Appendix D

Data Link Service Access Point and Primitives

D.1. Model of a Data-Link Connection

A Data Link Service Access Point (DLSAP) is the point at which the data-link layer provides services to Layer 3. It provides a uniform programming interface to access the data-link services. This Appendix specifies this interface but does not specify the upper layer entities. In a basic user TNC, the upper layer entity may consist only of a Layer 7 user application with Layers 3-6 being null.

The following primitives are used to pass commands and receive responses from the DLSAP:

- **(Called Callsign)**. This primitive is used by the Layer 3 entity to request the establishment of an AX.25 connection.

- **DL-CONNECT Indication (Calling Callsign)**. This primitive is used by the Data-link State Machine to indicate an AX.25 connection has been requested.

- **DL-CONNECT Confirm (VOID)**. This primitive is used by the Data-link State Machine to indicate an AX.25 connection has been made.

- **DL-DISCONNECT Request**. This primitive is used by the Layer 3 entity to request the release of an AX.25 connection.

- **DL-DISCONNECT Indication**. This primitive is used by the Data-link State Machine to indicate an AX.25 connection has been released.

- **DL-DISCONNECT Confirm**. This primitive is used by the Data-link State Machine to indicate an AX.25 connection has been released and confirmed.

- **DL-DATA Request**. This primitive is used by the Layer 3 entity to request the transmission of data using connection-oriented protocol. This frame is examined and acted upon by the segmenter, if necessary.

- **DL-DATA Indication**. This primitive is used by the reassembler to indicate reception of Layer 3 data using connection-oriented protocol.

- **DL-UNIT-DATA Request**. This primitive is used by the Layer 3 entity to request the transmission of data using connectionless protocol. This frame is examined and acted upon by the segmenter, if necessary.

- **DL-UNIT-DATA Indication**. This primitive is used by the reassembler to indicate reception of Layer 3 data using connectionless protocol.

• **DL-ERROR Indication**. This primitive is used by the Data-link State Machine to indicate when frames have been received that are inconsistent with this protocol definition. This includes short frames, frames with inconsistent parameter values, etc. The error indications are discussed in the SDL appendices.

• **DL-FLOW-OFF Request**. This primitive is used by the Layer 3 entity to temporarily suspend the flow of incoming information.

• **DL-FLOW-ON Request**. This primitive is used by the Layer 3 entity to resume the flow of incoming information.

• **MDL-NEGOTIATE Request**. This primitive is used by the Layer 3 entity to request the Data-link State Machine to notify/negotiate.

• **MDL-NEGOTIATE Confirm**. This primitive is used by the Management Data-link State Machine to notify the Layer 3 entity that notification/negotiation is complete.

• **MDL-ERROR Indicate**. This primitive is used by the Management Data-link State Machine to notify the Layer 3 entity that notification/negotiation has failed.

D.2. Queue Model Concepts

The queue model represents the operation of the Data-Link Connection (DLC) in the abstract by a pair of queues linking the two DLSAPs. There is one queue for each direction of information flow (see Figure D.1).

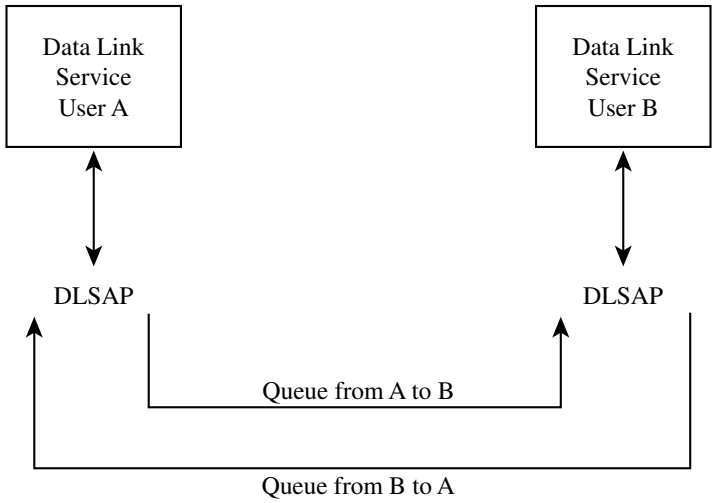

Figure D.1. Queue model of a data-link connection.

The pair of queues is considered to be available for each Data-Link Service (DLS) user. Each queue represents a DLS flow control function in one direction of transfer. The ability of a user to add objects to a queue will be determined by the behavior of the other DLS user in removing objects from the queue and by the state of the queue. Objects are entered or removed from the queue as a result of interactions at the two DLSAPs.

The following objects may be placed in a queue by a DLS user:

- A connect object, representing a DL-CONNECT primitive and its parameters.
- A data object, representing a DL-DATA primitive and its parameters.
- A disconnect object, representing a DL-DISCONNECT primitive and its parameters.

The following object may be placed in a queue by the DLS provider:

- A disconnect object, representing a DL-DISCONNECT primitive and its parameters.

The queues are defined to have the following general properties:

- A queue is empty before a connect object has been entered and can be returned to this state, with loss of its contents, by the DLS provider.
- Objects are entered into a queue by the sending DLS user, subject to control by the DLS provider. Objects may also be entered by the DLS provider.
- Objects are removed from the queue, under the control of the receiving DLS user.
- Objects are normally removed in the same order that they were entered.
- A queue has a limited capacity, but this capacity is not necessarily either fixed or determinable.

D.3. DLC Establishment

A pair of queues is associated with a DLC between two DLSAPs when the DLS provider receives a DL-CONNECT request primitive at one of the DLSAPs, and a connect object is entered into one of the queues. From the standpoint of the DLS users of the DLC, the queues remain associated with the DLC until a disconnect object representing a DL-DISCONNECT primitive is either entered or removed from the queue.

DLS user A, who initiates a DLC establishment by entering a connect object representing a DL-CONNECT request primitive into the queue from DLS user A to DLS user B, is not allowed to enter any other object, other than a disconnect object, into the queue until after the connect object representing the DL-CONNECT confirm primitive has been removed from the DLS user B to DLS user A queue. In the queue from DLS user B to DLS user A, objects can be entered only after DLS user B has entered a connect object representing a DL-CONNECT response primitive.

The properties exhibited by the queues while the DLC exists represent the agreements reached among the DLS users and the DLS provider during this connection establishment procedure.

D.4. Data Transfer

Flow control on the DLC is represented in this queue model by the management of the queue capacity, allowing objects to be added to the queues. The addition of an object may prevent the addition of a further object.

Once objects are in the queue, the DLS provider may manipulate pairs of adjacent objects, resulting in deletion. An object may be deleted if, and only if, the object that follows it is defined to be destructive with respect to the object. If necessary, the last object on the queue will be deleted to allow a destructive object to be entered; thus, a destructive object may always be added to the queue. Disconnect objects are defined to be destructive with respect to all objects.

The relationship between objects that may be manipulated in the above fashion are summarized in Figure D.2.

Preceding Object \ Following Object	Connect	Data	Sync	Disconnect
Connect	N/A	—	N/A	DES
Data	N/A	—	N/A	DES
Sync	N/A	—	N/A	DES
Disconnect	N/A	N/A	N/A	DES

Where:
N/A Not a valid state of the queue.
— Not to be destructive nor to be able to advance ahead.
DES To be destructive to the preceding object.

Figure D.2. Relationships between queue model objects.

Whether the DLS provider performs actions resulting in deletion or not will depend upon the behavior of the DLC users. In general, if a DLS user does not remove objects from a queue, the DLS provider shall, after some unspecified period of time, perform all the permitted deletions.

D.5. DLC Release

The insertion into a queue of a disconnect object, which may occur at any time, represents the initiation of the DLC release procedure. The release procedure may be destructive with respect to other objects in the two queues and eventually results in the emptying of the queues and the dissociation of the queues with the DLC.

The insertion of a disconnect object may also represent the rejection of a DLC establishment attempt or the failure to complete DLC establishment. In such cases, if a connect object representing a DL-CONNECT request primitive is deleted by a disconnect object, then the disconnect object is also deleted. The disconnect object representing the DL-CONNECT response.

D.6. Relationship of Primitives at the Two DLC Endpoints

A primitive issued at one DLC endpoint will, in general, have consequences at the other DLC endpoint. The relationship of primitives of each type at one DLC endpoint to primitives at the other DLC endpoint are defined in the appropriate subclauses discussed in Section 5.

A simple connection oriented transmission of data would be handled by the following primitives at the DLSAPs as shown in Figure D.3. Notice that the MDL primitives do not generate an Indicate primitive nor require a Response primitive from the Layer 3 entity in the station B. The MDL entities in the stations A and B work on a peer-to-peer relationship. The other primitives work in groups of four with the Request from the station A causing an Indicate in the station B, and a Response in station B causing a Confirm in the station A.

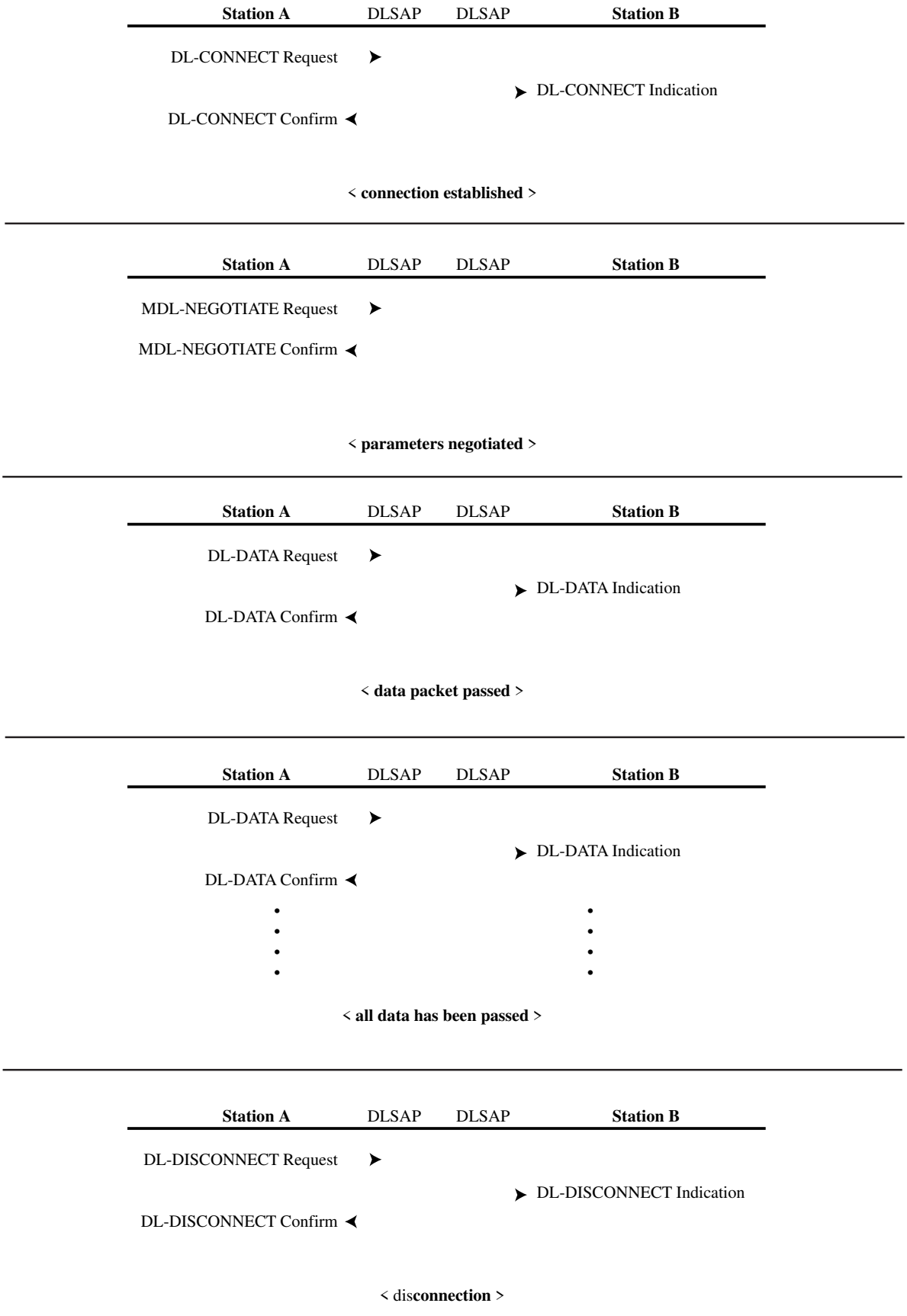

Figure D.3. Example of a connection-oriented data exchange.