to be mapped to the IP address of the machine where that remote transport user resides. This is the address mapping problem.

There are three kinds of address mapping techniques:

- algorithmic
- the use of a protocol specific directory
- a general purpose MPTN address mapper.

The use of the protocol-specific directory mechanism is described below.

The TCP PMM uses the domain name server (DNS) to map user addresses to IP addresses. The name must be converted to ASCII in a protocol-specific manner and put in the form *Hostaddr.Netid*. Finally, a standard extension is, for example, added to this string to yield a valid domain name. The form of this extension is *<PROTOCOL>.company_name.COM*. A DNS query can then be made (via a **gethostbyname** call) to map the domain name to an IP address. For example, an APPC name, which must be fully qualified, is first converted from EBCDIC to ASCII. Then, the name is converted to the form *LUname.NetID*.

For an SNA transport user that is a dependent LU with DLUR, the connection name is registered with the IP address of ‘‘127.0.0.4’’. As part of its initialisation process, the TCP/IP PMM will try to resolve address 127.0.0.4, and if an SNA address is returned it will place the SNA connection network name in the PROV_INFO structure.

Since the DNS has no dynamic update facility, this form of address mapping requires that for all transport user names that are to be reachable over TCP/IP, the corresponding domain names as described above must be entered in the database(s) of the organisational DNS(s).

9.2.3 Use of a Well-known Port

The address mapping mechanism yields an IP address for the machine where a given transport user address is located. To actually communicate with the PMM on that remote node, a port number is required. MPTN has been assigned the *well-known port* number 397 by the Internet Activities Board (IAB). This number is appended to all IP addresses to yield the complete address to use to send connection requests (as well as connectionless data) to a remote TCP PMM.

9.2.4 Sending Expedited Data

The reason an expedited data compensation is required is that expedited data cannot *only* be sent on the underlying TCP connection where all data traffic occurs, because if the connection is congested, the expedited data will not reach the partner. It is possible to send the transport user’s expedited data, including the length prefix and compensation headers, using the TCP urgent data mechanism. However, if the connection is congested (TCP window size = 0), it is not clear that in all TCP implementations even the urgent data indication would flow. In TCP, the sending of urgent data is signalled to the partner by setting the **URG** bit in the *next* TCP packet which flows to the partner. If the connection is completely congested, *there is no next TCP packet that is ready to flow immediately*, and therefore, the partner does not realise that urgent data is being sent. The Berkeley implementation addresses this problem by sending a TCP packet with the **URG** bit set even when the window size is 0. According to RFC 793[5], TCP urgent data will not flow if the TCP pacing window size is 0.

Rather than depend on this implementation choice, it is more prudent to assume that TCP does not support *true* expedited data and define a compensation.

5. RFC 793 is the original Request For Comment document that describes the TCP protocol. The C source code of the Berkeley implementation of TCP/IP is the most accurate description of a majority of the TCP/IP implementations in existence today.

9.3 BSPB Considerations

This section describes the relationship between the BSPB verbs and the socket calls (highlighted in **bold** font) that usually occur.

P_INIT
In terms of socket calls, the PMM makes an **accept** call to receive non-native connection requests, and a **recvfrom** call to receive non-native datagrams.

P_LOCATE
If a protocol-specific directory is used, the PMM transforms the transport user name to a domain name as described in Section 9.2.2 on page 109, and calls **gethostbyname**. Once the IP address is returned (or there is an error), the PMM makes a P_LOCATE upcall to the CMM to report the results of the name lookup.

P_CREATE
The PMM simply saves the information internally. No socket calls are made in response to a P_CREATE downcall.

P_CONNECT
The init_user_addr and the connect_corr fields together uniquely identify an MPTN connection. These fields are saved internally for future use - in an out-of-band datagram (OOB DG) for example. The comp_hdrs field contains the list of compensations that the PMM will be required to provided for this connection. Since some compensations (such as session-outage-notification, or SON) may be negotiated down, the final comp_hdrs field is passed back to the PMM on the return from the P_CONNECT upcall. Therefore, this particular value of the comp_hdrs field is ignored.

P_ACCEPT_UC
The P_ACCEPT upcall is made by the PMM when it receives a native connection request from a remote node. Once the PMM is initialised (see P_INIT in this section), it makes an **accept** socket call to receive connection requests on the well-known TCP port. When this **accept** call unblocks because of an inbound TCP connection request, the PMM calls the socket **recv** function to receive the MPTN_Connect, the first packet which flows on the native connection, and makes a P_ACCEPT upcall to inform the CMM about the inbound connection request.

P_ACCEPT_DC
The CMM makes a P_ACCEPT downcall in response to the P_ACCEPT upcall. The P_ACCEPT downcall indicates if the partner's connection request was accepted by the local transport user, the compensations to be performed for this connection, the local transport user characteristics (some of which may be negotiated down values of what the partner requested), and the MPTN_Connect RSP packet.

Note: The P_ACCEPT downcall may arrive *either before or after* the P_ACCEPT upcall returns.

On receiving the P_ACCEPT downcall, the PMM calls the socket **send** function to send the MPTN_Connect RSP to the partner. Then, it sets up internal structures to be able to receive incoming data and to receive P_SEND downcalls locally.

P_SEND
Before sending the data on the TCP connection, the PMM builds a 4-byte length prefix and a 1-byte compensation header that will precede the actual data in the line flow. The compensation headers that are used are defined in Section 9.4 on page 113. Then, the PMM issues sockets calls to send the header and the data field to the partner. These calls could be a single **writev** or a series of **send** calls (or any other equivalent set of calls). If the entire

data and the headers are successfully sent (that is, the TCP stack accepts the data), the PMM returns RC_OK to the call and the transport user frees the data buffer. If the entire data could not be sent (because of congestion on the TCP connection), the PMM must queue the data internally and try to send it when the congestion clears. In that case, the PMM returns RC_PENDING to the downcall, and when the data is finally sent successfully, informs the transport user.

P_RECEIVE

When the PMM realises that data is being sent to it on a given connection, it must read the length prefix, the compensation header, and the data itself using the socket **recv** function. The data is read into a buffer whose size is determined from the length prefix. Whether the data is normal or expedited, and whether it is a partial of a whole record, can be decided by examining the compensation byte.

P_CLOSE

On receiving a duplex abortive close request, the PMM schedules the close to be performed in the background (since it may take a considerable amount of time) and returns RC_PENDING to the CMM. On receiving a close request, TCP PMM changes its internal state such that it stops receiving any other requests on the connection, regardless of whether the connection was initiated by the local or the partner node. Next it discards any unsent data queued inside the PMM (due to congestion on the TCP connection) that it might have been trying to send in the background. That enforces the semantics of abortive close. Finally, it sends the termination data to the partner, preceded by a 1-byte compensation header that identifies a duplex abortive close request. This request and data may or may not be sent as *expedited* data, since the semantics of duplex abortive close stipulate that the close request must reach the partner *immediately*, even if there is congestion on the connection. For example, in the SNA over TCP/IP combination, the DA_TERM header and the termination data are packaged inside an OOB datagram and sent to the partner. Once it is certain that the partner has received the close request and the termination data, the PMM calls the socket **soclose** function to close the TCP connection, and makes a P_CLOSE upcall to signal that the connection is closed. It does not wait for a termination data reply.

P_CLOSED

Once the P_CLOSED upcall returns, the PMM calls the **soclose** function to close the underlying TCP connection, and cleans up all data structures associated with the connection.

9.4 Compensations

The following table identifies the compensations required for support for specific transport users[6] over the TCP transport provider. The native transport user, sockets, is not shown.

Compensation Value	**Shorthand Name: Compensation Description**	**SNA**
X'00'	NOCOMP: On a record-oriented transport network, a message with no associated compensation. On a stream-oriented transport network, a record boundary marker (when no other compensation is needed).	Yes
X'01'	EXP: Expedited message.	Yes
X'03'	EXP_ACK: Expedited message acknowledgement.	Yes
X'10'	DA_TERM: Duplex-abortive termination.	Yes
X'12'	DA_TERM_ACK: Duplex-abortive termination acknowledgement.	
X'14'	DO_TERM: Duplex-orderly termination.	
X'16'	DO_TERM_ACK: Duplex-orderly termination acknowledgement.	
X'18'	SA_TERM: Simplex-abortive termination.	
X'1A'	SA_TERM_ACK: Simplex-abortive termination acknowledgement.	
X'1C'	SO_TERM: Simplex-orderly termination.	
X'1E'	SO_TERM_ACK: Simplex-orderly termination acknowledgement.	
X'20'	RECSEG: Segmented message.	Yes
X'21'	EXPSEG: Segmented expedited message.	
X'83'	DG_OOB_DATA: MPTN_DG_OOB_Data message needed for expedited data.	Yes

6. NetBEUI over TCP/UDP is defined by RFCs 1001 and 1002, so there are no MPTN compensation considerations for that combination. Similarly, OSI over TCP/UDP is defined by RFC 1006, so there are no compensation considerations for that combination.

Compensation Value	**Shorthand Name: Compensation Description**	**SNA**
X'84'	DG_KEEPALIVE: MPTN_DG_KEEPALIVE_Hdr message needed for session outage notification.	Yes

Table 9-2 Compensations Required to Support TCP Transport Providers

Chapter 10

SNA Transport User Characteristics

10.1 Introduction

For an MPTN Systems Network Architecture (SNA) Transport User, this chapter defines the architecture for the interface between the SNA transport user and both the CMM and the PMM at the Transport Layer Protocol Boundary (TLPB). Specifically, it defines the information that flows at those interfaces for the SNA transport user. The component that provides this information is the SNA-MPTN Syntax Mapper, hereafter referred to as the syntax mapper.

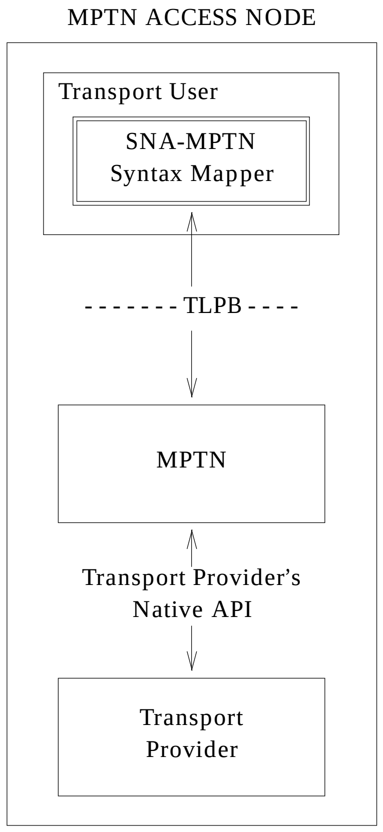


Figure 10-1 The SNA-MPTN Syntax Mapper, from an MPTN Perspective

10.2 Components of the SNA Transport User

Figure 10-2 shows the components involved in SNA session establishment and data transfer and their interfaces to MPTN components.

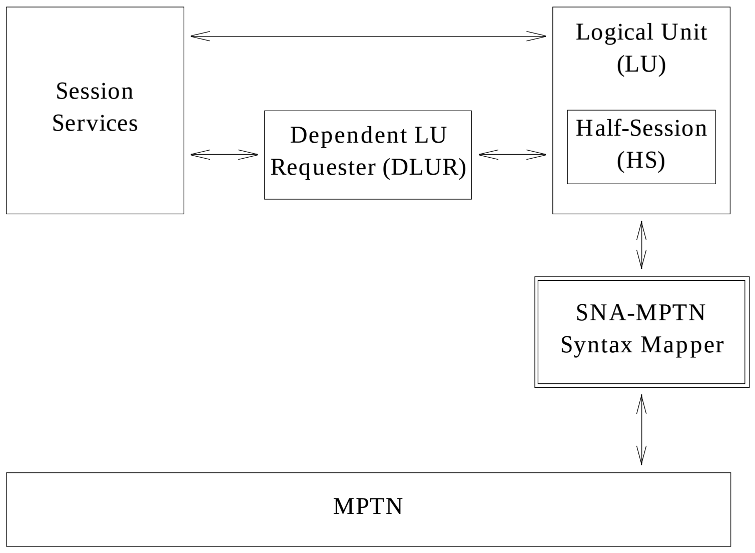


Figure 10-2 The SNA-MPTN Syntax Mapper, from an SNA Perspective

The SNA Transport User consists of:

- parts of session services
- the Dependent LU Requester (DLUR)
- the Logical Unit
- the SNA-MPTN Syntax Mapper.

With a few small exceptions, the LU components operate without change, and access MPTN via path control interfaces. Thus the syntax mapper appears to the SNA components as another path control. It appears to the MPTN components as a transport user, issuing TLPB calls such as M_CONNECT or M_SEND.

When each LU is defined, it issues a call to the syntax mapper, which in turn registers that name with MPTN and receives any connection requests coming over MPTN for that LU.

10.3 Design Considerations

The major items that must be considered when designing an SNA Transport User product are:

- treatment of SNA transmission headers
- session establishment
- pacing
- logical unit address mapping
- dependent logical unit support.

Each of these is discussed in following sections.

Throughout this chapter, references to SNA operations are highlighted in **bold** type.

10.3.1 Treatment of SNA Transmission Headers

The syntax mapper deals with the 2-byte sequence number of any format identifier type of transmission header (TH). From the TH, it must extract the Expedited Flow Indicator and the Mapping Field. These are reflected in parameters of the M_SEND_DC verb.

10.3.2 Session Establishment

Session Manager needs only one change to make it possible to run sessions over non-native transports. When a session is requested, the Session Manager accesses a local table to determine whether the connection should be native or whether it should run over a different transport protocol.

10.3.3 Pacing and Buffer Management

Half-session

No change is necessary for the half-session component. When the half-session sends data to path control, the path control that receives the message is the syntax mapper.

In order to control the rate at which the half-session introduces traffic on the connection, the syntax mapper carries out SNA session pacing protocols with the local half-session, treating the interface between them as a pacing stage. The following description assumes the reader understands SNA pacing and SNA pacing negotiation. (Information on SNA pacing can be found in the Session Level Pacing section of the Intermediate Session Routing chapter of **Systems Network Architecture, Advanced Peer-to-Peer Networking, Architecture Reference**, IBM reference SC30-3422.)

Some transport providers, for example TCP/IP, offer flow control protocols. It is the job of the syntax mapper to tie the SNA pacing to the transport provider flow control. Thus, when the syntax mapper issues M_SEND calls that block because the transport connection is not accepting further traffic yet, the syntax mapper withholds pacing responses from the half-session so that the half-session stops accepting data from the application. The protocols used between these two components are exactly the same protocols used on a single pacing stage. Therefore no changes are needed in half-session to handle the intra-node pacing. However, care must be taken in the syntax mapper to participate correctly in pacing, as described in the following section.

Data may arrive on the transport connection corresponding to a session when no SNA buffers are available to receive the data. The data will not be received from the transport user until an

SNA buffer is freed. When the application receives the data, the SNA buffer will be freed. On the user's side, the user may try to send data, but the thread may be blocked if no pacing window is available. The availability of buffers for each session is controlled by SNA pacing. SNA performs hop-by-hop pacing for native SNA sessions by flowing pacing indications (normally *Isolated Pacing Responses (IPRs)* and *Isolated Pacing Messages (IPMs)*) on the session to the next hop.

For non-native SNA sessions, *no hop-by-hop pacing takes place*. Instead, the syntax mapper pretends to be the next hop, and sends IPRs or IPMs to, and receives IPRs or IPMs from, the local half-session.

The logic used by the syntax mapper to exchange IPRs or IPMs with the local half-session, and to respond to buffer allocation requests by the PMM, is described below.

When the syntax mapper receives a frame containing inbound data, it adds a pacing indicator to the frame when necessary, to simulate the condition where the partner requests an IPR or IPM from the local half-session. In all other cases, the pacing indicator is turned off before the frame is passed to SNA. When the local half-session sends an IPR or IPM, the syntax mapper intercepts it and updates internal structures which determine whether requests for more buffers from the PMM's receive thread should be satisfied.

When data is sent from the local half-session to the syntax mapper, the frame may have the pacing indicator set if the half-session wants the partner to send it an IPR or IPM. If the conditions are right, the syntax mapper sends an IPR or IPM to the half-session. Otherwise, the syntax mapper waits for pending M_SEND requests to complete before sending the IPR or IPM to the half-session.

The pseudo-code below describes the syntax mapper's pacing logic in greater detail. This pseudo-code assumes that the transport user provides buffers for MPTN and that adaptive session pacing is used. The important concept is that the MPTN components wait to get a buffer from the transport user. The technique shown here is only one way of implementing the concept. The term *HS* in the pseudo-code means local half-session.

Example 10-1 Pacing Logic on the Receive Path

```
Initialisation code:
   receive_current_window_size = 0;
   receive_new_window_size     = 1;
   MPTN_waiting_for_buffer     = FALSE;

/********************
 * Routine to process a PMM request for a memory buffer.
 ********************/
Process_GET_BUFFER ( ... )
{
   if ((receive_current_window_size + receive_new_window_size ) > 0)
      return frame to PMM's receive thread;
   else {
      MPTN_waiting_for_buffer = TRUE;
      return RC_NOBUFFS;
      /* This will cause receive thread to stop receiving data
         on that transport connection. */
   }
}

/********************
 * Routine to process an M_RECEIVE upcall from the CMM, containing
 * a frame with inbound data.
 ********************/
Process M_RECEIVE ( ... )
{
   if (receive_current_window_size == 0) {
      receive_current_window_size = receive_new_window_size;
      receive_new_window_size = 0;
      receive_current_window_size = receive_current_window_size - 1;
      set pacing indicator in frame;
   }
   else {
      receive_current_window_size = receive_current_window_size - 1;
      turn off pacing indicator in frame;
      }
   send frame to HS;
   .......
}

/********************
 * Routine to process a Pacing response from the local half-session
 ********************/
Process_Pacing_rsp_From_HS ( ... )
{
   receive_new_window_size = IPR size;
   if (PMM_waiting_for_buffer) {
      make buffer available;
      MPTN_waiting_for_buffer = FALSE;
   }
}
```

Example 10-2 Pacing Logic on the Send Path

```
    next_window_size       = implementation choice;

    send_buffers_allocated = 1;
    HS_waiting_for_IPR    = FALSE;

/********************
 * Process data (in a frame) being sent by the local half-session
 ********************/
Process_HS_Send ( ... )
{
    if (pacing indicator set in frame)
        remember that this buffer contain a pacing request;
    .......
    make M_SEND downcall;
    if (return code from downcall == RC_OK) {
        /* data was sent successfully */
        if (buffer being freed contained a pacing request)
            send IPR of size (next_window_size) to HS;

    }
    .........
}

/********************
 * Routine to process a PMM request to release a memory buffer
 * PMM releases memory only if RC from M_SEND_DC was PENDING
 ********************/
Process_FREE_BUFFER ( ... )
{
    /* data was sent successfully */
    if (buffer being freed contained a pacing request)
        send IPR of size (next_window_size) to HS;
    }
}
```

10.3.4 Addresses

MPTN Parameter	**Contents**
MPTN Qualifier	X'0B'
Address Mode	X'01' - individual
Node Address	NetID.LUname in EBCDIC. Source address = fully-qualified PLUname Destination address = fully-qualified SLUname
Local Address	NULL

Table 10-1 Address Values

Dynamic Address Verification Required?

SNA does not require dynamic address verification. SNA addresses are centrally administered to guarantee uniqueness.

10.3.5 SNA Transport User Data

SNA transport users may send transport user data when registering to the CMM (M_BIND_DC) or setting up an MPTN connection. The format of the SNA transport user data is the same as the format of MPTN optional fields (refer to the **MPTN Data Formats** Specification for details on MPTN optional fields).

The tables below describe the various SNA transport user data formats and explains when each format is used.

SNA Transport User Data Formats

The currently defined SNA transport user data commands are:

User Data Name	Reference	Command Type
Link Activation	Table 10-9 on page 129	X'01'
Link Activation +Rsp	Table 10-10 on page 130.	X'02'
Link Activation -Rsp	Table 10-11 on page 131.	X'03'
Link Deactivation	Table 10-12 on page 131.	X'04'
Link Deactivation +Rsp	Table 10-13 on page 132.	X'05'
Link Deactivation -Rsp	Table 10-14 on page 132.	X'0B'
Diagnostics	Table 10-3.	X'06'
LU Type	Table 10-4 on page 124.	X'07'
Extended Connection Indicator	Table 10-5 on page 124.	X'08'
Non-queued Response Indicator	Table 10-6 on page 125.	X'09'
CP Name	Table 10-7 on page 126.	X'0A'
General Negative Response	Table 10-8 on page 126.	X'0C'

Table 10-2 SNA Transport User Data Formats

SNA Transport User: Diagnostics

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X'06'	1	
processing specification		1	
command length		2	
transport user data		2	

Table 10-3 SNA Transport User, Diagnostics

Processing Specification

The processing specification bits indicate whether this is a request or response, action to be taken by a destination that does not understand this transport user data command, and whether the destination understood the command or not.

The format of the Processing specification is:

Bit	**Meaning**
0 to 5	Reserved
6	When set to 1 in a request, indicates that the destination must reject the command if this transport user data command is unrecognised. Echoed in a response.
7	Reserved in a request. When set to 1 in a response, indicates that the destination did not recognise this transport user data command. Bits 6 and 7 set to B'11' in a response indicate that the command failed because this transport user data command was not recognised by the destination.

Transport User Data for Diagnostics

This command is used to include diagnostic information about the previous request. It is used only for link activation and MPTN_Connect flows.

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not required if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X'06'.

The Transport User Data is the PRC which indicates why the transport user data was rejected.

PRC	**Meaning**
X'0001'	The SNA stack on the node that received the link activation was in the process of being brought down.
X'0002'	A link was already being activated by the other side. In this case, the local and remote CP names (one could be a 4-byte ID block / ID num) are converted to ASCII, compared, and the alphabetically larger CP name rejects the link activation request.
X'0003'	There was a format error in the user data; that is, invalid command length value.
X'0004'	The command failed, reason unknown.
X'0005'	Semantic error.

SNA Transport User: LU Type

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X'07'	1	
processing specification		1	
command length		2	
transport user data		1	

Table 10-4 SNA Transport User, LU Type

Description

This command is used to register the LU Type flag with the address mapper.

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not required if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is 5 bytes.

The M_BIND_DC for an SNA transport user must include Transport User Data. The LU Type field is a required field in the user data. This transport user data, which is sent on the MPTN_AM_Register_Request, is of the following format:

1 byte LU type flag:

— 0 if Independent LU

— 1 if Dependent LU

— 2 if Node ID (aka IDBLK / IDNUM).

Presence Rule

This transport user data command flows in the user data field of the ABM_AM_Register_Request for all SNA transport users.

SNA Transport User: Extended Connection Data Indicator

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X'08'	1	
processing specification		1	
command length		2	
transport user data		1	

Table 10-5 SNA Transport User, Extended Connection Data Indicator

Description

This command is used to indicate to the partner that the connection data sent in the MPTN_Connect request is of the extended format; that is, it includes additional bytes prior to the 2 byte sequence number at the beginning of the connection data. The Transport User Data field indicates the format of the additional bytes. This transport user command is sent only when the connection data is of this extended format. The only time this extended format is used is for downstream LU support, refer to Section 10.3.7 on page 127 for details.

Bit 6 of the Processing Specification field is set to 1, indicating that the command is to be rejected with a negative response to the originator if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X’05’.

The Transport User Data is set to X’01’ to indicate that the connection data field includes the 2 byte DAF’ and OAF’.

Presence Rule

This transport user data command flows in the user data field of the MPTN_Connect to indicate that the connection data is in the extended format.

SNA Transport User: Non-queued Response Indicator

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X’09’	1	
processing specification		1	
command length		2	
transport user data		1	

Table 10-6 SNA Transport User: Non-queued Response Indicator

Description

This command is used to indicate that this transport user supports sending Non-queued responses as expedited data.

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not required if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X’05’.

The Transport User Data is set to X’01’ to indicate that the connecting side wants to send non-queued responses as expedited data. This can be negotiated down to X’00’ if the receiving end does not want to send non-queued responses as expedited.

Presence Rule

This transport user data command flows in the user data field of the MPTN_Connect to indicate that this user supports sending non-queued responses as expedited data.

SNA Transport User: CP Name

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X'0A'	1	
processing specification		1	
command length		2	
transport user data		0 to 17	

Table 10-7 SNA Transport User, CP Name

Description

This command is used to register the CP Name with the address mapper.

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not required if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is 4 to 21 bytes.

The M_BIND_DC for an SNA transport user must include Transport User Data. This transport user data, which is sent on the MPTN_AM_Register_Request, may include the optional CP name, in the following format:

0-17 byte CP name

Presence Rule

This optional transport user data command flows in the user data field of the ABM_AM_Register_Request for all SNA transport users.

Presence when the transport user needs to register it's CP name with the address mapper.

SNA Transport User: General Negative Response

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X'0C'	1	
processing specification		1	
command length		2	
transport user data		0	

Table 10-8 SNA Transport User, General Negative Response

Description

This command is used to send a negative response to a previous transport user data signal. This command may be followed by a diagnostic command giving more information on the error.

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not required if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

Presence Rule

This optional transport user data command flows in the user data field when a negative response needs to be given to a previous transport user command.

10.3.6 Dependent Logical Unit Support with DLUR

As part of SNA network activation, a *dependent logical unit (DLU)* requires the assistance of a systems services control point (SSCP) to activate its LU-LU session; therefore it uses a DLUR-DLUS session. Dependent LUs act as secondary LUs and have an LU-LU session limit of one.

All LU-LU sessions look alike to MPTN. Session services functions (for example, secondary initiation) are above MPTN. A DLUR-DLUS session appears as a pair of standard LU type 6.2 sessions to MPTN. Refer to **APPN Dependent Logical Unit Requester Architecture** (copies are available[7] on request) for a description of the dependent logical unit requester.

If the name of a dependent LU is not known ahead of time, then when an *Activate Logical Unit*(ACTLU) arrives at the DLUR node indicating the LU name, the name must be made known to MPTN by issuing the appropriate M_CREATE_DC, M_BIND_DC and M_LISTEN_DC verbs, using fully-qualified LU names.

The SNA transport user learns its connection network name from the M_REPORT_INFO_UC. The presence of the connection network name in the M_REPORT_INFO_UC informs the SNA transport user that it is attached to a connection network. When the DLUS / DLUR session is established, the connection name is reported to the DLUS as part of the normal DLUS / DLUR flows.

10.3.7 Dependent Logical Unit Support without DLUR

When the link between a node containing an SSCP and a node containing dependent LUs is activated, the required SSCP-PU and SSCP-LU sessions are established. When the path between the SSCP and the dependent LU is non-native (for example, SNA over TCP / IP), there is no SNA link activation to signal the establishment of the SSCP-PU and SSCP-LU sessions. Figure 10-3 on page 128 explains how the link activation is simulated and how the SSCP-PU and SSCP-LU session are established in a non-native environment.

7. The APPN DLUR Architecture is available from the AIW Home Page at URL / /wwwidd.raleigh.ibm.com /app /aiwhome.htm

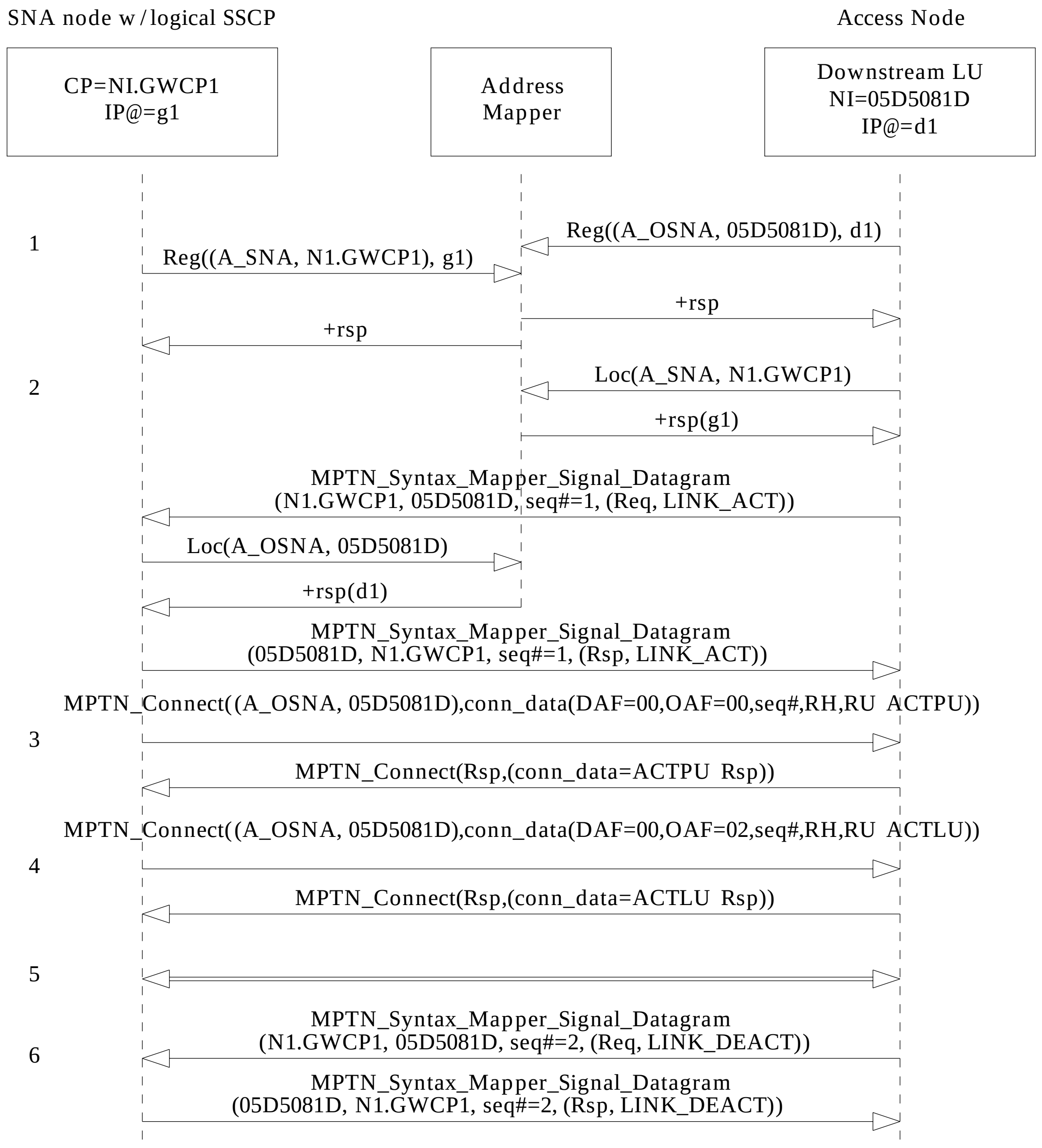

SNA node w/logical SSCP
Access Node
CP=NI.GWCP1
IP@=g1
Address Mapper
Downstream LU
NI=05D5081D
IP@=d1
1
Reg((A_OSNA, 05D5081D), d1)
Reg((A_SNA, N1.GWCP1), g1)
+rsp
+rsp
2
Loc(A_SNA, N1.GWCP1)
+rsp(g1)
MPTN_Syntax_Mapper_Signal_Datagram
(N1.GWCP1, 05D5081D, seq#=1, (Req, LINK_ACT))
Loc(A_OSNA, 05D5081D)
+rsp(d1)
MPTN_Syntax_Mapper_Signal_Datagram
(05D5081D, N1.GWCP1, seq#=1, (Rsp, LINK_ACT))
MPTN_Connect((A_OSNA, 05D5081D),conn_data(DAF=00,OAF=00,seq#,RH,RU ACTPU))
3
MPTN_Connect(Rsp,(conn_data=ACTPU Rsp))
MPTN_Connect((A_OSNA, 05D5081D),conn_data(DAF=00,OAF=02,seq#,RH,RU ACTLU))
4
MPTN_Connect(Rsp,(conn_data=ACTLU Rsp))
5
6
MPTN_Syntax_Mapper_Signal_Datagram
(N1.GWCP1, 05D5081D, seq#=2, (Req, LINK_DEACT))
MPTN_Syntax_Mapper_Signal_Datagram
(05D5081D, N1.GWCP1, seq#=2, (Rsp, LINK_DEACT))

Figure 10-3 Downstream LUs

The following notes correspond to the numbers in Figure 10-3:

1. The transport user address/transport provider address pair in the access node (dependent LU in the downstream node), and the node containing the logical SSCP is registered with the Address Mapper.

2. ACTPU and ATCLU are normally issued when the link between the node containing the logical SSCP and the downstream node is activated. In this case, there is no SNA link to be activated. Instead, when the access node (downstream node) is activated it will send a LINK_ACT request via an MPTN syntax mapper signal datagram. Before this request can be sent, the access node needs to issue an ABM_Locate_Request to get the IP address for the node containing the logical SSCP.

The node containing the logical SSCP will send an LINKACT response to the access node via an MPTN syntax mapper signal datagram, but it first must get the IP address for the access node.

3. In response to the LINKACT request, the node containing the logical SSCP will establish two or more session between itself and the access node.

   An MPTN_Connect is sent to the access node. The first byte of the destination node's local address is the ODAI bit, the second and third bytes are the OAF and DAF. Notice that the ACTPU is sent as the connection data. This establishes the SSCP-PU session.

4. A second MPTN_Connect is sent to the access node. The first byte of the destination node's local address is the ODAI bit, the second and third bytes are the OAF and DAF. Notice that the ACTLU is sent as the connection data. This establishes the SSCP-LU session. One SSCP-LU session is established for each dependent LU in the access node.

5. At some point, the actual LU-LU session(s) is established, used to exchange data, and finally closed. The details of these flows are not shown.

6. Since there is no SNA link, the link deactivation also needs to be simulated. This is simulated by the access node sending a LINK_DEACT request to the node containing the logical SSCP via an MPTN syntax mapper signal datagram. The node containing the logical SSCP will send the LINK_DEACT response back to the access node via an MPTN syntax mapper signal datagram.

10.3.8 Syntax_Mapper_Signal_Datagram: Activation/Deactivation

Table 10-9 and Table 10-12 on page 131 show the payload format when the MPTN_Syntax_Mapper_Signal_Datagram is used for link activation and link deactivation.

All other MPTN_Syntax_Mapper_Signal_Datagrams will be considered semantically incorrect and will be returned immediately with:

- a X'0C' command appended (General Negative Response)
- a X'06' command appended (Diagnostic with PRC=X'0005', semantic error)
- the destination unrecognised bit set in all unrecognised commands.

The datagram portion of this datagram uses the same format and name space as the SNA Transport User Data (refer to Section 10.3.5 on page 122).

SNA Transport User: Link Activation

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X'01'	1	
processing specification		1	
command length		2	
transport user data		0	

Table 10-9 SNA Transport User, Link Activation

Description

This is the Transport User Data sent to simulate an SNA link activation; refer to Section 10.3.7 on page 127 for details.

This transport user command is sent only as the data packet in an MPTN_Syntax_Mapper_Signal_Datagram. If used, the command is sent by the transport user in the data field of an M_SEND_DG_DC (with dg_type=syntax_mapper).

Bit 6 of the Processing Specification field is set to 1, indicating that if the command is not understood by the destination, bit 7 should be set to 1 and the Link Activation command returned to the originator in another MPTN_Syntax_Mapper_Signal_Datagram. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X'04'. There is no specific Transport User Data, the presence of the link activation command is all that is needed.

SNA Transport User: Link Activation +RSP

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X'02'	1	
processing specification		1	
command length		2	
transport user data		0	

Table 10-10 SNA Transport User, Link Activation +RSP

Description

This is the Transport User Data sent to simulate an SNA link activation positive response; refer to Section 10.3.7 on page 127 for details.

This transport user command is sent only as the data packet in an MPTN_Syntax_Mapper_Signal_Datagram. If used, the command is sent by the transport user in the data field of an M_SEND_DG_DC (with dg_type=syntax_mapper).

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not needed if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X'04'. There is no specific Transport User Data, the presence of the link activation +rsp command is all that is needed.

SNA Transport User: Link Activation -RSP

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X’03’	1	
processing specification		1	
command length		2	
transport user data		0	

Table 10-11 SNA Transport User, Link Activation -RSP

Description

This is the Transport User Data sent to simulate an SNA link activation negative response; refer to Section 10.3.7 on page 127 for details.

This transport user command is sent only as the data packet in an MPTN_Syntax_Mapper_Signal_Datagram. If used, the command is sent by the transport user in the data field of an M_SEND_DG_DC (with dg_type=syntax_mapper).

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not needed if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X’0A’. There is no specific Transport User Data, the presence of the link activation -rsp command is all that is needed. However, there will be a diagnostic (see Table 10-3 on page 122) will be present.

SNA Transport User: Link Deactivation

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X’04’	1	
processing specification		1	
command length		2	
transport user data		0	

Table 10-12 SNA Transport User, Link Deactivation

Description

This is the Transport User Data sent to simulate an SNA link deactivation; refer to Section 10.3.7 on page 127 for details.

This transport user command is sent only as the data packet in an MPTN_Syntax_Mapper_Signal_Datagram. If used, the command is sent by the transport user in the data field of an M_SEND_DG_DC (with dg_type=syntax_mapper).

Bit 6 of the Processing Specification field is set to 1, indicating that the command is to be rejected with a negative response to the originator if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X’04’. There is no specific Transport User Data, the presence of the link deactivation command is all that is needed.

SNA Transport User: Link Deactivation +RSP

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X’05’	1	
processing specification		1	
command length		2	
transport user data		0	

Table 10-13 SNA Transport User, Link Deactivation +RSP

Description

This is the Transport User Data sent to simulate an SNA link deactivation positive response; refer to Section 10.3.7 on page 127 for details.

This transport user command is sent only as the data packet in an MPTN_Syntax_Mapper_Signal_Datagram. If used, the command is sent by the transport user in the data field of an M_SEND_DG_DC (with dg_type=syntax_mapper).

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not needed if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X’04’. There is no specific Transport User Data, the presence of the link deactivation rsp command is all that is needed.

SNA Transport User: Link Deactivation -RSP

Field Name	Identifier (if any)	Size Range (in bytes)	Occurrences
common prefix		4	1
command type	X’0B’	1	
processing specification		1	
command length		2	
transport user data		0	

Table 10-14 SNA Transport User, Link Deactivation -RSP

Description

This is the Transport User Data sent to simulate an SNA link deactivation negative response; refer to Section 10.3.7 on page 127 for details.

This transport user command is sent only as the data packet in an MPTN_Syntax_Mapper_Signal_Datagram. If used, the command is sent by the transport user in the data field of an M_SEND_DG_DC (with dg_type=syntax_mapper).

Bit 6 of the Processing Specification field is set to 0, indicating that a negative response is not needed if the command is not understood by the destination. All other bits in the Processing Specification field are set as described under this heading immediately following Table 10-3 on page 122.

The Command Length is set to X’04’. There is no specific Transport User Data, the presence of the link deactivation rsp command is all that is needed.

10.4 Characteristics of Connections

10.4.1 User Transport Requirements

This section lists the communication characteristics of connections that the SNA transport user needs. Not all of these characteristics necessarily need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 10-18.

MPTN Parameter	Contents
Maximum record length	Sequence number (2 bytes)[1] + Request Header (RH) (3 bytes) + Request Unit (RU)
Maximum expedited record length	95 bytes[2]
Maximum connection termination data length	95 bytes[3]
Close types	Duplex abortive, with UNBIND sent as expedited data.
Expedited marking	Not used.

Table 10-15 User Transport Requirements

This information corresponds to the CONN_CHARS data structure.

1 The sequence number is the last 2 bytes of the (TH).

2 The maximum expedited record length requirement is driven by EXPD RU, which carries expedited user data. The 95 bytes is composed of:

- 2 bytes of sequence number
- 3 bytes of RH
- 90 bytes of expedited RU:
  - 4 bytes of SNA expedited data header:
    - 2 bytes of request code
    - 1 byte reserved
    - 1 byte of length
  - 86 bytes of user data.

3 The maximum connection termination data length requirement is driven by the UNBIND RU. The 95 bytes is composed of:

- 2 bytes of sequence number
- 3 bytes of RH
- 90 bytes of UNBIND RU:
  - 6 bytes of UNBIND RU header
  - 84 bytes of UNBIND RU:
    - 56 bytes of control vector 35
    - 28 bytes of control vector 60

10.4.2 Service Mode

MPTN Service Mode

SNA Class of Service	**MPTN Service Mode**	**Meaning**
X’7B’ CONNECT	X’01’	Basic service
X’7B’ CPSVCMG	X’01’	Basic service
X’7B’ CPSVCMGR	X’01’	Basic service
X’7B’ SNASVCMG	X’01’	Basic service
X’7B’ BATCH	X’02’	High bandwidth
X’7B’ INTER	X’03’	Fast response time
X’7B’ BATCHSC	X’04’	Secure and high bandwidth
X’7B’ INTERSC	X’05’	Secure and fast response time

Table 10-16 MPTN Service Modes

Not all transport providers offer the full complement of MPTN service modes; refer to the documentation for the appropriate transport provider.

User Service Mode

Any user-defined service mode may be passed as an MPTN user-defined service mode.

The user is responsible for assuring that the user service mode is consistent across the entire network.

10.5 Characteristics of Datagrams

Datagrams are not used by the SNA Syntax Mapper.

10.6 Characteristics of Data

This section lists the characteristics of data that the SNA transport user needs. Not all of these characteristics will need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 10-18 on page 138.

Type of Data	**Contents**
Service type	ST_CO (connection-oriented)
Data type	Record
Maximum Length of Connection Data	517 bytes: Sequence number (2 bytes)[1] + RH (3 bytes) + BIND request RU (up to 512 bytes, including control vectors)
Maximum Length of Connection Reply Data	Same as above.
Maximum Length of Data on a Datagram	0 (Datagrams not used by SNA transport user.)
Maximum length of Transport User's address	17 bytes: 8-byte NetID, a period, and 8-byte LUname
partial_records	Implementation dependent
Connection Characteristics	See Section 10.4.1 on page 133.
Session outage notification	Yes
In line	No
Non-queued Response Indicator (see note 2)	Yes
dg_seq#_recognized	Yes, if downstream LUs are supported No, if downstream LUs not supported
tuser_data_recognized	Yes, if downstream LUs are supported No, if downstream LUs not supported
direct_tlpb_type	0 (not direct TLPB user)
Connect-connect	No
Parallel connections	Yes
Not listening notification	Yes

Table 10-17 Types of Data

This information corresponds to the **M_INFO** data structure.

1 The sequence number is the last 2 bytes of the Transmission Header (TH).

2 Some back-level SNA Transport Users do not send non-queued response as expedited. Thus, non-queued responses are only sent as expedited if the SNA transport users on both ends of the connection support it. A partner's support of this is indicated in the transport_user_data field of either the M_ACCEPT_UC (if the partner initiated the connection) or on the M_CONNECT_UC (if partner did not initiate the connection).

10.6.1 Connection Data

For an SNA transport user that is an independent LU, or a dependent LU with DLUR, the Connection Data consist of the following fields:

1. Sequence number (last 2 bytes of the TH)
2. RH
3. RU

For an SNA transport user that is an dependent LU without DLUR, the Connection Data consist of the following fields:

1. DAF/OAF (from the TH)
2. Sequence number (last 2 bytes of the TH)
3. RH
4. either
   - RU BIND
   - RU ACTPU
   - RU ACTLU.

10.6.2 Send Data Payload

For an SNA transport user, the send data payload consist of the following:

- For non-segmented records, or the first segment of a segmented record:
  1. Sequence number (last 2 bytes of the TH)
  2. RH
  3. RU
- For second and remaining segments of a segmented record:
  1. RU

10.7 Compensations

The following table identifies the compensations required for specific transport providers.[8] The native transport provider, SNA, is not shown.

Compensation Value	Shorthand Name: Compensation Description	TCP/UDP	NetBIOS	SPX/IPX
X'00'	NOCOMP: On a record-oriented transport network, a message with no associated compensation. On a stream-oriented transport network, a record boundary marker (when no other compensation is needed).	Yes	Yes	Yes
X'01'	EXP: Expedited message.	Yes	Yes	Yes
X'03'	EXP_ACK: Expedited message acknowledgement.	Yes	Yes	Yes
X'10'	DA_TERM: Duplex-abortive termination.	Yes	Yes	Yes
X'12'	DA_TERM_ACK: Duplex-abortive termination acknowledgement.			
X'14'	DO_TERM: Duplex-orderly termination.			
X'16'	DO_TERM_ACK: Duplex-orderly termination acknowledgement.			
X'18'	SA_TERM: Simplex-abortive termination.			
X'1A'	SA_TERM_ACK: Simplex-abortive termination acknowledgement.			
X'1C'	SO_TERM: Simplex-orderly termination.			
X'1E'	SO_TERM_ACK: Simplex-orderly termination acknowledgement.			
X'20'	RECSEG: Segmented message.	Yes	Yes	Yes

8. The compensation for the OSI transport provider will be added as work proceeds.

Compensation Value	**Shorthand Name: Compensation Description**	**TCP/UDP**	**NetBIOS**	**SPX/IPX**
X'21'	EXPSEG: Segmented expedited message.			
X'83'	DG_OOB_DATA: MPTN_DG_OOB_Data message needed for expedited data.	Yes	Yes	Yes
X'84'	DG_KEEPALIVE: MPTN_DG_KEEPALIVE_Hdr message needed for session outage notification.	Yes	Yes	

Table 10-18 Compensations Required to Support SNA Transport Users

10.8 Mapping to TLPB Verbs

SNA Functions	TLPB Verb(s)
Defining an LU	M_CREATE_DC M_BIND_DC (see note 1) M_LISTEN_DC
Sending Session Activation req	M_CONNECT_DC (see note 2)
Sending Session Activation rsp	M_ACCEPT_DC (see note 2)
Sending RU	M_SEND_DC
Sending Session Deactivation req	M_CLOSE_DC(abortive)
Registration complete	M_BIND_UC
Receiving Session Activation rsp	M_CONNECT_UC (see note 2)
Receiving Session Activation req	M_ACCEPT_UC (see note 2)
Receiving RU	M_RECEIVE_UC
Send Session Deactivation rsp	M_CLOSE_UC
Receiving Session Deactivation req	M_CLOSED_UC

Table 10-19 Mapping to TLPB Verbs

1 The M_BIND_DC for an SNA Transport User must include SNA Transport User Data. Refer to Section 10.3.5 on page 122 for details on SNA Transport User Data.

2 The M_ACCEPT_DC, M_ACCEPT_UC, M_CONNECT_DC and M_CONNECT_UC may include SNA Transport User Data. Refer to Section 10.3.5 on page 122 for details on SNA Transport User Data.

Chapter 11

SNA Transport Provider Characteristics

11.1 Introduction

For an MPTN Systems Network Architecture (SNA) Transport Provider, this chapter defines the semantic rules that govern the way the SNA Transport Provider acts when it is invoked by the transport user. The semantics of the B-Specific Protocol Boundary (BSPB) are defined in terms of the actions taken by the SNA Transport Provider PMM when its BSPB entry point is called by the CMM or the transport user.

Figure 11-1 illustrates the position of the SNA Transport Provider PMM in relation to the transport user component and the transport provider component of the MPTN architecture.

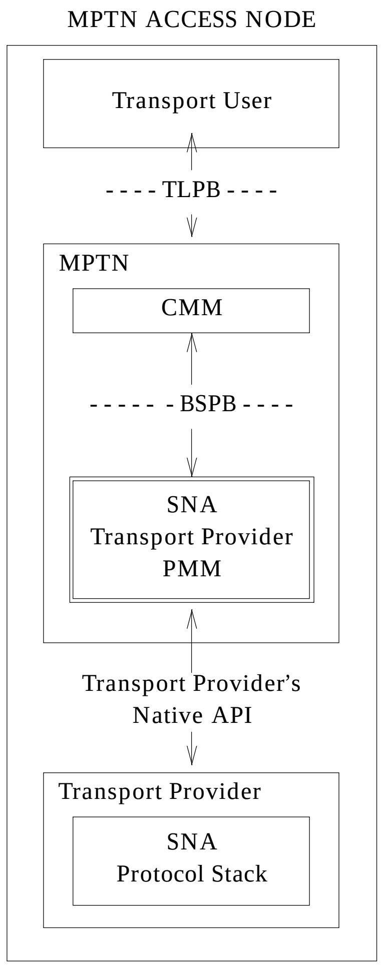


Figure 11-1 The SNA Transport Provider, from an MPTN Perspective

Figure 11-2 illustrates the position of the SNA Transport Provider PMM in relation to the Advanced Program-to-Program Communication (APPC) protocol stack. Specifically, the SNA transport provider PMM sees the APPC interface at the top of Logical Unit (LU) type 6.2.

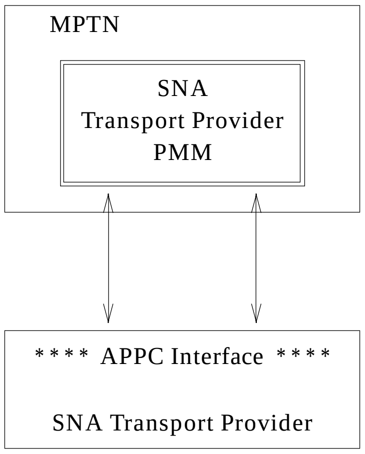


Figure 11-2 The SNA Transport Provider, from an SNA Perspective

11.2 Design Considerations

The major topics that must be considered when designing an SNA Transport Provider product are:

- initialisation
- naming and addressing
- datagrams
- connection setup
- data transfer
- pacing
- connection termination.

Each of these is discussed in a following section. Throughout this chapter, references to SNA operations are highlighted in **bold** type.

11.2.1 Initialisation

The CMM performs a one time initialisation of the PMM after it has been initialised by the loader. Whether the PMM returns an error code or later makes an upcall is an implementation choice. The PMM must initialise its LU, its transaction program (TP) name, and its mode name, or this could have been done manually before the program is started.

P_INIT

The PMM appears as an APPC Transaction Program (TP). The TP is defined using the well-known TP name 0x28F0F0F1. The conversation type is basic.

When the LU name is algorithmically generated from the user address, the user address (and any necessary mapping table) must be set before the initialisation call.

The values that the SNA PMM assigns to the fields of this structure are defined below.

This chapter lists the characteristics that the SNA transport provider supports. The compensations needed depend on the characteristics required by the transport user.

Note: In some cases (particularly for termination) compensations may be needed even when the characteristics of the transport user match those of the transport provider.

Transport Provider Characteristic	Value assigned by SNA PMM
prov_tsdu	**INFINITE**
prov_etsdu	**86**
prov_send_maxdgm	**INFINITE**
Note: Segmenting of large MPTN datagrams is done using LL chaining, not via MPTN datagram segmentation. However, an SNA transport provider must be able to receive either LL chained, or MPTN segmented, datagrams.	
prov_receive_maxdgm	**INFINITE**
prov_protocol_maxdgm	**INFINITE**
addr_service	**Algorithmic or address mapper**
Note: AF_INET Sockets uses algorithmic mapping; NetBEUI uses an address mapper.	
connect_supported	**TRUE**
stream_supported	**FALSE**
record_supported	**TRUE**
expedited_marking	**FALSE**
datagram_supported	**TRUE**
multicast_supported	**FALSE**
simplex_abortive	**FALSE**
simplex_orderly	**TRUE**
duplex_abortive	**TRUE**
duplex_orderly	**TRUE**
session_outage	**TRUE**

Table 11-1 SNA Transport Provider Characteristics

This information corresponds to the PROV_INFO data structure.

After initialisation is complete, the PMM must be ready to receive incoming connections and incoming datagrams.

In terms of SNA calls, the PMM makes a **receive_allocate** call to receive any incoming data.

11.2.2 Naming and Addressing

The description in this section will assist in the understanding of connection setup.

When a transport user requests that a connection be set up non-natively, and the CMM chooses to use the SNA transport provider, the address of the transport user to which the connection must be set up has to be mapped to the SNA address of the machine where that remote transport user resides. This is the address mapping problem.

There are three kinds of address mapping techniques:

- algorithmic
- the use of a protocol-specific directory
- a general purpose MPTN address mapper.

Table-driven mapping is used in conjunction with algorithmic mapping for sockets transport users of the INET address family (AF_INET) type, and a protocol-specific directory is used for NetBEUI transport users.

Details of IP and SNA Addressing

AF_INET sockets employ IP addresses, which are 32-bit numbers. This 32-bit number consists of two fields: a network ID and a host ID. The network ID field identifies a physical network, while the host ID field identifies a particular host interface attached to that network.

The number of bits devoted to the network and host ID portions of an internet address is not fixed for all IP addresses. Rather, the IP address space is partitioned into network classes. The class to which a given IP address belongs is determined by the value of the three highest bits in the address. The class, in turn, determines which of the remaining bits make up the network ID and which are devoted to the host ID.

The use of subnetting also influences the partitioning of an IP address into network and host ID fields. If subnetting is used, an addresses' class does not identify the size of the network portion of the address. Rather, the subnet mask is used for this purpose. For example, a subnet mask of 0xFFFFFE00 indicates that the upper 23 bits of the IP address are the network ID while the lower 9 bits are the host ID.

IP addresses are typically written in *dotted decimal* notation. In this notation, an IP address consists of four decimal numbers separated by periods. Each decimal digit corresponds to one byte of the IP address. For example, the IP address with hexadecimal value of:

```
0x09060F6F
```

appears in dotted decimal notation as:

```
9.6.15.111.
```

SNA addresses consist of a network identifier, an LU name, and a TP name. The network identifier and LU name together are sufficient to identify an access point in a SNA network; these are typically written in the form:

NetID.LUname

where both **netID** and **LUname** may be up to eight characters long.

IP-LU Mapping Mechanism

The address mapping mechanism used is a two step process. First, the network ID portion of the internet address is mapped to a SNA network name and an LUname template. This step is table driven, using a local table. Second, the host ID portion of the internet address is used to generate the remainder of the LU name. This step is algorithmic.

The table driven step in the mapping allows network administrators to specify part or all of the LU names generated by the mapping mechanism. Using the LUname template, administrators can ensure that the LU names conform to local conventions and do not conflict with pre-existing LU names in the network.

The algorithmic portion of the mapping provides a way to greatly reduce the size of the mapping table. This both eases administration and reduces the impact on system resources.

- **Step 1: Table-driven Mapping**

  Each entry in the mapping table contains four fields: an internet network number, a subnet mask, a SNA network name, and an LUname template. For example, consider the mapping table shown in Table 11-2. The first entry in the table indicates that IP addresses whose upper 23 bits have the value 128.109.130 should be mapped to the SNA network USIBMSER. The LUname template is the three character prefix *NRA*; the remaining five characters of the LU name are generated by the algorithmic step.

Network ID	**Subnet Mask**	**Network Name**	**LUname template**
128.109.130	0xFFFFFE00	USIBMSER	NRA
128.109.141.99	0xFFFFFFFF	USIBMSER	NRICKETY

Table 11-2 Mapping Table Example

  The subnet mask only needs to be specified explicitly if it does not match the entry's network ID. If the subnet mask is omitted from this entry, the mask defaults to 0xFFFFFF00 to match the supplied network ID.

  The second entry in the mapping table completely specifies both the IP address of the destination and its LU name. Only references to the destination 128.109.141.99 will match this entry. Completely specified entries are appropriate when administrators require explicit control over the entire LU name, where there are only a few machines using the code in a larger IP subnetwork, or when the code is being added to machines with already defined LU names.

- **Step 2: Algorithmic Mapping**

  The algorithmic mapping step is responsible for filling in the unspecified characters of the LUname template identified in the first mapping step. The algorithm takes the host ID portion of the IP address; this is the n bits of the IP address not used to match a network ID in the mapping table. The algorithm generates the m characters unspecified in the LUname template.

  The number of input bits, n, places a lower bound on the number of output characters, m. As described below, each output character represents five input bits. Thus, for a given n, m must be greater than or equal to $^{n}/_{5}$, rounded up to an integer value. This requirement restricts the number of characters that may be specified in the LUname template field of the mapping table described above. The most restrictive case is that of a class A IP address, which has a 24 bit host ID field. For these IP addresses, the algorithmic mapping step must generate a minimum of five output characters; thus, the LUname template for a class A network ID can only specify up to three of the eight LU name characters.

The algorithmic mapping step uses 32 of the 39 characters allowed in an LU name. In addition, it relies on an established ordering of these 32 characters. The choice of characters and ordering is:

```
0123456789ABCDFGHJKLMNPQRSTVWXYZ.
```

Each character is assigned an index value which is its place in the ordering. For this example, ‘‘0’’ has an index value of zero while ‘‘Z’’ has an index value of 31. The first character of the LU name must be specified by the template (the algorithm could generate a number, and LU names may not begin with a number).

The algorithm operates by first grouping the *n* input bits into *m* groups of five bits each (where *n* is the number of IP address bits to be mapped, and *m* is the number of LUname characters to be generated). If there are insufficient bits to create m groups, then zeroes are prepended to the input n bits to make the number of input bits equal to m*5. Each 5-bit group is then replaced with the character assigned its index value, and the mapping is complete.

- **Mapping Example**

  Given the mapping table of Table 11-2, and the character ordering from above, consider the mapping of the IP address 128.109.131.45. This IP address will match the first table entry (applying the subnet mask in the first entry gives a network ID of 128.109.130), yielding a SNA network name of *USIBMSER* and an LUname template of *NRA*.

  The algorithmic mapping step will take as input the nine bits:

  1 0010 1101

  and will output five characters. After prepending zeroes and grouping the input bits can be represented:

  00000 00000 00000 01001 01101

  The decimal values of the groups are: 0, 0, 0, 9, and 13; the associated characters are: ‘‘0’’, ‘‘0’’, ‘‘0’’, ‘‘9’’ and ‘‘D’’. Thus the IP address:

  128.109.131.45

  will be mapped to the SNA name:

  USIBMSER.NRA0009D

Details of Use of a Nameserver

For name spaces (like NetBEUI’s) that are bigger than the APPC name space, algorithmic mapping is infeasible. In these cases, the SNA transport provider PMM must use an address mapper.

Use of a Well Known TP Name

The TP name of the destination is the same well-known TP name that is used by all boxes running a SNA PMM. It is 0x28F0F0F1.

11.2.3 Datagrams

The user may send datagrams to a remote destination, and datagrams may arrive from a remote destination.

P_SEND_DG

The PMM uses a half-duplex conversation to send datagrams to a particular destination. All datagrams that go to one destination go over the same SNA conversation. Each conversation may be deallocated after a period of inactivity. The data must be flushed.

If the size of the data, including the compensation header, is larger than the maximum size supported by the PMM, the datagram must be segmented. Segmenting of large MPTN datagrams is done using LL chaining, not via MPTN datagram segmentation.

P_RCV_DG

When datagrams are received, a P_RCV_DG upcall is made. If they had been fragmented using MPTN segmentation, the CMM needs to reassemble them before passing them to the transport user. If the datagram had been fragmented using LL concatenation, the PMM must reassemble the fragments before making the upcall.

11.2.4 Connection Setup

There are two kinds of connection setup: active and passive. A connection setup is described as active when the transport user on the local node initiates the connection setup with an M_CONNECT downcall. When a remote node initiates the connection setup by creating an SNA connection to the local LU and then flowing an MPTN_Connect packet, the connection setup is passive.

SNA provides full-duplex conversations and half-duplex conversations. If only half-duplex conversation is available on either side, then twin opposed half-duplex conversations are used to simulate a full-duplex conversation.

When the local node wants to create a connection it makes a TLPB M_CONNECT downcall. The CMM then:

- makes a P_LOCATE downcall if the mapping from the transport user name to the SNA address is not available already
- makes a P_CREATE downcall to create a PMM control block for the connection
- makes a P_CONNECT downcall to request an SNA connection to be set up and an MPTN_Connect to be flowed as the first packet on the connection.

For passive connection setup, the PMM is informed when an SNA connection setup is requested by a remote node. The PMM makes a P_ACCEPT upcall to inform the CMM about the arrival of a connection request, along with the MPTN_Connect packet.

These calls are described in the sections below.

P_LOCATE

The CMM makes P_LOCATE downcalls to discover the transport provider name (SNA address) of the node where a given transport user name (for example, an IP address) resides. This is discussed in detail in Section 11.2.2 on page 145.

P_CREATE

No SNA calls are made in response to a P_CREATE downcall.

P_CONNECT

A P_CONNECT downcall is always preceded by a P_CREATE downcall. The CMM passes the destination address and the MPTN_Connect packet to the PMM. The PMM allocates a conversation and then sends the MPTN_Connect on that conversation.

SNA has the potential to use either full-duplex or half-duplex connectivity between transaction programs. If full-duplex connectivity is available it is tried first; if the full-duplex **associate** fails, then half-duplex **allocate** is tried:

- In the case of full-duplex connectivity, the MPTN_CONNECT response will come over the same connection on which the MPTN_CONNECT request was sent.
- In the case that only half-duplex connectivity is available, the other side will allocate a connection and send the MPTN_CONNECT response on it; half-duplex connections use twin opposed conversations.

P_ACCEPT

The P_ACCEPT upcall is made by the PMM when it receives a connection request from a remote node. Once the PMM is initialised (see Section 11.2.1 on page 143), it makes a **receive_allocate** SNA call to receive connection requests on its LU name. When this **receive_allocate** call unblocks because of an inbound SNA conversation, the PMM calls the SNA **receive_and_wait** function to receive either an MPTN_Connect or MPTN_Datagram, the first packet which flows on the native connection. If an MPTN_Connect is received, a P_ACCEPT_UC is made; otherwise, datagram processing is initiated.

In the case of MPTN_Connect, the CMM decides whether to send an MPTN_Connect_Response or not, and whether it should be positive or negative. The PMM allocates a conversation to the remote destination if necessary, and sends the connect response.

11.2.5 Data Transfer

The CMM makes a P_SEND downcall in response to an M_SEND downcall by the transport user, to send data on an established connection. The M_SEND downcall, as well as the P_SEND downcall, can be used for both normal and expedited data.

When either normal or expedited data comes in on an established connection, the PMM makes a P_RECEIVE upcall to the CMM to give it the data. These calls are described below.

P_SEND

The PMM adds a 1-byte compensation header that will precede the actual data in the line flow, either MPTN_NOCOMP or MPTN_EXP. Then the PMM issues SNA calls to send the header and the data field to the partner. These calls could be a single **SEND_DATA** call or a series of **SEND_DATA** calls (or any other equivalent set of calls). Stream data may be sent either using LL concatenation or using a new 1-byte compensation header for each send. If the entire data could not be sent (because of congestion on the SNA connection), the PMM will queue the data internally and try to send it when the congestion clears.

P_RECEIVE

The P_RECEIVE upcall to the CMM is used when the PMM receives data on an APPC **RECEIVE_AND_WAIT** call. Any data that is received is passed to the CMM. Pacing is discussed in Section 11.2.6.

For a Sockets transport user, if the data is expedited, then additional work must be done. This includes marking the expedited spot, saving the expedited byte, and indicating that the socket has out-of-band data.

If stream data arrives but is concatenated via LL concatenation, it can still be given to the CMM as it arrives (the transport user will perform record reassembly).

11.2.6 Pacing

The transport provider PMM obtains memory buffers on behalf of the TLPB user for carrying data. If the user has no buffers available, the PMM will not continue to receive data for that conversation; it will rely on SNA pacing to prevent problems.

11.2.7 Connection Termination

As in connection setup, connection termination may be initiated by the local or the remote end of a connection. The transport user indicates that a connection is to be terminated by making a TLPB M_CLOSE downcall. The CMM in turn makes a P_CLOSE downcall. When the termination is complete, the PMM responds with a P_CLOSE upcall.

P_CLOSE

The normal case for handling a close request is to send the compensation header, followed by **deallocate (FLUSH)**. In the case of an internal error, either **deallocate (FLUSH)** or **deallocate (ABEND)** may be issued, without first sending any compensation header. The deallocate type chosen on error paths is implementation-dependent.

Request	**Action**
SO_TERM	Send simplex orderly termination. If an SO_TERM has already been received, deallocate the conversation.
DO_TERM	Send queued data; send duplex orderly termination; deallocate the conversation.
DA_TERM	Deallocate the conversation.

Table 11-3 Termination Actions

P_CLOSED

If a compensation header of type=close is received, the PMM expects a **deallocate (FLUSH)** to follow. If a **deallocate** is received without first receiving a compensation header of type=close, the implementation that sent it had already chosen a **deallocate (ABEND)** or **deallocate (FLUSH)**.

There are several incoming conditions that deal with termination:

- The PMM receives an SO_TERM.

  The PMM notifies the CMM that no more data will be arriving on the connection. If the user has already indicated no more data will be sent on the connection, the SNA conversation is deallocated.

- The PMM receives a DO_TERM.

  The PMM notifies the CMM that no more data can be sent or received on the connection, and deallocates the conversation, after sending any data currently queued for transmission.

- The PMM receives a DA_TERM.

  The PMM notifies the CMM that no more data can be sent or received on the connection, deallocates the conversation, discarding any data currently queued for transmission, and returns an error code.

- The PMM receives a deallocation of any type without first receiving a close compensation header.

  The PMM does the same as for DA_TERM, but also returns an error code. The type of deallocation that the PMM uses may be FLUSH or ABEND, depending on the implementation.

11.3 Compensations

The following table identifies the compensations required for specific transport users over the SNA transport provider. The native transport user, SNA, is not shown.

Compensation Value	**Shorthand Name: Compensation Description**	**Sockets**	**NetBEUI**	**OSI**
X'00'	NOCOMP: On a record-oriented transport network, a message with no associated compensation. On a stream-oriented transport network, a record boundary marker (when no other compensation is needed).	Yes	Yes	Yes
X'01'	EXP: Expedited message.	Note 1		Yes
X'03'	EXP_ACK: Expedited message acknowledgement.			
X'10'	DA_TERM: Duplex-abortive termination.			Yes
X'12'	DA_TERM_ACK: Duplex-abortive termination acknowledgement.			
X'14'	DO_TERM: Duplex-orderly termination.	Yes	Yes	
X'16'	DO_TERM_ACK: Duplex-orderly termination acknowledgement.			
X'18'	SA_TERM: Simplex-abortive termination.			
X'1A'	SA_TERM_ACK: Simplex-abortive termination acknowledgement.			
X'1C'	SO_TERM: Simplex-orderly termination.	Yes		
X'1E'	SO_TERM_ACK: Simplex-orderly termination acknowledgement.			
X'20'	RECSEG: Segmented message.		Yes	Yes
X'21'	EXPSEG: Segmented expedited message.			
X'83'	DG_OOB_DATA: MPTN_DG_OOB_Data message needed for expedited data.			

Compensation Value	**Shorthand Name: Compensation Description**	**Sockets**	**NetBEUI**	**OSI**
X'84'	DG_KEEPALIVE: MPTN_DG_KEEPALIVE_Hdr message needed for session outage notification.			

Table 11-4 Compensations Required of SNA Transport Providers

1 The EXP compensation is used to mark where in the normal data the urgent data is located.

Chapter 12

NetBEUI Transport User Characteristics

12.1 Introduction

For an MPTN NetBIOS extended user interface (NetBEUI) transport user, this chapter defines the architecture for the interface between the NetBEUI transport user and MPTN at the Transport Layer Protocol Boundary (TLPB). Specifically, it defines the information that flows at that interface (that is, the verbs and parameters) for the NetBEUI transport user. The component that provides that interface is the NetBEUI-MPTN Syntax Mapper, hereafter referred to as the syntax mapper.

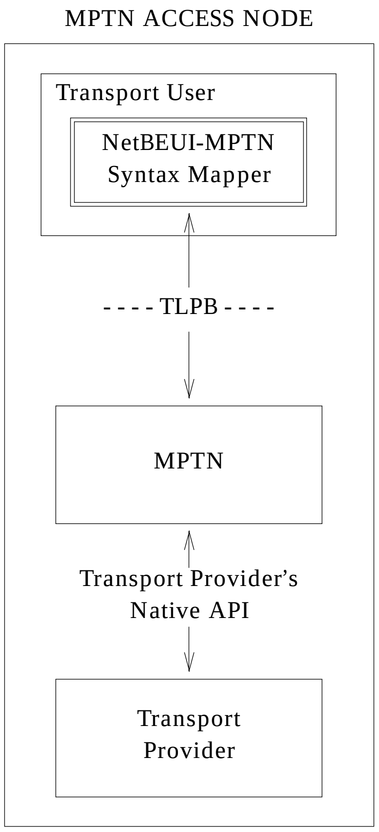


Figure 12-1 The NetBEUI-MPTN Syntax Mapper, from an MPTN Perspective

12.2 Components of the NetBEUI Transport User

Figure 12-2 depicts the NetBEUI-MPTN Syntax Mapper's relationship to other MPTN architecture components.

NetBIOS calls are routed to MPTN or native NetBIOS based on the adapter number provided in the NetBIOS Control Block (NCB).

Multiple adapter numbers may be directed to MPTN. These *virtual* adapters provide the semantics of multiple NetBIOS adapters attached to a single LAN (except for the items noted in Section 12.3.5 on page 158).

NetBIOS calls that are determined to be non-native are mapped into the appropriate calls to the TLPB (see Section 12.8 on page 164).

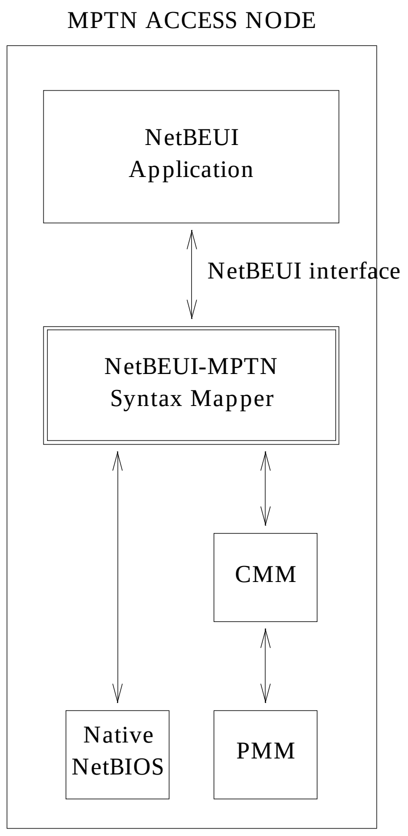


Figure 12-2 The NetBEUI-MPTN Syntax Mapper, from a NetBIOS Perspective

12.3 Design Considerations

The major things that must be considered when designing a NetBEUI Transport User product are:

- NetBIOS names
- NetBIOS **remote_STATUS** command
- flow control
- addresses
- NetBIOS API considerations.

Each of these is discussed in the following sections.

Throughout this chapter, references to NetBEUI operations are highlighted in **bold** type.

12.3.1 NetBIOS Names

Each NetBIOS **ADD_NAME** or **ADD_GROUP_NAME** binds that single NetBIOS name, with name registration requested, to two TLPB control blocks. One control block has an M_LISTEN and the other an M_RCVDG_INIT command posted to it. They serve as the reception point for all connections and datagrams sent to the NetBIOS name. The control blocks are deleted when a **DELETE_NAME** or **RESET** is issued.

If multiple applications add the same group name, only one TLPB control block can be created and bound to that name to accept datagrams and incoming connections, so the applications must share a single TLPB control block.

A NetBIOS Name Number 1 is generated for each adapter number. The name consists of 10 zero bytes, followed by 6 pseudo-random bytes. This name is used as the endpoint for NetBIOS **remote_STATUS** commands (see Section 12.3.2), and may be used by NetBIOS applications, except as a means of determining a physical adapter address.

At initialisation, the mapper sets up a control block and issues an M_RCVDG_INIT_DC for broadcast datagrams in order to receive broadcast datagrams. The CMM then registers the NetBIOS group name MPTNBROADCASTGRP. This name is used as the destination of all NetBIOS broadcast datagrams. Only a single control block is maintained to receive broadcast datagrams.

12.3.2 NetBIOS remote_STATUS Command

The **remote_STATUS** command is implemented by sending an MPTN control datagram to the remote name. The content of the control datagram is, except as noted below, as specified in the IBM LAN Technical Reference: IEEE 802.2 and NetBIOS Application Program Interface (SC30-3587-00), page 5-14, table 5-14, for the **STATUS_QUERY** frame for native NetBIOS. The mapper receiving the **STATUS_QUERY** control datagram replies with the **STATUS_RESPONSE** frame from the IBM LAN Technical Reference above, page 5-22, table 5-21.

The sender name is the NetBIOS Name Number 1 corresponding to the adapter number used to issue the **STATUS** command.

The Data1 field is set to indicate NetBIOS 2.1 format. However, additional **STATUS_QUERY** frames are not sent if a single frame is not big enough for the response; only a single query and response are used.

The query is sent several times if a response is not received, in case of datagram loss.

12.3.3 Flow Control

Only one TLPB M_SEND_DC per connection is issued at a time. Thus, if transport provider flow control causes the PMM to queue the P_SEND_DC calls, the mapper will not continue to send additional data to the TLPB, and NetBIOS **SEND** commands from the application will not complete, until the underlying flow control allows traffic again.

12.3.4 Addresses

MPTN Parameter	Contents
MPTN Qualifier	X'12'
Address Mode	X'01' - individual or X'02' - group
Node Address	16 byte NetBIOS name
Local Address	None

Table 12-1 Address Values

This information corresponds to the ADDR data structure.

Dynamic Address Verification Required?

NetBEUI requires dynamic address verification.

12.3.5 NetBIOS API Considerations

Inherent to MPTN:

SENDs
: Complete as soon as the data has been sent by the transport provider, instead of blocking until the data is received by the partner. This has the additional effects that **SEND_NO_ACK** is equivalent to **SEND**, and Return code 7 is never returned.

FIND_NAME
: Not supported.

UNLINK
: Not supported.

DELETE_NAME
: Names deleted by **DELETE_NAME** remain globally visible until all such sessions using them have been ended. (In native NetBIOS, they appear to be gone, except on the machine where they are still in use.)

12.4 Characteristics of Connections

12.4.1 User Transport Requirements

This section lists the communication characteristics of connections that the NetBEUI transport user needs. Not all of these characteristics necessarily need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 12-4 on page 163.

MPTN Parameter	Contents
Maximum record length	128k - 2 (131,070)
Maximum expedited record length	0 (not needed)
Maximum connection termination data length	0 bytes
Close types	Duplex orderly, duplex abortive
Expedited marking	No

Table 12-2 User Transport Requirements

This information corresponds to the CONN_CHARS data structure.

12.4.2 Service Mode

Any service mode is acceptable.

12.5 Characteristics of Datagrams

The NetBEUI-MPTN Syntax Mapper employs two types of datagrams:

- user datagrams
- control datagrams.

User datagrams are used for datagrams generated by the application program using the NetBIOS commands **SEND_DATAGRAM** and **SEND_BROADCAST_DATAGRAM**, and control datagrams are used to provide the NetBIOS **STATUS** command.

Not all of the characteristics of datagrams will need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 12-4 on page 163.

12.6 Characteristics of Data

This section lists the characteristics of data that the NetBEUI transport user needs. Not all of these characteristics will need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 12-4 on page 163.

Type of Data	**Contents**
Service type	ST_CO (connection-oriented) or ST_CL (connectionless)
Data type	Record
Maximum Length of Connection Data	0 bytes
Maximum Length of Connection Reply Data	0 bytes
Maximum Length of Data on a Datagram	512 bytes
Maximum length of transport user's address	16 (not including the MPTN qualifier)
partial_records	No
Connection characteristics	See Section 12.4 on page 159.
Session outage notification	Yes
In line	No
Non-queued Response sent as expedited	No
dg_seq#_recognized	No
tuser_data_recognized	No
direct_tlpb_type	0 (not direct TLPB user)
Connect-connect	No
Parallel connections	Yes
Not listening notification	Yes

Table 12-3 Types of Data

This information corresponds to the M_INFO data structure.

12.7 Compensations

The following table identifies the compensations required for support of a NetBEUI transport user on specific transport providers[9]. The native transport provider, NetBIOS, is not shown.

Compensation Value	**Shorthand Name: Compensation Description**	**SNA**	**SPX/IPX**
X'00'	NOCOMP: On a record-oriented transport, a message with no associated compensation. On a stream-oriented transport network, a record boundary marker (when no other compensation is needed).	Yes	Yes
X'01'	EXP: Expedited message		
X'03'	EXP_ACK: Expedited message acknowledgement		
X'10'	DA_TERM: Duplex-abortive termination		
X'12'	DA_TERM_ACK: Duplex-abortive termination acknowledgement		
X'14'	DO_TERM: Duplex-orderly termination	Yes	Yes
X'16'	DO_TERM_ACK: Duplex-orderly termination acknowledgement		
X'18'	SA_TERM: Simplex-abortive termination		
X'1A'	SA_TERM_ACK: Simplex-abortive termination acknowledgement		
X'1C'	SO_TERM: Simplex-orderly termination		
X'1E'	SO_TERM_ACK: Simplex-orderly termination acknowledgement		
X'20'	RECSEG: Segmented message	Yes	Yes
X'21'	EXPSEG: Segmented expedited message		

9. NetBIOS over TCP/UDP is defined by RFCs 1001 and 1002, so there are no compensation considerations for that combination. There are no compensation considerations for NetBEUI over OSI because that combination is defined in the MAP/TOP Users Group Technical Report Specification of NetBIOS Interface and Name Service Support by Lower Layer OSI Protocols, Version 1.0, September 27, 1989. This specification can be found in Appendix E of Protocols for X/Open PC Interworking: SMB, Version 2.

Compensation Value	Shorthand Name: Compensation Description	SNA	SPX/IPX
X'83'	DG_OOB_DATA: MPTN_DG_OOB_Data message needed for expedited data.		
X'84'	DG_KEEPALIVE: MPTN_DG_KEEPALIVE_Hdr message needed for session outage notification.		

Table 12-4 Compensations Required to Support NetBEUI Transport Users

12.8 Mapping of NetBIOS Calls to TLPB Verbs

In the following table, *none* means no TLPB calls are needed to implement the NetBIOS call. Some NetBIOS calls require no TLPB downcalls when issued, but are completed by a TLPB upcall.

NetBIOS Verb/Call	TLPB Verb(s)
Names	
ADD_NAME, ADD_GROUP_NAME	Two control blocks are created: • Control block 1: — M_CREATE_DC — M_BIND_DC — M_LISTEN_DC. • Control block 2: — M_CREATE_DC — M_BIND_DC — M_RCVDG_INIT_DC
DELETE_NAME	M_CLOSE_DC
Sessions	
CALL	M_CREATE_DC M_BIND_DC M_CONNECT_DC
HANGUP	M_CLOSE_DC
LISTEN	LISTEN completes when M_ACCEPT_DC completes.
SEND, CHAIN_SEND, SEND_NO_ACK, CHAIN_SEND_NO_ACK	M_SEND_DC
RECEIVE, RECEIVE_ANY	none when issued; RECEIVE or RECEIVE_ANY completes when M_RECEIVE_UC completes
Datagrams	
SEND_DATAGRAM, SEND_BROADCAST_DATAGRAM	M_SEND_DG_DC
RECEIVE_DATAGRAM, RECEIVE_BROADCAST_DATAGRAM	none when issued; RECEIVE_DG, RECEIVE_BROADCAST_DG complete when M_RCV_DG_UC completes.

NetBIOS Verb/Call	TLPB Verb(s)
Miscellaneous	
CANCEL	none (mapper internal operation)
FIND NAME	not implemented
LAN STATUS ALERT	not implemented
RESET	see DELETE_NAME and HANGUP
SESSION STATUS	none (mapper internal operation)
local STATUS	none (mapper internal operation)
remote STATUS	M_SEND_DG_DC (control) M_RCV_DG_UC
TRACE	not implemented
UNLINK	not implemented

Table 12-5 TLPB Mapping

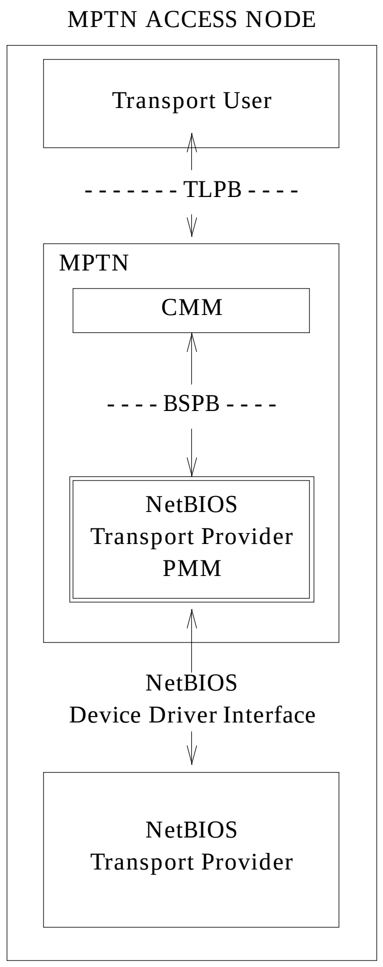
Chapter 13

NetBIOS Transport Provider Characteristics

13.1 Introduction

For an MPTN Network Basic Input / Output System (NetBIOS) Transport Provider, this chapter defines the semantic rules that govern the way the NetBIOS Transport Provider acts when it is invoked by the MPTN CMM. The semantics of the B-Specific Protocol Boundary (BSPB) are defined in terms of the actions taken by the NetBIOS Transport Provider PMM when its BSPB entry point is called by the CMM.

Figure 13-1 illustrates the position of the NetBIOS Transport Provider PMM in relation to the transport user and the transport provider components of the MPTN architecture, and in relation to the NetBIOS protocol stack.



MPTN ACCESS NODE
Transport User
TLPB
MPTN
CMM
BSPB
NetBIOS Transport Provider PMM
NetBIOS Device Driver Interface
NetBIOS Transport Provider

Figure 13-1 The NetBIOS Transport Provider PMM

This chapter describes, for each BSPB request code, the main NetBIOS calls the PMM makes in response to a downcall. Upcalls initiated by external events are also described. All BSPB downcalls are made from the CMM, and all BSPB upcalls are made to the CMM.

13.2 Design Considerations

The major topics that should be considered when designing a NetBIOS transport provider are:

- transport provider characteristics
- initialisation
- name and address mapping
- flow control
- connection termination types.

Each of these is discussed in following sections.

Throughout this chapter, references to NetBIOS operations are highlighted in **bold** type.

13.2.1 Transport Provider Characteristics

The characteristics of the transport provider are collected into the PROV_INFO data structure of the MPTN architecture.

Transport Provider Characteristic	Value assigned by NetBIOS PMM
prov_tsdu	131,070 (that is, 128K - 2) bytes
prov_etsdu	Not applicable
prov_send_maxdgm	512
prov_receive_maxdgm	512
prov_protocol_maxdgm	512
addr_service	Algorithmic
connect_supported	TRUE
stream_supported	FALSE
record_supported	TRUE
expedited_marking	FALSE
datagram_supported	TRUE
multicast_supported	TRUE
simplex_abortive	FALSE
simplex_orderly	FALSE
duplex_abortive	TRUE
duplex_orderly	FALSE
session_outage	FALSE

Table 13-1 NetBIOS Transport Provider Characteristics

13.2.2 Initialisation

At system initialisation, the CMM will issue a P_INIT downcall to the PMM to initialise the local transport endpoint. The CMM may also register addresses using P_REGISTER and P_JOIN_GROUP downcalls. See P_INIT, P_REGISTER, and P_JOIN_GROUP for details.

13.2.3 Naming and Addressing

The NetBIOS transport provider protocol uses 16-byte names to identify its endpoints. There are two types of NetBIOS names: unique and group. A unique name is unique within a physical network, while a group name can be shared among multiple nodes. Thus a unique name is used to identify a particular endpoint. There are a limited number of NetBIOS names available at each node.

IP Address Mapping

IP transport users use a 4-byte IP address to identify a node and a 2-byte port ID to identify an application on that node. The PMM maps the IP node address to a NetBIOS name by converting the 4-byte IP address to an 8-byte upper-case hexadecimal string, and prepending ‘‘MPTN.IN.’’ to form a 16-byte NetBIOS name:

```
Byte:          1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
NetBIOS name:  M P T N . I N . <-- IP addr.-->
```

For example, the IP address 127.87.15.03 would be mapped to the NetBIOS name ‘‘MPTN.IN.7F570F03’’.

The IP broadcast address is mapped to the NetBIOS name ‘‘MPTN.$$BROADCAST’’.

13.2.4 Flow Control

Flow control is implemented by issuing a NetBIOS **RECEIVE** on a session only when prepared to receive data.

Use of **RECEIVE_ANY** complicates flow control by allowing data to be received on any session (since many sessions share one NetBIOS name).

Applying flow control to all sessions indiscriminately (by posting or cancelling **RECEIVE_ANY** calls without using any **RECEIVE** calls) is unacceptable, as some transport users could deadlock under those conditions.

13.2.5 Connection Termination Types

Duplex Abortive is the native termination type of NetBIOS.

13.3 BSPB Considerations

The following BSPB request codes are understood by the NetBIOS PMM. The BSPB request codes are placed in different categories based on the different phases of the life cycle of a connection where they are used.

13.3.1 Initialisation

P_INIT

The PMM reserves resources on each adapter using **RESET**. Upon successful **RESET**, the PMM issues a **STATUS** command to obtain the maximum datagram size to provide to the CMM.

13.3.2 Address Binding and Registration

P_REGISTER

The transport user address is mapped to a NetBIOS name as specified above. The NetBIOS name is added using **ADD_NAME**. Then at least one **LISTEN** is posted to receive any inbound connection requests, and at least one **RECEIVE_DATAGRAM** to receive any incoming datagrams. Whenever one of these commands completes, another is issued.

When there are multiple NetBIOS adapters attached to independent NetBIOS networks, the PMM registers each name on all adapters and fans out requests as appropriate.

P_JOIN_GROUP

The transport user address is mapped to a NetBIOS name as specified above. The NetBIOS name is added using **ADD_GROUP_NAME**. Then at least one **RECEIVE_DATAGRAM** is issued to receive any incoming datagrams. Whenever one of these commands completes, another is issued.

When there are multiple NetBIOS adapters attached to independent NetBIOS networks, the PMM registers each name on all adapters and fans out requests as appropriate.

P_LOCATE

The target transport user address is mapped to a NetBIOS name (see Section 13.2.3 on page 170).

P_QUIT_GROUP

The transport user address is mapped to a NetBIOS name as specified above. The NetBIOS name is deleted using **DELETE_NAME**.

P_DEREGISTER

The transport user address is mapped to a NetBIOS name as specified above. The NetBIOS name is deleted using **DELETE_NAME**.

13.3.3 Datagrams

P_SEND_DG

Generates a **SEND_DATAGRAM**. MPTN headers are prefixed to allow fragmentation and reassembly as needed.

P_RCV_DG

When a datagram is received (a **RECEIVE_DATAGRAM** completes), a P_RCV_DG upcall is generated. A new **RECEIVE_DATAGRAM** is posted as soon as possible.

The MPTN_DATAGRAM header is passed to the CMM with the datagram data, to allow the CMM to handle reassembly of fragmented datagrams.

13.3.4 Connection Setup

P_CREATE

This CMM call does not cause any NetBIOS calls to be made.

P_CONNECT

The P_CONNECT downcall generates a **CALL** command. If the **CALL** is successful, the MPTN_CONNECT is sent to the destination of the connection using **SEND**. **RECEIVE** is used to receive the MPTN_CONNECT response. If the connection is accepted, a **RECEIVE** is posted whenever flow control would allow data to be received on the session. If a negative MPTN_CONNECT response is received, the PMM terminates the session with **HANGUP**.

If the **CALL** fails with NetBIOS return code X'12', it indicates that the destination is present but does not have a **LISTEN** available. The **CALL** could be retried in that case, on the assumption that the other side just missed getting its **LISTEN** up (since a NetBIOS PMM tries to have a **LISTEN** up at all times). Details of retry are implementation-specific.

P_ACCEPT

This upcall is generated whenever a **LISTEN** completes, indicating a incoming connection request. The MPTN_CONNECT is received and passed to the CMM. The MPTN_CONNECT response is sent (to accept or reject the connection) using **SEND**. If the CMM indicates that the connection is to be accepted, the positive MPTN_CONNECT response is sent and **RECEIVE** is posted on the new session as for P_CONNECT. If the CMM indicates that the connection is to be rejected, the negative MPTN_CONNECT response is sent.

A new **LISTEN** is always posted as soon as possible.

13.3.5 Data Transfer

P_SEND

The P_SEND downcall generates a **SEND** command for the data, preceded by the appropriate compensation (see Table 13-2 on page 174).

P_RECEIVE

When data is received on a session, a P_RECEIVE upcall is generated to pass the data to the CMM.

13.3.6 Connection Termination

P_CLOSE

See Section 13.2.5 on page 170. P_CLOSE may generate a **SEND** of a termination compensation, and possibly a **HANGUP** of the session.

P_CLOSED

P_CLOSED is generated when session termination is detected. If no close compensation was received, Duplex Abortive termination is assumed.

13.4 Compensations

The following table identifies the compensations required for specific transport users over the NetBIOS transport provider. The native transport user, NetBEUI, is not shown.

Compensation Value	Shorthand Name: Compensation Description	AF/INET Sockets	SNA	OSI
X'00'	NOCOMP: For record-oriented tspt network, a message with no associated compensation. For stream-oriented tspt network, a record boundary marker (when no other compensation is needed).	Yes	Yes	Yes
X'01'	EXP: Expedited message.	Yes	Yes	Yes
X'03'	EXP_ACK: Expedited message acknowledgement.		Yes	Yes
X'10'	DA_TERM: Duplex-abortive termination.		Yes	Yes
X'12'	DA_TERM_ACK: Duplex-abortive termination acknowledgement.			
X'14'	DO_TERM: Duplex-orderly termination.	Yes		
X'16'	DO_TERM_ACK: Duplex-orderly termination acknowledgement.			
X'18'	SA_TERM: Simplex-abortive termination.			
X'1A'	SA_TERM_ACK: Simplex-abortive termination acknowledgement.			
X'1C'	SO_TERM: Simplex-orderly termination.	Yes		
X'1E'	SO_TERM_ACK: Simplex-orderly termination acknowledgement.			
X'20'	RECSEG: Segmented message.		Yes	Yes
X'21'	EXPSEG: Segmented expedited message.			
X'83'	DG_OOB_DATA: MPTN_DG_OOB_Data message needed for expedited data.		Yes	Yes
X'84'	DG_KEEPALIVE: MPTN_DG_KEEPALIVE_Hdr message needed for session outage notification.		Yes	Yes

Table 13-2 Compensations Required to Support NetBIOS Transport Providers

Chapter 14

SPX/IPX Transport Provider Characteristics

14.1 Introduction

The semantics of the B-Specific Protocol Boundary (BSPB) of an MPTN Transport Provider Manager (PMM) are defined in terms of the actions taken by the SPX / IPX protocol. Familiarity with the BSPB is a prerequisite to understanding this document.

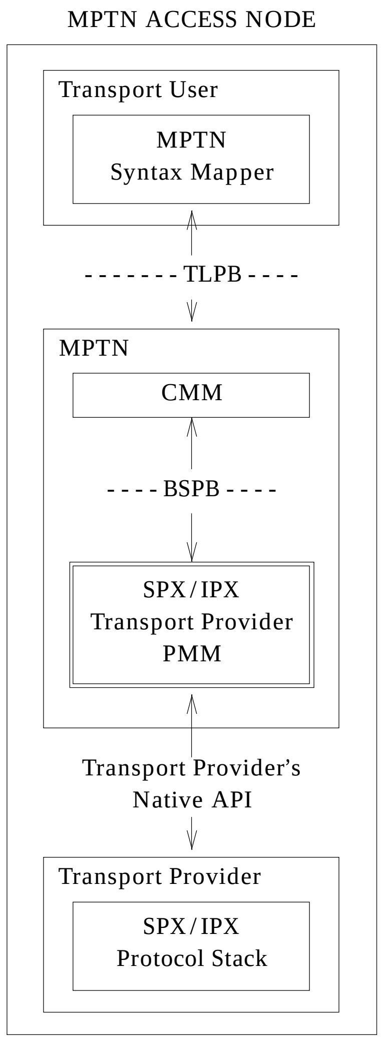


Figure 14-1 The SPX / IPX Transport Provider, from an MPTN Perspective

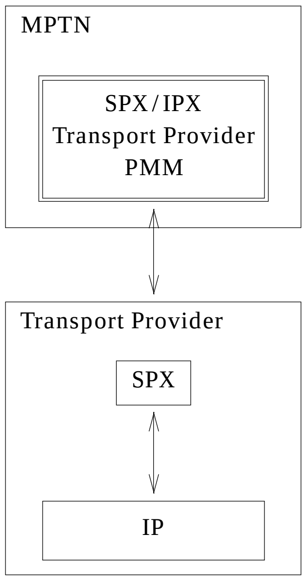


Figure 14-2 The SPX/IPX Transport Provider, from a SPX/IPX Perspective

14.2 Design Considerations

The major topics that should be considered when designing an SPX/IPX Transport Provider product are:

- naming and address mapping
- use of two well-known sockets (one for SPX and one for IPX)
- sending expedited data.

Each of these is discussed in a following section.

14.2.1 Transport Provider Characteristics

This section lists the characteristics that the SPX/IPX transport provider supports. The compensations needed depend on the characteristics required by the transport user. Note that in some cases (particularly for termination) compensations may be needed even when the characteristics of the transport user match those of the transport provider. See the relevant sections of the MPTN Access Node specification for details.

Transport Provider Characteristic	Value assigned by SPX PMM
prov_tsdu	**534 bytes**
Note: It is possible to negotiate for higher prov_tsdu sizes, up to 65535 bytes. However, there are no guarantees that packets larger than 576 bytes (including protocol header) will succeed through the network.	
prov_etsdu	**0**
Note: Urgent data mechanism not available.	
prov_maxdgm	**546 bytes**
Note: Again, negotiable higher, upto 65535 bytes, with no guarantees.	
addr_service	None
connect_supported	**TRUE**
stream_supported	**FALSE**
record_supported	**TRUE**
expedited_marking	**FALSE**
datagram_supported	**TRUE**
multicast_supported	**FALSE**
simplex_abortive	**FALSE**
simplex_orderly	**FALSE**
duplex_abortive	**TRUE**
duplex_orderly	**FALSE**[1]
session_outage	**TRUE**

Table 14-1 SPX Transport Provider Characteristics

This information corresponds to the PROV_INFO data structure.

1 SPX has a duplex orderly close; however, it is not always possible for the partner to determine the close type; that is, whether the close was orderly or abortive.

If there is no receive outstanding at the time the termination arrives, a later receive will get CONN_NOT_FOUND. It is not possible to determine from this if the close was abortive (CONN_ABORT) or orderly (CONN_TERMINATED). There are two cases when a receive will not be posted:

- When pacing buffers are full.
- When a previous receive is being processed.

In order to guarantee that the partner can determine if the close was orderly or abortive, SPX transport providers must perform DO_TERM compensation.

14.2.2 Naming and Address Mapping

When a transport user requests that a connection be set up non-natively, and the CMM chooses to use the SPX transport provider, the address of the transport user to which the connection must be set up has to be mapped to IPX address of the machine where that remote transport user resides. This is the address mapping problem.

There are three kinds of address mapping techniques:

- algorithmic

- the use of a protocol-specific directory
- a general purpose MPTN address mapper.

The most appropriate solution to the name mapping problem is the protocol-independent Address Mapper. Details on the Address Mapper may be found in the Address Mapper specification (see Referenced Documents).

IP-IPX Mapping Mechanism

The 4-byte IPX network address and 6-byte IPX node address are mapped in separate steps. First, the IP address to be converted is compared with the IP Network ID column of the mapping table. The entries of the longest match are used to do the mapping.

If there is an entry in the IPX Network ID column, that is used as the 4-byte IPX Network address. If the entry is blank, a 4-byte Network ID is generated by ANDing the IP Network ID with the IP Subnet Mask. The result is converted from decimal to hex.

If there is an entry in the IPX Node ID column, that is used as the 6-byte IPX node address. If the entry is blank, the first 2-bytes are set to X'4000'. The remaining 4-bytes are generated by converting the decimal IP address to hex.

IP Network ID	IP Subnet Mask	IPX Network ID	IPX Node ID
128.109.130	255.255.0.0	X'01020304'	
20.15	255.255.0.0	X'05060708'	
20.15.35	255.255.0.0		

For example, the IP address 128.109.130.10 would be converted to the IPX address :

X'01020304.4000806D820A'

while the IP address 20.15.35.2 would be converted to:

X'140F0000.4000140F2302'

The IP broadcast address maps to IPX node address X'FFFFFFFFFFFF'.

14.2.3 Use of Well-known Sockets

The address mapping mechanism yields an IPX address (4 bytes of network identifier and 6 bytes of node address) where a given transport user is located. To actually communicate with the PMM on that remote node, a socket number is required. IBM has obtained from Novell, Inc., two well-known sockets for MPTN. Socket 8795H is used by MPTN for SPX, and socket 8796H is used for IPX. This socket number is appended to all IPX addresses to yield the complete address to use to send connection requests (as well as connectionless data) to a remote SPX/IPX PMM.

14.2.4 Sending Expedited data

Since SPX/IPX does not support the notion of expedited data, it is necessary to build in an expedited data capability compensation.

14.3 BSPB Considerations

This section describes the relationship between the BSPB verbs and the SPX/IPX socket calls that usually occur.

P_INIT
The PMM has to open two sockets, one to listen for incoming SPX connection requests, and the other to listen for incoming IPX datagrams. The PMM has to issue an SPXOPENSOCKET call on the MPTN SPX well-known socket number, 8795H, followed by an SPXLISTENFORCONNECTION for connection requests. For datagrams, the PMM needs to issue an IPXOPENSOCKET call on the MPTN IPX well-known socket, 8796H, followed by an IPXRECEIVE.

P_LOCATE
In order for a PMM to locate a destination, it needs to place a request to the Address Mapper. The destination host must register itself with the Address Mapper in order for name resolution to work. Thus, when the CMM and PMM are up on a host, address mappings must be registered with the Address Mapper. When a PMM terminates gracefully, these mappings should be de-registered at the Address Mapper. Address Mapper requests are packaged as datagrams.

P_CREATE
The PMM simply saves the information internally. No socket calls are made in response to a P_CREATE downcall.

P_CONNECT
The *init_user_addr* and the *connect_corr* fields together uniquely identify an MPTN connection. These fields are saved internally for future use, for example, in an out-of-band datagram (OOB DG). The *comp_hdrs* field contains the list of compensations that the PMM will be required to provided for this connection. Since some compensations (such as *session-outage-notification*, abbreviated as SON) may be negotiated down, the final *comp_hdrs* field is passed back to the PMM on the return from the P_CONNECT upcall (see below). Therefore, this particular value of the *comp_hdrs* field is ignored.

In terms of socket calls, the PMM needs to issue a native connection setup call, SPXESTABLISHCONNECTION.

P_ACCEPT_UC
The P_ACCEPT upcall is made by the PMM when it receives a native connection request from a remote node. Once the PMM is initialised (see P_INIT), it makes an SPXLISTENFORCONNECTION socket call to receive connection requests on the well-known SPX socket. When this **listen** call unblocks because of an inbound SPX connection request, the PMM calls the SPXRECEIVESEQUENCEDPACKET socket function to receive the MPTN_Connect, the first packet which flows on the native con- nection, and makes a P_ACCEPT upcall to inform the CMM about the inbound connection request.

P_ACCEPT_DC
The CMM makes a P_ACCEPT downcall in response to the P_ACCEPT upcall. The P_ACCEPT downcall indicates whether the partner's connection request was accepted by the local transport user, and passes the compensations to be performed for this connection and the local transport user characteristics (some of which may be negotiated down values of what the partner requested). The MPTN_Connect RSP packet is also passed down.

Note: The P_ACCEPT downcall may arrive either before or after the P_ACCEPT upcall returns.

On receiving the P_ACCEPT downcall, the PMM calls the SPXSENDSEQUENCEDPACKET socket function to send the MPTN_Connect_RSP to the partner. Then, it sets up internal structures to be able to receive incoming data and to respond to P_SEND downcalls locally.

P_SEND

Before sending the data on the SPX connection, the PMM builds a 1-byte compensation header that will precede the actual data in the line flow. The compensation headers that are used are defined in the formats document. The PMM issues socket calls to send the header and the data field to the partner. These calls could be a single SPXSENDSEQUENCEDPACKET with chained Event Control Blocks, or multiple sends. If the entire data and the headers are successfully sent (that is, the SPX stack accepts the data), the PMM returns RC_OK to the call and the transport user frees the data buffer.

P_RECEIVE

When the PMM realises that data is being sent to it on a given connection, it must read the compensation header and the data itself using the SPXRECEIVESEQUENCEDPACKET socket function. Whether the data is normal or expedited, and whether it is a partial of a whole record, can be decided by examining the compensation byte. Since IPX discards data if it does not fit in the buffer posted in the Event Crontol Block, ‘‘receives’’ must always be done into buffers large enough, possibly as large as the maximum (negotiated) record length.

P_CLOSE

Compensation is required for duplex orderly termination and whenever termination data is to be sent with the close. If there is no termination data to be sent, then the function call SPXABORTCONNECTION can be used for duplex abortive close.

If termination data needs to be sent on a duplex abortive close, the PMM schedules the close to be performed in the background (since it may take time) and returns RC_PENDING to the CMM. The PMM changes its internal state such that it stops receiving any other requests on the connection, initiated by the local or the partner node. Next, it discards any unsent data queued inside the PMM (due to congestion on the SPX connection) that it may have been trying to send in the background. It also issues an SPXCANCELPACKET call to abort any pending sends queued up inside the protocol stack. This enforces the semantics of abortive close locally. Finally, it sends the termination data to the partner, preceded by a 1-byte compensation header that identifies a duplex abortive close request. This request and data must be sent as expedited data, since the semantics of duplex abortive close stipulate that the close request must reach the partner ‘‘immediately’’, even if there is congestion on the connection. Once it is certain that the partner has received the close request and the termination data, the PMM calls the SPXTERMINATECONNECTION socket function to close the SPX connection, and makes a P_CLOSE upcall to signal that the connection is closed. It does not wait for a termination data reply.

If there is termination data on a duplex orderly close, then the data should be sent first, followed by the SPXTERMINATECONNECTION call which will be queued up behind all the pending data to be sent and the termination data.

P_CLOSED

If a connection is terminated abortively, the watchdog timer kicks into action and discovers the termination. It posts a status code in the ECB to denote this event only if there is a listening control block for the socket. Thus, all sockets should have a listening control block. This notification from the watchdog timer can be used to make a P_CLOSED upcall at the passive end of a connection termination.

The P_CLOSED upcall is also made for orderly closes. In such situations, on the return from the upcall, the PMM calls the SPXTERMINATECONNECTION socket function to close the underlying SPX connection, and cleans up all data structures associated with the connection.

14.4 Compensations

The following table identifies the compensations required for the support of specific transport users over the SPX transport provider. The native transport user, SPX, is not shown.

Compensation Value	**Shorthand Name: Compensation Description**	**SNA**	**AF_INET**	**NetBEUI**	**OSI**
X'00'	NOCOMP: On a record-oriented transport network, a message with no associated compensation. On a stream-oriented transport network, a record boundary marker (when no other compensation is needed).	Yes	Yes	Yes	Yes
X'01'	EXP: Expedited message.	Yes	Yes		Yes
X'03'	EXP_ACK: Expedited message acknowledgement.	Yes			Yes
X'10'	DA_TERM: Duplex-abortive termination.	Yes			Yes
X'12'	DA_TERM_ACK: Duplex-abortive termination acknowledgement.				
X'14'	DO_TERM: Duplex-orderly termination.		Yes		Yes
X'16'	DO_TERM_ACK: Duplex-orderly termination acknowledgement.				
X'18'	SA_TERM: Simplex-abortive termination.				
X'1A'	SA_TERM_ACK: Simplex-abortive termination acknowledgement.				
X'1C'	SO_TERM: Simplex-orderly termination.		Yes		
X'1E'	SO_TERM_ACK: Simplex-orderly termination acknowledgement.				

Compensation Value	**Shorthand Name: Compensation Description**	**SNA**	**AF_INET**	**NetBEUI**	**OSI**
X'20'	RECSEG: Segmented message.	Yes		Yes	Yes
X'21'	EXPSEG: Segmented expedited message.				
X'83'	DG_OOB_DATA: MPTN_DG_OOB_Data message needed for expedited data.	Yes			Yes
X'84'	DG_KEEPALIVE: MPTN_DG_KEEPALIVE_Hdr message needed for session outage notification.				Yes

Table 14-2 Compensations Required to Support SPX Transport Providers

Chapter 15

OSI Transport User Characteristics

15.1 Introduction

For an MPTN Open Systems Interconnection (OSI) transport user, this chapter defines the architecture for the interface between the OSI transport user and MPTN at the Transport Layer Protocol Boundary (TLPB). Specifically, it defines the information that flows at that interface for the OSI transport user. The component that provides that information is the OSI-MPTN Syntax Mapper (hereafter referred to as the syntax mapper).

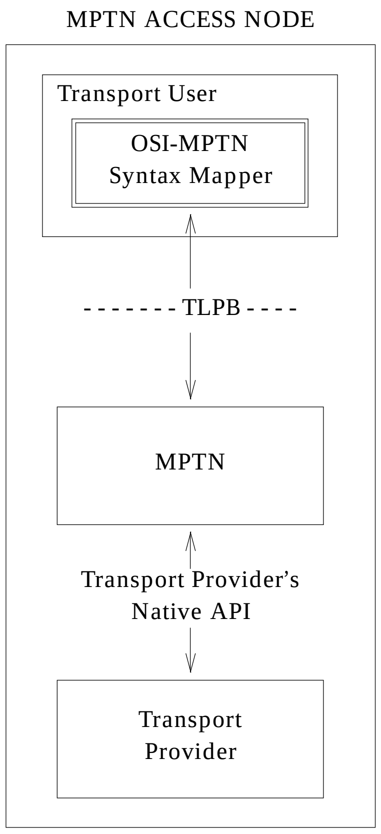


Figure 15-1 The OSI-MPTN Syntax Mapper, from an MPTN Perspective

15.2 Components of the OSI Transport User

Figure 15.2 shows the components involved in OSI session establishment and data transfer and their interfaces to MPTN components.

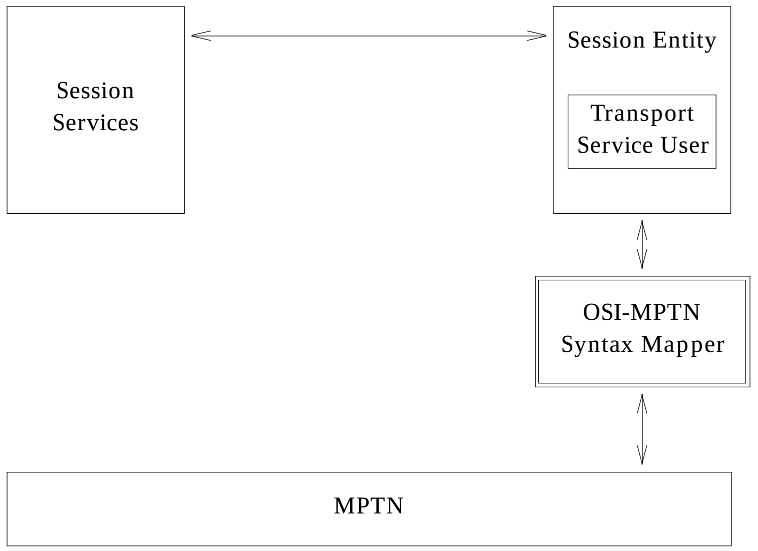


Figure 15-2 The OSI-MPTN Syntax Mapper, from an OSI Perspective

The OSI Transport User consists of:

- the Session Services User (Presentation Entity)
- the Session Entity
- the Transport Service User
- the OSI-MPTN Syntax Mapper.

Without exception, the OSI upper layers operate without change and access MPTN via transport layer interfaces. Thus the OSI-MPTN Syntax Mapper appears to the OSI components as another connection-oriented transport service. It appears to the MPTN components as a transport user, issuing TLPB calls such as M_CONNECT or M_SEND.

When each Transport Selector (that is, each Transport Service Access Point) is defined, it issues a call to the OSI-MPTN Syntax Mapper, which in turn registers that name with MPTN and receives any connection requests coming over MPTN for that Transport Service Access Point.

15.3 Design Considerations

The major things that must be considered when designing an OSI Transport User product are:

- treatment of Protocol Control Information
- transport Connection Establishment
- management of Transport Pacing (that is, credits to peer)
- address mapping.

15.3.1 Treatment of OSI Protocol Control Information

From the viewpoint of the Transport Layer, the OSI transport service user issues four sets of primitives to the OSI-MPTN Syntax Mapper; these are T-Connect, T-Data, T-Expedited-Data, and T-Disconnect. The OSI-MPTN Syntax Mapper encodes each of these into the corresponding transport message of the transport network in use, and transmits it over the MPTN connection to the ultimate destination. The target OSI-MPTN Syntax Mapper decodes the transport message and presents it to the remote transport service user.

As an example of the process, T-Connect request is encoded by the OSI-MPTN Syntax Mapper into an the appropriate connection establishment operation for the transport provider; the MPTN connection ensures reliable delivery to the target OSI-MPTN Syntax Mapper, which decodes the PDU into a T-Connect indication and presents it to its local session service user. The T-PDUs are not built by MPTN.

The OSI-MPTN syntax mapper propagates all Transport Primitives; the mapper does not perform any ‘‘spoofing’’ or emulation on behalf of the MPTN connection partner, as keep alive messages are transparent to the OSI Transport Service user.

15.3.2 Transport Connection Establishment

The OSI-MPTN Syntax Mapper supports only the OSI connection-oriented transport service, and as such the T-Connect/T-Disconnect primitives. It does not support connectionless transport service (and therefore does not support T-Unitdata service primitives).

15.3.3 Flow Control and Buffer Management

No change is necessary for the OSI transport connection relative to flow control when the OSI MPTN Syntax mapper is in use.

In order to control the rate at which the OSI transport services user introduces Session Service Protocol Data Units onto the MPTN provided connection, the OSI MPTN Syntax Mapper manages the transport service locally with the OSI Transport Service User and Session Entity, including the withholding of send permission.

Some transport providers, for example, Open Systems Interconnection and TCP/IP, offer pacing protocols. It is the responsibility of the OSI-MPTN Syntax Mapper to tie the OSI flow control to the transport provider pacing. Therefore, when the syntax mapper issues M_SEND calls that block because the transport connection is not accepting further traffic yet, the syntax mapper will withhold acknowledgements (that is, the functional equivalent of T_Acknowledge) from the transport services user, so that the session entity stops accepting data from the application.

Data may arrive on the transport connection for a particular transport services access point when no OSI buffers within the session entity are available to receive the data; the data will not be received from the transport services user until a session entity buffer is freed. When the application receives the data, the session entity buffer is freed. On the user’s side, the user may try to send data, but the thread may be blocked if no credits are available. The availability of buffers for each session entity is controlled by the MPTN emulation of the OSI transport layer credit/acknowledgement protocol. In any case, the transport services user is completely shielded, and the method of blocking that service until buffers are freed is entirely a local issue.

15.3.4 Addresses

MPTN Parameter	**Contents**
MPTN Qualifier	X'07'
Address Mode	X'01' - individual
Node Address	Network Service Access Point Source address = Source NSAP Destination address = Destination NSAP
Local Address	t selector = transport service access point

Table 15-1 Address Values

Dynamic Address Verification Required?

OSI does not require dynamic address verification. OSI node addresses are defined in such a way that uniqueness is guaranteed. For additional information on this subject see International Standard ISO/IEC 8348/Add.2.

15.4 Characteristics of Connections

15.4.1 User Transport Requirements

This section lists the characteristics of connections that the OSI transport user needs. Not all of these characteristics necessarily need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table in Table 15.4.

MPTN Parameter	Contents
Maximum record length	$2^{32} - 1$ (no limits on record size)
Maximum expedited record length	16 bytes
Maximum connection termination data length	64 bytes
Close types	Duplex abortive, with T-Disconnect sent as expedited data.
In line	No
Expedited marking	Not used.

Table 15-2 User Transport Requirements

This information corresponds to the CONN_CHARS data structure.

15.4.2 Service Mode

MPTN Service Mode

MPTN Service Modes are not used by the OSI MPTN user.

User Service Mode

Any user-defined service mode may be passed as an MPTN user-defined service mode. The user is responsible for assuring that the user service mode is consistent across the entire network.

15.5 Characteristics of Datagrams

Datagrams are not currently used by the OSI Syntax Mapper.

15.6 Characteristics of Data

This section lists the characteristics of data that the OSI transport user needs. Not all of these characteristics will need compensation in the transport provider. The compensations needed can be determined from the specifications of the particular transport provider that will be employed, and the compensation table found in Table 15.4.

Type of Data	**Contents**
Service type	ST_CO (connection-oriented)
Data type	Record
Maximum Length of Connection Data	32 bytes
Maximum Length of Connection Reply Data	32 bytes
Maximum Length of Data on a Datagram	0 (Datagrams not used by OSI transport user.)
Maximum length of Transport User's address	32 bytes
Partial Records	Yes
Connection Characteristics	See User Transport Requirements, above.
Session outage notification	Yes
In-line	No
Non-Queued Responses sent expedited	No
dg_seq#_recognized	No
tuser_data_recognized	No
direct_tlpb_type	0 (not direct TLPB user)
Connect-connect	No
Parallel connections	Yes
Not listening notification	Yes

Table 15-3 Types of Data

This information corresponds to the **M_INFO** data structure.

15.7 Compensations

The following table identifies the compensations required for specific transport providers.[10]

Compensation Value	**Shorthand Name: Compensation Description**	**SNA**	**SPX/IPX**
X'00'	NOCOMP: On a record-oriented transport, a message with no associated compensation. On a stream-oriented transport network, a record boundary marker (when no other compensation is needed).	Yes	Yes
X'01'	EXP: Expedited message.	Note 1	Yes
X'03'	EXP_ACK: Expedited message acknowledgement.		Yes
X'10'	DA_TERM: Duplex-abortive termination.	Yes	Yes
X'12'	DA_TERM_ACK: Duplex-abortive termination acknowledgement.		
X'14'	DO_TERM: Duplex-orderly termination.		
X'16'	DO_TERM_ACK: Duplex-orderly termination acknowledgement.		
X'18'	SA_TERM: Simplex-abortive termination.		
X'1A'	SA_TERM_ACK: Simplex-abortive termination acknowledgement.		
X'1C'	SO_TERM: Simplex-orderly termination.		
X'1E'	SO_TERM_ACK: Simplex-orderly termination acknowledgement.		
X'20'	RECSEG: Segmented message.	Yes	Yes
X'21'	EXPSEG: Segmented expedited message.		
X'83'	DG_OOB_DATA: MPTN_DG_OOB_Data message needed for expedited data.		Yes

10. The native transport provider, OSI, is not shown.

Compensation Value	**Shorthand Name: Compensation Description**	**SNA**	**SPX/IPX**
X'84'	DG_KEEPALIVE: MPTN_DG_KEEPALIVE_Hdr message needed for session outage notification.		Yes

Table 15-4 Compensations Required to Support OSI Transport Users

1 The compensations for the NetBIOS and IPX transport providers are to be added.

2 FDX LU 6.2 supports native SNA expedited data, HDX LU 6.2 does not. An SNA Transport Provider is defined to use FDX LU 6.2. Thus, the only time expedited compensation is needed is when two HDX LU 6.2 connections are used to simulate FDX.

15.8 Mapping to TLPB Verbs

OSI Verb/Call	TLPB Verb
Downcalls:	
Create a transport endpoint	M_CREATE_DC
Request registration	M_BIND_DC
Listen for incoming connection	M_LISTEN_DC
Sending OSI T-Connect response	M_ACCEPT_DC
Sending OSI T-Connect request	M_CONNECT_DC
Sending OSI T-Disconnect	M_CLOSE_DC (abortive)
Sending OSI T-Data or T-Expedited-Data	M_SEND_DC
Deregistration request	M_UNBIND_DC
Upcalls:	
Registration complete	M_BIND_UC
Receiving OSI T-Connect confirmation	M_CONNECT_UC
Receiving OSI T-Connect indication	M_ACCEPT_UC
local after completing M_CLOSE_DC	M_CLOSE_UC
Receiving OSI T-Disconnect indication	M_CLOSED_UC
Receiving OSI T-Data or T-Expedited-Data	M_RECEIVE_UC

Table 15-5 Mapping to TLPB Verbs

15.9 OSI Transport Service Primitives Supported

The following OSI transport service primitives are supported by the OSI-MPTN Syntax Mapper:

- T-Connect
- T-Disconnect
- T-Data
- T-Expedited-Data.

X/Open CAE Specification

Part 4:

Supplementary Information

X/Open Company Ltd.

Appendix A

Data Structures

This appendix defines some of the more important MPTN data structures in detail. For each structure, a table is given indicating the name of the structure and a list of the fields the structure contains. Those fields are then given a short definition.

A.1 ADDR

This data structure is used to specify the parameters of a particular address.

ADDR	*Fields*	mptn_qual addr_mode node_addr_len node_addr local_addr_len local_addr

mptn_qual
Indicates the address family in use. See the **MPTN Data Formats** Specification (Section 8.3.1, MPTN_Qualifier).

addr_mode
Indicates the type of address in use. See the **MPTN Data Formats** Specification (Section 8.3.2, Address Mode).

node_addr_len
Specifies the length of the node address.

node_addr
The node address.

local_addr_len
Specifies the length of the local address.

local_addr
The local address.

A.2 CONN_CHARS

This data structure contains the user transport requirements for a connection. It is ignored if the *servtype* parameter of the M_INFO data structure is connectionless (ST_CL).

CONN_CHARS	*Fields*	tsdu etsdu discon close_types expedited_marking

tsdu
Indicates the maximum record length that a transport user will send or receive.

etsdu
: Indicates the maximum expedited record length that a transport user will send or receive. This parameter should be set to zero if expedited data is not supported.

discon
: Indicates the maximum length of any connection termination data which may flow on the MPTN connection termination.

close_types
: This parameter has four flags that indicate the transport users choice of close sequences. Any combination of flags may be set. The flags are:

- SIMPLEX_ABORTIVE
- SIMPLEX_ORDERLY
- DUPLEX_ABORTIVE
- DUPLEX_ORDERLY

expedited_marking
: Indicates that the transport user wants the ability to mark the position in the normal data where expedited data was sent. This field is ignored if etsdu=0.

A.3 M_INFO

This data structure is used to indicate the transport service type and transport characteristics associated with a transport user.

M_INFO	*Fields*	servtype
		datatype
		connect
		creplen
		maxdgm
		addr
		partial_records
		conn_chars
		datagram_segmentation
		tu_conn_id
		session outage notification
		in_line
		dg_seq#_recognized
		tuser_data_recognized
		direct_tlpb_user_type
		connect_connect
		parallel_connections
		not_listening_notification

servtype
: Indicates the type of MPTN services the transport user requires. Valid values for the servtype parameter are ST_CO (connection-oriented) or ST_CL (connectionless).

datatype
: Indicates the type of data to be sent on the connection, either stream or record. Valid values are DT_STREAM (stream type) or DT_RECORD (record type). This field is ignored if the parameter *servtype* has the value ST_CL.

connect
: Indicates the maximum length of connection data in the conn_data parameter of the M_CONNECT_DC. A value of zero indicates no connection data will be sent. This field is ignored if servtype is ST_CL.

creplen
: Indicates the maximum length of connection reply data sent by the transport user. This field is ignored if servtype is ST_CL, or if servtype is ST_CO and connect=0.

maxdgm
: Indicates to the CMM the maximum length of datagrams that the transport user will send or receive. This field is ignored if the servtype parameter is ST_CO.

addr
: Indicates the maximum length of the transport user's address.

partial_records
: Indicates whether the CMM can pass incomplete records to the transport user in the M_RECEIVE_UC, and whether the transport user may pass incomplete records to the CMM in the M_SEND_DC. If the value is FALSE, the CMM must pass only complete records to the transport user. The two ends of a connection can have different values for this parameter. This field is ignored if the *servtype* parameter has the value ST_CL.

conn_chars
: Indicates the user transport requirements if the servtype parameter is ST_CO. See Section A.2 on page 197.

datagram_segmentation
: Indicates whether the transport user supports datagram segmentation. This field is ignored if servtype is ST_CO.

tu_conn_id
: This is the transport user connection identifier. It is passed on subsequent upcalls to identify the target transport user connection endpoint. This field is ignored if servtype is ST_CL.

session outage notification
: Indicates whether session outage notification is required. This field is ignored if servtype is ST_CL.

in_line
: Indicates that this Transport User's expedited should be sent in line (that is, with the rest of the normal data). It should not be sent using the Transport Providers expedited facility.

dg_seq#_recognized
: Indicates whether the transport user recognises sequence numbers on MPTN Datagram. If dg_seq#_recognized=No, and the CMM receives as datagram with a sequence number on it, the CMM will reject the datagram and send a negative response back to the originator.

tuser_data_recognized
: Indicates whether the transport user recognises transport user data. If tuser_data_recognized=No, and the CMM receives as MPTN format that includes transport user data, the CMM will reject the format and send a negative response back to the originator.

direct_tlpb_user_type
: For a direct TLPB user, indicates the type direct TLPB user this is. Valid values include:

0 = Not a direct TLPB user
1 = Reserved

connect_connect
: Indicates whether the transport user supports connect/connect. This field is ignored if servtype is ST_CL, or if servtype is ST_CO and parallel_connections=yes.

parallel_connections
: Indicates whether the transport user supports parallel connections between the same address pair. This field is ignored if servtype is ST_CL.

not_listening_notification
: Indicates whether the transport user requires a notification (that is, MPTN_CONNECT(-rsp)) if the requested partner is not listening for incoming connections. This field is ignored if servtype is ST_CL.

A.4 PROV_INFO

This data structure contains the characteristics of a transport provider.

PROV_INFO	*Fields*	conn_loc_addr dg_loc_addr prov_tsdu prov_etsdu prov_send_maxdgm prov_receive_maxdgm prov_protocol_maxdgm addr_service connect_supported stream_supported record_supported expedited_marking datagram_supported multicast_supported simplex_abortive simplex_orderly duplex_abortive duplex_orderly prov_id session_outage_notification sptn_info

conn_loc_addr
: The well known local address for connections.

dg_loc_addr
: The well known local address for datagrams.

prov_tsdu
: The maximum unsegmented record size supported by the transport provider, including the MPTN header.

prov_etsdu
: The maximum unsegmented expedited data size supported by the transport provider.

prov_send_maxdgm
: The maximum unsegmented datagram size that can be sent by this instance of the transport provider.

prov__receive_maxdgm
: The maximum unsegmented datagram size that can be received by this instance of the transport provider. This value is expected to be equal to or greater than prov_protocol_maxdgm.

prov_protocol_maxdgm
: The largest datagram guaranteed to be receivable by any implementation of this transport protocol used by the PMM.

addr_service
: The type of address service supported by the transport provider.

connect_supported
: Indicates whether connection-oriented transport services are supported

stream_supported
: Indicates whether stream data is supported.

record_supported
: Indicates whether record data is supported.

expedited_marking
: Indicates whether expedited marking is supported.

datagram supported
: Indicates whether connectionless transport service is supported.

multicast_supported
: Indicates whether multicast operation is supported. Valid values are:

No	Multicast is not supported
Yes	Multicast is inherently supported, that is, it is supported as a function of this transport provider protocol.
MCS	Multicast is supported via a Multicast Server.

simplex_abortive
: Indicates whether simplex-abortive close is supported.

simplex_orderly
: Indicates whether simplex-orderly close is supported.

duplex_abortive
: Indicates whether duplex-abortive close is supported.

duplex_orderly
: Indicates whether duplex-orderly close is supported.

prov_id
: Indicates the type of transport provider.

session_outage_notification
: Indicates whether the provider supports timely notification of session outages.

sptn_info
: The connection network name needed for support of DLUR. If DLUR is not supported, this field is null.

A.5 SERVICE_MODE

This data structure is used to specify the quality of service parameters for a connection. See the **MPTN Data Formats** Specification (Section 8.4, MPTN Service Modes).

SERVICE_MODE	*Fields*	mptn_mode len userdef_mode

mptn_mode
: The architected service mode. See the **MPTN Data Formats** Specification (Section 8.4, MPTN Service Modes).

len
: Length of the parameter userdef_mode.

userdef_mode
: User defined service mode (when present). Format is ASCII character string.

Appendix B

TLPB Verbs

This appendix defines each of the TLPB verbs in detail. Parameters are grouped as *input*, *output* or *input/output*. Input parameters are set by the caller, and read by the called routine. Output parameters are set by the called routine and read by the caller after the call returns. Input/output parameters are set and read in both directions.

Verbs are listed in the heading by their generic names. Different forms of the verb (upcall or downcall) are shown in separate tables which list the parameters of the verb. Upcalls and downcalls are distinguished in the tables by the following symbols:

(↑) Upcalls.

(↓) Downcalls.

After each table, the parameters listed are defined. For all parameters described as addresses, see Section A.1 on page 197 for details.

B.1 M_ACCEPT

The verb M_ACCEPT_UC indicates to the transport user the arrival of a connection request from a remote matching transport user.

M_ACCEPT_UC (↑)	*Input Parameters*	cmm_conn_id conn_data init_user_addr targ_user_addr prov_id service_mode transport_user_data
	Input/output Parameters	tu_conn_id conn_chars
	Output Parameters	none

Parameters

cmm_conn_id
: CMM's connection identifier. This will be used on subsequent downcalls to identify the target CMM connection endpoint.

conn_data
: The transport user-specific data received as part of connection processing from the matching transport user's M_CONNECT_DC.

init_user_addr
: The address of the remote matching transport user initiating the connection.

targ_user_addr
: The address of the transport user that is the target of the connection request.

prov_id
: Identifies the transport provider used for this connection.

service_mode
: The service mode requested.

transport_user_data
: This is optional transport user data that is exchanged between syntax mappers as part of the connection set up. Refer to each individual Transport User Characteristic chapter to see if, and how, this field is used by a particular transport user.

tu_conn_id
: This is the listening transport user connection identifier. It is passed on subsequent upcalls to identify the transport user endpoint connection.

  On return from the upcall, this parameter may be changed by the transport user to identify a different endpoint. This is the transport user connection identifier the CMM will use on subsequent upcalls to identify the target transport user.

conn_chars
: Identifies a set of characteristics for a connection. See Section A.2 on page 197.

The verb M_ACCEPT_DC indicates to the CMM the acceptance/rejection of an incoming connection request from a remote matching transport user.

M_ACCEPT_DC **(↓)**	*Input Parameters*	cmm_conn_id status conn_reply conn_corr conn_char tu_conn_id transport_user_data
	Output Parameters	none

Parameters

cmm_conn_id
: The CMM's connection identifier.

status
: Status of the previous M_Accept_UC, indicates whether the transport user is accepting, or rejecting the connection.

conn_reply
: Any transport user-specific data returned as part of the connection set-up reply. If no data is supplied, the length of the data structure will be set to zero.

conn_corr
: A unique identifier that can be used to retrieve information about this specific connection (for example, for network management facilities).

conn_chars
: Connection characteristics, possibly updated by the transport user.

tu_conn_id
: This is the transport user connection identifier the CMM will use on subsequent upcalls to identify the target transport user.

transport_user_data
: This is optional transport user data that is exchanged between syntax mappers as part of the connection set up. Refer to each individual Transport User Characteristic chapter to see if, and how, this field is used by a particular transport user.

B.2 M_BIND

This verb associates a local address with a control block and registers that address with the MPTN network.

M_BIND_DC (↓)	*Input Parameters*	cmm_conn_id local_user_addr local_user_data regist
	Output Parameters	none

Parameters

cmm_conn_id
: The CMM's connection identifier.

local_user_addr
: The transport user's address.

local_user_data
: User data that is to be registered along with the transport user's address.

regist
: Indicates whether or not the address should be registered externally with a directory server or address mapper.

M_BIND_UC (↑)	*Input Parameters*	tu_conn_id local_user_addr status
	Output Parameters	none

Parameters

tu_conn_id
: The transport user's connection identifier.

local_user_addr
: The transport user's address.

status
: The result of the bind attempt.

B.3 M_CLOSE

This verb terminates an already established connection in the manner specified by the close type indicated. It can be issued by either the initiator or the target of the connection.

M_CLOSE_DC (↓)	*Input Parameters*	cmm_conn_id term_data close_type linger_time
	Output Parameters	none

Parameters

cmm_conn_id
The CMM's connection identifier.

term_data
The transport user-specific data that is sent as part of the termination processing.

close_type
Indicates the termination type requested. Valid values are:

- SIMPLEX_ABORTIVE
- SIMPLEX_ORDERLY
- DUPLEX_ABORTIVE
- DUPLEX_ORDERLY
- CLOSE_CB (close a non-connected communication end point).

linger_time
Time to linger before issuing close. This allows data that might be in send buffers to be sent before the close is issued.

M_CLOSE_UC (↑)	*Input Parameters*	tu_conn_id status
	Output Parameters	none

Parameters

tu_conn_id
The transport user's connection identifier.

status
The result of the M_CLOSE_DC processing.

B.4 M_CLOSED

This verb notifies the transport user of a connection termination initiated externally to the local node.

M_CLOSED_UC **(↑)**	*Input Parameters*	tu_conn_id term_data close_indicator
	Output Parameters	none

Parameters

tu_conn_id
: The transport user's connection identifier.

term_data
: The transport user-specific termination data sent by the transport user initiating the termination.

close_indicator
: Indicates how the connection was closed. Valid values are:

- CI_SIMPLEX_ABORTIVE
- CI_SIMPLEX_ORDERLY
- CI_DUPLEX_ABORTIVE
- CI_DUPLEX_ORDERLY
- CI_ABNORMAL

B.5 M_CONFLICT

This verb notifies the transport user of a conflict between one of its addresses and the address used by another transport user.

M_CONFLICT_UC **(↑)**	*Input Parameters*	tu_conn_id local_user_addr
	Output Parameters	none

Parameters

tu_conn_id
: The transport user's connection identifier

local_user_addr
: The transport user's address that is in conflict with another transport user.

B.6 M_CONNECT

This verb is used by a transport user to establish a connection with a matching transport user at a specified address.

M_CONNECT_DC (↓)	*Input Parameters*	cmm_conn_id targ_user_addr service_mode conn_data conn_corr time_to_live transport_user_data
	Output Parameters	none

Parameters

cmm_conn_id
The CMM's connection identifier.

targ_user_addr
The address of the matching transport user that is the target of the connection attempt.

service_mode
The service mode which is requested for the connection. See Section A.5 on page 202 and the **MPTN Data Formats** Specification (Section 8.4, MPTN Service Modes).

conn_data
The transport user-specific data sent as part of connection processing, provided by the initiating transport user (for example, the BIND image of an SNA transport user).

conn_corr
A unique identifier that can be used to retrieve information about a specific connection (for example, for network management facilities).

time_to_live
Time to live. Setting time_to_live=0 means use the default MPTN TTL value of TTL=42.

transport_user_data
This is the optional transport user data that is exchanged between syntax mappers as part of the connection set up. Refer to each individual Transport User Characteristic chapter to see if, and how, this field is used by a particular transport user.

M_CONNECT_UC (↑)	*Input Parameters*	tu_conn_id conn_reply conn_chars status prov_id transport_user_data
	Output Parameters	none

Parameters

tu_conn_id
: The transport user's connection identifier.

conn_reply
: Transport user-specific data provided as part of the reply to the connection request (for example, the bind response of a SNA transport user).

conn_chars
: The set of characteristics that were returned on the connection by the matching transport user. See Section A.2 on page 197.

status
: The result of the M_CONNECT processing.

prov_id
: Identifies the transport provider used for this connection.

transport_user_data
: This is the optional transport user data that is exchanged between syntax mappers as part of the connection set up. Refer to each individual Transport User Characteristic chapter to see if, and how, this field is used by a particular transport user.

B.7 M_CREATE

This verb creates an MPTN control block.

M_CREATE_DC (↓)	*Input Parameters*	dom proto user_upcall req
	Output Parameters	cmm_conn_id

Parameters

dom
: Specifies the type of TLPB user:

- Syntax Mapper:
  - — SNA
  - — Old SNA
  - — TCP / IP Socket
  - — NetBIOS, and so on.
- Direct TLPB user.

proto
: Indicates which transport provider is to be used by this transport user. A value of B_MPTN indicates that subsequent TLPB calls may use any of the installed transport providers.

user_upcall
: Entry point of the transport user.

req
: Indicates the transport service type(s) and characteristics that must be supported for this control block. See Section A.3 on page 198.

cmm_conn_id
: The CMM's connection identifier. This will be used on subsequent downcalls to identify the target CMM connection endpoint.

B.8 M_LISTEN

This verb indicates that the transport user is willing to receive an incoming connection request.

M_LISTEN_DC (↓)	*Input Parameters*	cmm_conn_id multiple
	Output Parameters	none

Parameters

cmm_conn_id
: The CMM's connection identifier.

multiple
: Indicates whether or not the transport user wants to listen for multiple connections. A value of TRUE indicates that the transport provider should continue listening for multiple connection attempts. A value of FALSE indicates that the transport provider should stop listening after the first connection request.

B.9 M_REPORT_INFO_UC

This verb passes initialisation information to the transport user.

M_REPORT_INFO_UC	*Input Parameters*	connection_network_name report_info_key
	Output Parameters	none

Parameters

connection_network_name
: The name of the connection network(s) that this access node is attached to.

report_info_key
: The MPTN_Qualifier of this transport user.

B.10 M_RCV_DG

This verb indicates that a datagram from a remote transport user has arrived.

M_RCV_DG_UC (↑)	*Input Parameters*	tu_conn_id dg_type sequence# incomplete_segment segment_offset total_dg_len init_user_addr targ_user_addr corr_suffix data broadcast status
	Output Parameters	none

Parameters

tu_conn_id
: The transport user's connection identifier.

dg_type
: Indicates whether this is a user datagram or a control datagram.

sequence#
: Sequence number of this datagram. If a sequence number is not needed, enter 0.

incomplete_segment
: Indicates that the datagram has been segmented.

segment_offset
: Offset of this segment if the datagram has been segmented.

total_dg_len
: Total length of the datagram. Used if this datagram has been segmented.

init_user_addr
: The address of the remote matching transport user.

targ_user_addr
: The address of the local transport user.

corr_suffix
: Correlator suffix used if the transport user reassembles datagram segments.

data
: The data received.

broadcast
: Indicates whether or not the datagram received was from a broadcast.

status
: Indicates whether the datagram has been truncated.

B.11 M_RCVDG_INIT

This verb indicates that the transport user is willing to receive incoming datagrams.

M_RCVDG_INIT_DC **(↓)**	*Input Parameters*	broadcast multiple
	Output Parameters	none

Parameters

broadcast
: Indicates whether or not the transport user will accept datagrams sent to its broadcast address. A value of FALSE indicates that the transport user wishes only to receive datagrams addressed to its unique address or to its group address.

multiple
: Indicates whether or not the transport user wants to receive multiple datagrams. A value of TRUE indicates that the CMM should accept all datagrams until the control block is destroyed. A value of FALSE indicates that the CMM should stop receiving datagrams after one has been received.

B.12 M_RECEIVE

This verb indicates to the transport user that data has arrived on a connection.

M_RECEIVE_UC **(↑)**	*Input Parameters*	tu_conn_id status data expedited_data expedited marker incomplete_record
	Output Parameters	none

Parameters

tu_conn_id
: The transport user's connection identifier.

status
: The status of the receive.

data
: The data received.

expedited_data
: Indicates whether or not the data received is expedited. Expedited data will be passed only on record boundaries if the datatype is record.

expedited_marker
: A marker provided by the CMM that correlates an expedited message with the position in the normal data field at which it was sent. The same correlator value is used on the expedited message and the corresponding normal data message.

incomplete_record
When record data is being received, this parameter indicates whether or not the data being passed to the transport user completes a record. Used if the transport user supports partial records.

B.13 M_SEND

This verb is issued by the transport user to send data on a connection.

M_SEND_DC **(↓)**	*Input Parameters*	cmm_conn_id data expedited_data incomplete_record flush
	Output Parameters	none

Parameters

cmm_conn_id
The CMM's connection identifier.

data
The data to be sent.

expedited_data
Indicates whether or not the data being passed is expedited data.

The amount of data being sent may exceed the maximum expedited record length for this Transport User. When this occurs, the last x bytes (where x is the maximum expedited record length for this Transport User) are considered expedited. Any preceding bytes are treated as normal data.

incomplete_record
If the value is TRUE, it indicates that the data being passed is an incomplete record and more will follow. If the value is FALSE, it indicates that the data being passed is either a complete record or completes a previous partial record.

This parameter must always be FALSE if record segmentation is not supported on this connection, or if the data type is stream.

flush
A value of TRUE in this parameter indicates a request to the CMM to send this data and any data that is queued to the PMM as quickly as possible. A positive return code does not imply that this has been done, only that the CMM has committed to do it.

B.14 M_SEND_DG

This verb is issued by the transport user to send a datagram.

M_SEND_DG_DC **(↓)**	*Input Parameters*	cmm_conn_id dg_type sequence# retry broadcast targ_user_addr service_mode data dg_segment total_dg_len time_to_live
	Output Parameters	none

Parameters

cmm_conn_id
: The CMM's connection identifier.

dg_type
: Indicates whether this is a user datagram or a control datagram.

sequence#
: Sequence number of this datagram. If a sequence number is not needed, enter 0.

retry
: Indicates that the CMM should discard any cached address information.

broadcast
: Indicates that this datagram is to be broadcast.

targ_user_addr
: The address of the remote matching transport user.

service_mode
: The service mode requested for sending the datagram.

data
: The data to be sent.

dg_segment
: Indicates whether this is the first, middle, last, or only segment of a segmented datagram. If the transport user does not support datagram segmentation, this parameter will indicate that this is the only datagram segment.

total_dg_len
: The total datagram length. Used if the datagram has been segmented.

time_to_live
: Time to live: setting time_to_live=0 means use the default MPTN TTL value of TTL=42.

M_SEND_DG_UC (↑)	*Input Parameters*	status sequence# data
	Output Parameters	none

Parameters

status
: Indicates the status of the datagram send.

sequence#
: Sequence number of this datagram. A value of 0 indicates that sequence numbers are not being used.

data
: The data buffer is issued or the M-Send_DG_DC is returned.

B.15 M_UNBIND

This verb requests the removal of the association between an MPTN control block and a transport user address. If that control block is the last or only control block to be associated with that address, this verb asks the CMM to remove the transport user's address from its' tables and to delete any prior network directory or address mapper registration.

M_UNBIND_DC **(↓)**	*Input Parameters*	cmm_conn_id
	Output Parameters	none

Parameters

cmm_conn_id
    The CMM's connection identifier.

Appendix C

BSPB Verbs

This Appendix defines each of the BSPB verbs in detail. Parameters are grouped as either *input*, *output* or *input/output*. Input parameters are set by the caller, and read by the called routine. Output parameters are set by the called routine and read by the caller after the call returns. Input/output parameters are set and read in both directions.

Verbs are listed in the heading by their generic names. Different forms of the verbs (upcall or downcall) are shown in separate tables which list the parameters of the verb. Upcalls and downcalls are distinguished by the following symbols:

(↑) Upcalls.

(↓) Downcalls.

After each table, the parameters listed are defined. For all parameters described as addresses, see Section A.1 on page 197 for details.

C.1 P_ACCEPT

The P_ACCEPT_UC verb is used to inform the Common MPTN Manager that a non-native connection request has arrived.

P_ACCEPT_UC (↑)	*Input Parameters*	mptn_connect init_prov_addr targ_prov_addr prov_id pmm_conn_id partner_node_init_id
	Input/Output Parameters	cmm_conn_id
	Output Parameters	none

Parameters

mptn_connect
The MPTN_Connect command that was received on the transport provider's well known local address.

init_prov_addr
The address of the remote matching transport provider.

targ_prov_addr
The address of the local transport provider.

prov_id
Identifies the local transport provider used for this connection.

pmm_conn_id
The PMM connection identifier to be used on subsequent downcalls.

partner_node_init_id
This is the PMM's record of the partner's node initialisation id. This is a value that uniquely

identifies the instance of the partner node; for example, a time stamp corresponding to when the partner node was initialised.

cmm_conn_id
The CMM's connection identifier.

The P_ACCEPT_DC verb is used to indicate if the partner's connection request (in a previous P_ACCEPT) was accepted by the local transport user.

P_ACCEPT_DC **(↓)**	*Input Parameters*	pmm_conn_id status comp_hdrs mptn_conn_rsp init_user_addr connect_corr p_create
	Output Parameters	none

Parameters

pmm_conn_id
The PMM connection identifier to be used on subsequent downcalls.

status
Indicates whether the local transport user accepted the incoming connection request. This request was in a previous P_ACCEPT_UC.

comp_hdrs
The MPTN compensation headers required for this connection.

mptn_conn_rsp
The MPTN_Connect response data that is to be returned.

init_user_addr
The address of the remote matching transport user. Used as an expedited data correlator prefix.

connect_corr
The correlator suffix from the MPTN_CONNECT. Used together with the previous parameter, init_user_addr, to form the *connection correlator* of the MPTN_DG_OOB_Data (see the **MPTN Data Formats** Specification (Section 5.3, MPTN_DG_OOB_Data)).

p_create
The transport user's connection characteristics. The CMM fills in all remaining parameters in the P_CREATE data structure. See Section C.6 on page 222 for details of the structure.

C.2 P_CLOSE

This verb terminates a previously established connection.

P_CLOSE_DC **(↓)**	*Input Parameters*	pmm_conn_id term_data close_type linger_time
	Output Parameters	none

Parameters

pmm_conn_id
: The PMM's connection identifier.

term_data
: The transport user-specific data that is sent as part of the termination processing.

close_type
: Indicates the termination type requested. Valid values are:

- SIMPLEX_ABORTIVE
- SIMPLEX_ORDERLY
- DUPLEX_ABORTIVE
- DUPLEX_ORDERLY
- CLOSE_CB (close a non-connected communication end point).

linger_time
: Time to linger before issuing close. If there is unsent data in the send queue, this is the time to attempt to send this data before the connection is closed.

P_CLOSE_UC **(↑)**	*Input Parameters*	cmm_conn_id status
	Output Parameters	none

Parameters

cmm_conn_id
: The CMM's connection identifier.

status
: The result of the P_CLOSE_DC processing.

C.3 P_CLOSED

This verb notifies the CMM of a connection termination, due to the partner transport provider closing the connection, or because of a connection failure.

P_CLOSED_UC **(↑)**	*Input Parameters*	cmm_conn_id term_data close_indicator
	Output Parameters	none

Parameters

cmm_conn_id
The CMM's connection identifier.

term_data
The transport user-specific data that is sent as part of the termination processing.

close_indicator
Indicates how the connection was closed. Valid values are:

- CI_SIMPLEX_ABORTIVE
- CI_SIMPLEX_ORDERLY
- CI_DUPLEX_ABORTIVE
- CI_DUPLEX_ORDERLY
- CI_ABNORMAL

C.4 P_CONFLICT

This verb notifies the CMM of an address conflict between two or more transport users. One of the transport users using the address was registered by this CMM.

P_CONFLICT_UC **(↑)**	*Input Parameters*	cmm_conn_id prov_id local_user_address
	Output Parameters	none

Parameters

cmm_conn_id
The CMM's connection identifier.

prov_id
Identifies the transport provider that discovered the conflict.

local_user_addr
Address that is in conflict.

C.5 P_CONNECT

The P_CONNECT verb is used to establish a connection with a matching transport user at a specified transport provider address.

P_CONNECT_DC (↓)	*Input Parameters*	pmm_conn_id init_prov_addr targ_prov_addr init_user_addr connect_corr mptn_connect comp_hdrs service_mode
	Output Parameters	none

Parameters

pmm_conn_id
The PMM's connection identifier.

init_prov_addr
The address of the local transport provider.

targ_prov_addr
The address of the remote matching transport provider.

init_user_addr
The address of the local transport user. Used as an expedited data correlator prefix.

connect_corr
The connection correlator.

mptn_connect
The MPTN_Connect request that should be forwarded to the matching PMM on the transport provider connection.

comp_hdrs
The MPTN compensation headers required for this connection.

service_mode
The service mode which is requested for the connection.

P_CONNECT_UC (↑)	*Input Parameters*	cmm_conn_id status mptn_conn_rsp init_prov_addr
	Output Parameters	tsdu comp_hdrs partner_node_init_id

Parameters

cmm_conn_id
: The CMM's connection identifier.

status
: The result of the P_CONNECT processing.

mptn_conn_rsp
: The MPTN_Connect Response returned by the matching node.

init_prov_addr
: The address of the local transport provider. Used for network management.

tsdu
: The negotiated maximum unsegmented record size supported by the transport provider.

comp_hdrs
: The MPTN compensation headers required for this connection.

partner_node_init_id
: This is the partner's node initialisation id. This is a value that uniquely identifies the instance of the partner node; for example, a time stamp corresponding to when the partner node was initialised.

C.6 P_CREATE

This verb creates a control block for use by the PMM in establishing a transport endpoint.

P_CREATE_DC **(↓)**	*Input Parameters*	cmm_conn_id partial_records tsdu etsdu in_line expedited_marking
	Output Parameters	pmm_conn_id

Parameters

cmm_conn_id
: CMM connection endpoint identifier to be used on all subsequent upcalls for this connection.

partial_records
: Indicates whether partial records are supported by the transport user.

tsdu
: Maximum unsegmented record size supported by the transport user.

etsdu
: Maximum unsegmented expedited data size supported by the transport user.

in_line
: Indicates that this Transport User's expedited data must be sent in line (that is, with the rest of the normal data). The expedited facility of the Transport Provider should not be used.

expedited_marker
: Indicates to the PMM whether or not an expedited marker is requires.

pmm_conn_id
: PMM connection endpoint identifier to be used on all subsequent downcalls for this connection.

C.7 P_DEREGISTER

This verb requests that a transport user's address be deregistered from whatever dynamic directory service is supported by the PMM.

P_DEREGISTER_DC (↓)	*Input Parameters*	pmm_conn_id local_user_addr
	Output Parameters	none

Parameters

pmm_conn_id
: PMM connection identifier.

local_user_addr
: The address of the transport user to be deregistered.

C.8 P_INIT

This verb is used by the CMM to initialise a PMM. The CMM performs a one time initialisation of the PMM. It makes a P_INIT_DC to request the PMM to initialise itself. The PMM in turn makes a P_INIT_UC to the CMM to report (successful or unsuccessful) completion of initialisation.

P_INIT_DC (↓)	*Input Parameters*	cmm_version p_init_corr
	Output Parameters	pmm_version

Parameters

cmm_version
: Common MPTN Manager version.

p_init_corr
: Correlator used on the P_INIT_UC.

pmm_version
: PMM version.

P_INIT_UC (↑)	*Input Parameters*	status p_init_corr prov_info
	Output Parameters	none

Parameters

status
Status of the initialisation.

p_init_corr
Correlator from the P_INIT_DC downcall.

prov_info
A pointer to the PROV_INFO data structure which contains the characteristics of the transport provider. Refer to Section A.4 on page 200 for details of the PROV_INFO structure.

C.9 P_JOIN_GROUP

This verb allows the transport user to add its address to a multicast group.

P_JOIN_GROUP_DC (↓)	*Input Parameters*	cmm_conn_id targ_prov_group mptn_cb
	Output Parameters	none

Parameters

cmm_conn_id
The CMM's connection identifier.

targ_prov_group
The transport provider address of the group to join.

If the transport provider's group address is non-null, this is a request to join the specified group.

If the transport provider's group address is null, this is a request to create a transport provider group and join it. The address of the group that was created/joined is returned via the P_JOIN_GROUP_UC.

P_JOIN_GROUP_UC **(↑)**	*Input Parameters*	cmm_conn_id status prov_id targ_prov_group
	Output Parameters	none

Parameters

cmm_conn_id
The CMM's connection identifier.

status
The status of the attempt to join the group.

prov_id
Identifies the transport provider used to join the group.

targ_prov_group
The transport provider address of the group joined.

C.10 P_LOCATE

This verb is used to acquire address mappings when the name service is transport provider-specific.

P_LOCATE_DC **(↓)**	*Input Parameters*	cmm_conn_id dg_locate targ_user_addr service_mode
	Output Parameters	none

Parameters

cmm_conn_id
The CMM identifier, returned on upcall.

dg_locate
True if this is for a connectionless request.

targ_user_addr
The address of the remote transport user for which a mapping is being sought.

service_mode
The service mode associated with this request.

P_LOCATE_UC **(↑)**	*Input Parameters*	cmm_conn_id status targ_user_addr targ_prov_addrs prov_id
	Output Parameters	none

Parameters

cmm_conn_id
: The CMM identifier, returned from downcall.

status
: The status of the attempted locate.

targ_user_addr
: The address of the matching transport user.

targ_prov_addrs
: List of one or more addresses of matching transport providers found. Ignored if the *status* parameter indicates failure.

prov_id
: Identifies the transport provider responding to the locate.

C.11 P_QUIT_GROUP

This verb allows the transport user to delete its address from a multicast group address.

P_QUIT_GROUP_DC (↓)	*Input Parameters*	targ_prov_group
	Output Parameters	none

Parameters

targ_prov_group
: The address of the transport provider group to quit.

C.12 P_RCV_DG

This verb indicates that a datagram from a remote transport user has arrived.

P_RCV_DG_UC (↑)	*Input Parameters*	data prov_id init_prov_addr targ_prov_addr service_mode broadcast
	Output Parameters	status

Parameters

data
: The MPTN datagram received.

prov_id
: Identifies the transport provider used for this connection.

init_prov_addr
: The address of the local transport provider.

targ_prov_addr
  The address of the remote matching transport provider.

service_mode
  The service mode used.

broadcast
  Indicates whether or not the datagram was a broadcast.

status
  The status of the received datagram.

C.13 P_RECEIVE

This verb indicates to the CMM that data has arrived on a connection.

P_RECEIVE_UC **(↑)**	*Input Parameters*	cmm_conn_id data expedited_data incomplete_record expedited_marker status
	Output Parameters	none

Parameters

cmm_conn_id
  The CMM's connection identifier.

data
  The data received.

expedited_data
  Indicates whether or not the data received is marked as expedited. Expedited data will only be passed on record boundaries if the datatype is record.

incomplete_record
  When record data format is being received, this parameter indicates whether or not the data being passed to the transport user completes a record. Used if the transport user supports partial records.

expedited_marker
  A marker used to correlate the position of expedited data in the normal field in which it was sent.

status
  The status of the receive.

C.14 P_REGISTER

This verb requests that a transport user's address be registered to whatever dynamic directory service is supported by the PMM.

P_REGISTER_DC **(↓)**	*Input Parameters*	cmm_conn_id local_user_addr
	Output Parameters	local_prov_group

Parameters

cmm_conn_id
The CMM identifier, returned on corresponding upcall.

local_user_addr
The address of the transport user to be registered.

local_prov_group
If the address in the *local_user_addr* parameter is a group address and algorithmic mapping is used, the corresponding transport provider group name is returned.

P_REGISTER_UC **(↑)**	*Input Parameters*	cmm_conn_id status prov_id local_user_addr local_prov_group
	Output Parameters	none

Parameters

cmm_conn_id
The CMM identifier, returned from downcall.

status
The status of the registration attempt.

prov_id
Identifies the transport provider used.

local_user_addr
The address of the transport user to be registered.

local_prov_group
If the address in the *local_user_addr* parameter is a group address and protocol specific name server is used, the corresponding transport provider group name is returned.

C.15 P_SEND

This verb is issued by the CMM to send data on a connection.

P_SEND_DC (↓)	*Input Parameters*	pmm_conn_id data expedited_data incomplete_record flush
	Output Parameters	none

Parameters

pmm_conn_id
: The PMM's connection identifier.

data
: The data to be sent.

expedited_data
: Indicates whether or not the data is expedited data. Expedited data will be passed only on record boundaries if the datatype is record. The transport user may not send expedited data if the previous M_SEND_DC issued by the transport user sent a partial record.

incomplete_record
: If the value is TRUE, it indicates that the data being passed is an incomplete record and more will follow. If the value is FALSE, it indicates that the data being passed is either a complete record or completes a previous partial record.

: This parameter is always FALSE if record segmentation is not supported on this connection, or the data type is stream.

flush
: A value of TRUE in this parameter indicates a request to the CMM to send this data and any data that is queued to the PMM as quickly as possible. A positive return code does not imply that this has been done, only that the PMM has committed to do it.

C.16 P_SEND_DG

P_SEND_DG_DC (↓)	*Input Parameters*	retry init_prov_addr targ_prov_addr service_mode mptn_hdr data
	Output Parameters	none

Input parameters

retry
: Indicates that the PMM should discard any cached address information.

init_prov_addr
: The address of the local transport provider.

targ_prov_addr
: The address of the remote matching transport provider.

service_mode
: The service mode requested for sending the datagram.

mptn_hdr
: The MPTN_Datagram command. Does not include user data.

data
: The user data to be sent.

Appendix D

Protocol-specific Transport Users

The characteristics of a transport user are defined during initialisation. Table D-1 illustrates the differences in transport characteristics for several protocols. Refer to Section A.3 on page 198 and Section A.2 on page 197 for further information on the data structures and parameters used to describe transport characteristics.

	Transport User				
Support	**SNA (LU)**	**NetBIOS**	**AF_INET Sockets (sock_stream)**	**AF_INET Sockets (sock_dgram)**	**OSI**
Service type (connection or datagram)	CO	Both	CO	CL	Both
Connection data	Yes	No	No	Not applicable	Yes
Connection reply data	Yes	No	No	Not applicable	Yes
Connection termination data	Yes	No	No	Not applicable	Yes
Data type	Record	Record	Stream	Not applicable	Record
Maximum expedited record length	95	Not applicable	1	Not applicable	16 if selected
Expedited marking	No	Not applicable	Yes	Not applicable	No
Termination types	Duplex Abortive	Duplex Abortive Duplex Orderly	Simplex Orderly Duplex Abortive Duplex Orderly	Not applicable	Duplex Abortive
Partial records	No	Yes	Not applicable	No	No
Maximum Normal record length	Configuration and Implementation dependent	128k-2 bytes	Not applicable	Not applicable	Implementation dependent
Maximum datagram length	Not applicable	Configured value	Not applicable	64k-9 bytes	Implementation dependent
Session outage notification	Yes	No	No	Not applicable	

Table D-1 Transport Characteristics for Various Transport Users

Appendix E

Processing Specification Values

The following table lists the set of valid and invalid values for the processing specification field in the common prefix for MPTN commands (optional fields do not have any invalid values). See the **MPTN Data Formats** Specification (Chapter 8, Formats Common to Multiple Messages) for details on the meaning of each bit.

Table E-1 Values of the Processing Specification Field

(Bit 0 is the most significant bit.)

0	1	2	3	4	5	6	7	Legal
0	0	0	R	0	0	0	0	Y
0	0	0	R	0	0	0	1	N
0	0	0	R	0	0	1	0	Y
0	0	0	R	0	0	1	1	N
0	0	0	R	0	1	0	0	Y
0	0	0	R	0	1	0	1	N
0	0	0	R	0	1	1	0	Y
0	0	0	R	0	1	1	1	N
0	0	0	R	1	0	0	0	Y
0	0	0	R	1	0	0	1	N
0	0	0	R	1	0	1	0	Y
0	0	0	R	1	0	1	1	N
0	0	0	R	1	1	0	0	N
0	0	0	R	1	1	0	1	N
0	0	0	R	1	1	1	0	N
0	0	0	R	1	1	1	1	N
0	0	1	R	X	X	X	X	N
0	1	X	R	X	X	X	X	N
1	0	0	R	0	0	0	0	Y
1	0	0	R	0	0	0	1	N
1	0	0	R	0	0	1	0	Y
1	0	0	R	0	0	1	1	N
1	0	0	R	0	1	0	0	Y
1	0	0	R	0	1	0	1	N
1	0	0	R	0	1	1	0	Y
1	0	0	R	0	1	1	1	N
1	0	0	R	1	0	0	0	Y
1	0	0	R	1	0	0	1	N
1	0	0	R	1	0	1	0	Y

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Legal**
1	0	0	R	1	0	1	1	N
1	0	0	R	1	1	0	0	N
1	0	0	R	1	1	0	1	N
1	0	0	R	1	1	1	0	N
1	0	0	R	1	1	1	1	N
1	0	1	R	X	X	X	X	N
1	1	0	R	0	0	0	0	Y
1	1	0	R	0	0	0	1	N
1	1	0	R	0	0	1	0	Y
1	1	0	R	0	0	1	1	Y
1	1	0	R	0	1	0	0	Y
1	1	0	R	0	1	0	1	N
1	1	0	R	0	1	1	0	Y
1	1	0	R	0	1	1	1	Y
1	1	0	R	1	0	0	0	Y
1	1	0	R	1	0	0	1	N
1	1	0	R	1	0	1	0	N
1	1	0	R	1	0	1	1	Y
1	1	0	R	1	1	0	0	Y
1	1	0	R	1	1	0	1	N
1	1	0	R	1	1	1	0	Y
1	1	0	R	1	1	1	1	Y
1	1	1	R	0	0	0	0	Y
1	1	1	R	0	0	0	1	N
1	1	1	R	0	0	1	0	Y
1	1	1	R	0	0	1	1	N
1	1	1	R	0	1	0	0	Y
1	1	1	R	0	1	0	1	N
1	1	1	R	0	1	1	0	N
1	1	1	R	0	1	1	1	N
1	1	1	R	1	0	0	0	Y
1	1	1	R	1	0	0	1	N
1	1	1	R	1	0	1	0	Y
1	1	1	R	1	0	1	1	N
1	1	1	R	1	1	0	0	N
1	1	1	R	1	1	0	1	N
1	1	1	R	1	1	1	0	N
1	1	1	R	1	1	1	1	N

Appendix F

Protocol Flow Notation

The following diagram shows the symbols used in the flow diagrams presented in this specification.

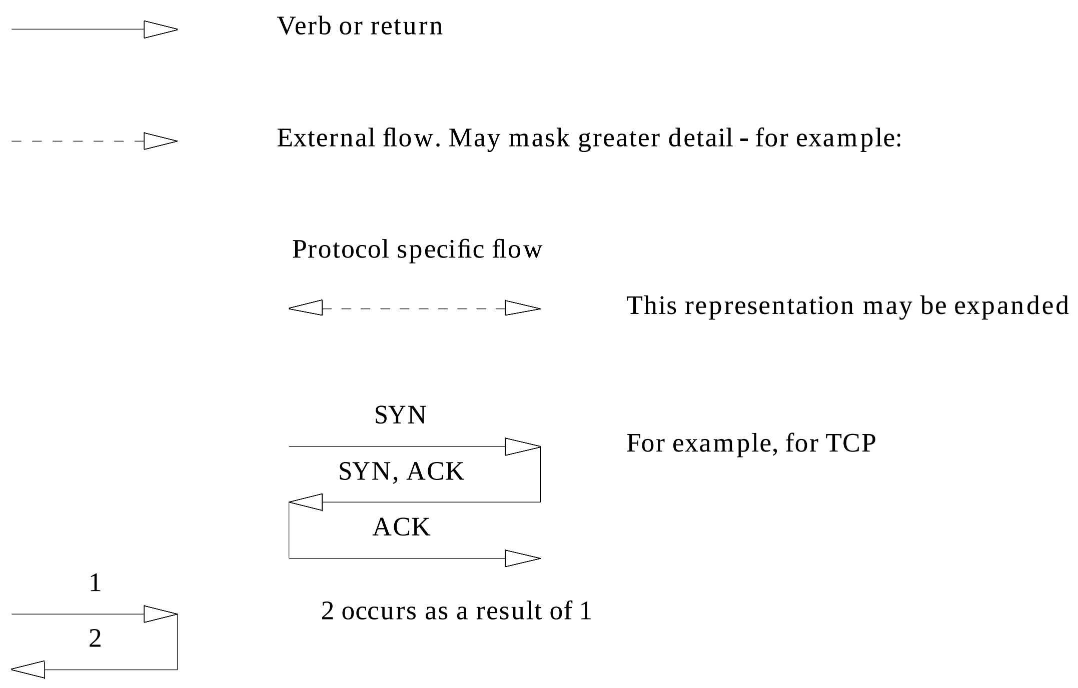

Verb or return
External flow. May mask greater detail - for example:
Protocol specific flow
This representation may be expanded
SYN
SYN, ACK
ACK
For example, for TCP
1
2
2 occurs as a result of 1

Figure F-1 Notation Used in Flow Diagrams

Glossary

The following MPTN definitions of terms and abbreviations are common to the *Multiprotocol Transport Networking (MPTN)* environment. Not all of them are used in this document.

address-mapper function
An MPTN component that maps non-native transport-user addresses to a form used in the native transport network.

address space
The set of all legal transport addresses that may be formed according to the rules of a given address type. These rules include the maximum number of characters that can be in the address and the permissible characters. Each protocol has it own set of rules. Since addresses in one protocol may also be legitimate in another protocol, MTPN qualifies all transport addresses with an address type.

address type
An identifier in an MPTN header that indicates the protocol category (for example, OSI or TCP/IP) and hence the specific syntax and structure of the accompanying address. A given transport-user address plus its address type form an MPTN-qualified address.

API
Application programming interface.

application programming interface
An interface between the application program and the application support layer.

ASCII
American National Standard Code for Information Interchange.

below-specific protocol boundary (BSPB)
The interface between the common MPTN manager (CMM) and the protocol-specific MPTN manager (PMM).

below-specific
Specific to one transport provider that exists below the CMM.

BSD
Berkeley software distribution.

BSPB
Below-specific protocol boundary.

CMIP
Common management information protocol.

CMM
Common MPTN manager.

common MPTN manager (CMM)
The component of the MPTN architecture that provides services independent of any protocol. Examples include registering transport users with the MPTN address mapper component, selecting a transport provider, and establishing MPTN connections.

compensation
The function of making up for differences in functions requested by the transport user and those provided by the transport provider.

connectionless service
A service that treats each packet or datagram as a separate entity that contains the source address and destination address. Connectionless services are on a best-effort basis and do not guarantee reliable or in-sequence delivery.

connection-oriented service
A service that establishes a logical connection between two partners for the duration that they want to communicate. Data transfer takes place in a reliable, sequenced manner.

CPI-C
Common Programming Interface for Communications.

datagram
A self-contained packet, independent of other packets, that carries information sufficient for routing from the source transport user to the destination transport user.

datagram segment
A part of a datagram. A datagram may be segmented (that is, split into more than one part) if it contains too many bytes of data to send at one time.

EBCDIC
Extended binary-coded decimal interchange code.

expedited data
Data that is considered urgent. Such data may be delivered ahead of normal data that preceded it.

group address
A single transport address identifying a collection of users. The collection of users is formed so that they can all receive common multicast datagrams.

IP
The networking protocol that forms part of the Internet Protocol suite referred to as TCP/IP. The internet protocol defines the internet datagram as the unit of information passed across the internet, and provides the basis for the internet connectionless, best-effort packet-delivery service.

LU
Logical unit.

LU 6.2
An SNA logical unit that supports general communication between programs in a

distributed processing environment.

matching
The relationship between peer transport users or peer transport providers that are of the same family.

MPTN
Multiprotocol Transport Networking.

MPTN access node
A node that has MPTN components installed, allowing transport users to use non-native transport providers.

MPTN-qualified transport address
A transport address that is qualified by its corresponding address type. The address conforms to the syntax and meaning of the specified address type. An example of an MPTN-qualified transport address is the pair: (SNA, Net ID.LUname).

multicast
A technique that allows a single packet (or datagram) to be passed to a selected group of destinations that share a group address.

multicast datagram
A packet that is sent to more than one partner.

multiprotocol node
An implementation that supports more than one transport protocol.

multiprotocol transport networking (MPTN)
The architecture for mixed-protocol networking.

native
A relationship between a transport user and a transport provider that are based on the same transport protocol.

native network
With respect to a particular transport user, a transport network that provides the address type and transport characteristics assumed in the design of the transport user. No MPTN address mapping or compensation protocols are used for data transfer.

native node
A node with no MPTN capability.

native transport address
A transport-user address having the address type that corresponds to the type employed by the transport network underlying the transport user, for example, an SNA name that is being registered within an SNA network.

NetBEUI
NetBIOS Extended User Interface.

NetBIOS
Network Basic Input/Output System.

NetBIOS extended user interface
NetBEUI: the API to the NetBIOS transport protocol.

net ID
Network Identifier. The address qualifier of a transport address that identifies a group of nodes according to the network in which they reside.

In an MPTN environment, the transport user and transport provider have separate NetId domains.

Network Basic Input/Output System
NetBIOS: a protocol used by many small computers for network access.

networking
Providing a relaying and routing service.

networking protocol
A specification of the rules governing the exchange of information between components of a network.

non-native
A relationship between a transport user and transport provider that are based on different transport protocols.

non-native network
With respect to a particular transport user, a transport network whose addressing structure and transport service are different from that assumed in the design of that transport user.

non-native protocols
Protocols used by a non-native network.

non-native transport address
A transport-user address having an address type different from the transport network address type, for example, an OSI address for the target of a connection request carried on an SNA transport network.

OSI
Open Systems Interconnection. The interconnection of open systems in accordance with the hierarchical arrangement of the seven layers of networking functionality described in specific International Standards Organization standards.

PMM
Protocol-specific MPTN manager.

protocol boundary
A generic description of a functional boundary defined by the architecture; implementations must conform to the semantics of the protocol boundary, but not necessarily the syntax.

protocol-specific MPTN manager (PMM)
A component of the MPTN architecture that performs management, routing, and binding functions that are performed differently for the different transport providers.

record data format
A format that maintains record boundaries for the data being transmitted.

RFC
Request for Comment. The process by which some standards bodies define specialised solutions. In the case of MPTN, it is the definition of specialised transport protocols.

service mode
The designation by a transport user of the characteristics that must be maintained for a given connection or datagram transmission. Each networking protocol has its own way of requesting these characteristics, which must be mapped to the MPTN service mode.

single-protocol transport network
A collection of physically connected nodes that implement a single common transport protocol. A single-protocol transport network may span multiple net IDs.

SNA
Systems Network Architecture.

SNMP
Simple network management protocol.

socket
The abstraction provided by Berkeley 4.3 BSD UNIX that allows an application program to access TCP/IP protocols.

SON
Session Outage Notification - this is a transport user characteristic.

SPTN
Single-protocol transport network.

stream data format
Data that has no record boundaries. The data is simply a stream of bits.

TCP
Transmission Control Protocol.

Transmission Control Protocol (TCP)
The Internet standard transport level protocol that provides the reliable, full-duplex, stream service for TCP applications.

TCP/IP
Abbreviation for the protocols (that is, TCP, IP, UDP) that comprise the Defense Advanced Research Projects Agency (DARPA) Internet protocol standards.

TLPB
Transport-layer protocol boundary

transport characteristics
The set of transport services that a transport user expects; for example, whether data will be sent using connections or datagrams, and formatted as streams or

records.

transport-layer protocol boundary (TLPB)
The MPTN protocol boundary that provides access in a protocol-independent fashion to multiple transport protocols.

transport network
An implementation of transport networking. Examples are parts of SNA, TCP/IP, OSI, IPX, NetBIOS, DECnet and Appletalk.

transport networking
The communications services provided at the transport layer and below.

transport networking protocol
A specification of the rules governing the exchange of information between components of a transport network.

transport provider
A component that provides the transport functions associated with a particular protocol stack.

transport-provider address
A transport address used to identify a transport provider.

transport user
An application program or application support element that uses transport services to convey data through a network. A program that directly requests transport services.

transport-user address
A transport address used to identify a transport user.

UDP
User Datagram Protocol.

unicast datagram
A packet that is sent to a single partner.

UNIX
An operating system developed by Bell Laboratories.

User Datagram Protocol (UDP)
The TCP/IP protocol that allows an application program in one node to send a datagram to an application program in another node. UDP uses the internet protocol (IP) to deliver datagrams.

XMPTN
X/Open specification of Multiprotocol Transport Networking (MPTN).

Index