Network Working Group Alan K. Bartky
Request for Comments: XXXX Sync Research Inc.
Obsoletes: RFC 1434 Editor

DecemberNovember 1994

Data Link Switching: Switch-to-Switch Protocol
EDITOR'S IN PROGRESS VERSION REVISION 0.165
12/22/9410/31/94

THIS IS AN IN PROGRESS DOCUMENT AND DOES NOT NECESSARILY REPRESENT CONSENSUS AGREEMENT AND IS FOR DISCUSSION AND REVIEW PURPOSES ONLY!!

THIS VERSION IS FOR PROPOSED CP STATUS. IF APPROVED, THEN ALL REVISION BARS, AND EDITORIAL COMMENTS WILL BE REMOVED. VERSION NUMBER IF APPROVED WILL BE VERSION 1.0.

REVISION BARS (IN THE POSTSCRIPT VERSION ONLY) SHOW THE DIFFERENCE (WHEREVER POSSIBLE) BETWEEN VERSION 0.15 AND VERSION 0.16. REVISION HISTORY HAS NOW ALSO BEEN REMOVED.

HAPPY READING

- ALAN (your editor)

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard. Distribution of this memo is unlimited.

Abstract

This RFC describes use of Data Link Switching over TCP/IP. The RFC is being distributed to members of the Internet community in order to solicit their reactions to the proposals contained in it. While the issues discussed may not be directly relevant to the research problems of the Internet, they may be interesting to a number of researchers and Implementers.

This RFC was created as a joint effort of the Advanced Peer-to-Peer Networking (APPN) Implementers Workshop (AIW) Data Link Switching (DLSw) Related Interest Group (RIG). The APPN Implementers Workshop is a group sponsored by IBM and consists of representatives of member companies implementing current and future IBM Networking interoperable products. The DLSw Related Interest Group was formed in this forum in order to produce a single version of the Switch to Switch Protocol (SSP) which could be implemented by all vendors, which would fix documentation problems with the existing RFC 1434, and which would enhance and evolve the protocol to add new functions and features.

This document is based on RFC 1434. This document contains significant changes to RFC 1434 and therefore obsoletes that document.

Any questions or comments relative to the contents of this RFC should be sent to the following Internet address:
aiw-dlsw@networking.raleigh.ibm.com.

NOTE 1: This is a widely subscribed mailing list and messages sent to this address will be sent to all members of the DLSw mailing list. For specific questions relating to subscribing to the AIW and any of it's working groups send email to:
appn@vnet.ibm.com

Information regarding all of the AIW working groups and the work they are producing can be obtained by copying, via anonymous ftp, the file aiwinfo from the Internet host networking.raleigh.ibm.com, located in directory aiw.

NOTE 2: These mailing lists and addresses are subject to change.

Table of Contents

Table of Contents 3
1. Introduction 5
1.1. Backwards Compatibility with RFC 1434 5
2. Overview 5
3. Transport Connection 8
3.1. SSP Frame Formats 8
3.2. Address Parameters 13
3.3. Correlators 15
3.4. Largest Frame Size Field 15
3.5. Message Types 17
4. Circuit Priority 19
4.1. Frame format: 19
4.2. Circuit Startup: 19
5. DLSw State Machine 21
5.1. Data Link Switch States 21
5.2. State Transition Tables 23
5.2.1. DISCONNECTED State 26
5.2.2. RESOLVE_PENDING State 27
5.2.3. CIRCUIT_START State 28
5.2.4. CIRCUIT_PENDING State 29
5.2.5. CONNECT_PENDING State 30
5.2.6. CIRCUIT_ESTABLISHED State 31
5.2.7. CONTACT_PENDING State 32
5.2.8. CONNECTED State 33
5.2.9. CIRCUIT_RESTART State 34
5.2.10. DISCONNECT_PENDING State 35
5.2.11. RESTART_PENDING State 35
5.2.12. HALT_PENDING State 36
5.2.13. HALT_PENDING_NOACK State 36
5.3. NetBIOS Datagrams 37
5.4. Explorer Traffic 38
5.4.1. CANUREACH/ICANREACH Explorer FSM 39
5.4.1.1. RESET State 39
5.4.1.2. SENT_EX State 40
5.4.1.3. RECEIVED_EX State 40
5.4.2. NETBIOS_NQ/NR Explorer FSM 41
5.4.2.1. RESET State 41
5.4.2.2. SENT_EX State 42
5.4.2.3. RECEIVED_EX State 42
5.4.2.4. SENT_REC_EX State 43
5.4.2.5. NetBIOS Session Numbers 43
6. Protocol Flow Diagrams 44
6.1. Connect Protocols 44
6.2. Link Restart Protocols 54
6.3. Disconnect Protocols 55
7. Capabilities Exchange Formats/Protocol 57
7.1. Control Vector Id Range 57
7.2. Control Vector Order and Continuity 58
7.3. Run-Time Capabilities Exchange 59
7.4. Capabilities Exchange Filtering Responsibilities 59
7.5. DLSw Capabilities Exchange Structured Subfields 60

7.5.1. Vendor Id (0x81) Control Vector 60
7.5.2. DLSw Version (0x82) Control Vector 61
7.5.3. Initial Pacing Window (0x83) Control Vector 61
7.5.4. Version String (0x84) Control Vector 61
7.5.5. MAC Address Exclusivity (0x85) Control Vector 62
7.5.6. SAP List Support (0x86) Control Vector 63
7.5.7. TCP Connections (0x87) Control Vector 65
7.5.8. NetBIOS Name Exclusivity (0x88) Control Vector 65
7.5.9. MAC Address List (0x89) Control Vector 66
7.5.10. NetBIOS Name List (0x8A) Control Vector 67
7.5.11. Vendor Context (0x8B) Control Vector 67
7.6. Capabilities Exchange Responses 68
8. Pacing/Flow Control 70
8.1. Basic Overview 70
8.2. Frame Format 71
8.3. Granting Permission to Send Data 72
8.3.1. Repeat Window Operator 72
8.3.2. Increment Window Operator 73
8.3.3. Decrement Window Operator 73
8.3.4. Reset Window Operator 73
8.3.5. Halve Window Operator 73
8.4. Acknowledging a Flow Control Operator 73
8.4.1. Acknowledging a Reset Window Operator 74
8.5. Capabilities Exchange Initial Window Size 74
8.6. Circuit Startup 74
8.7. Example Receiving Implementations 74
8.7.1. Fixed Pacing Example 75
8.7.1.1. Circuit Startup 75
8.7.1.2. Check Buffers Available 75
8.7.1.3. Buffers Become Available 75
8.7.1.4. Repeat Window Operator 75
8.7.1.5. Send FCIND(operator) 75
8.7.1.6. A Frame Arrives from Sender 76
8.7.2. Adaptive Pacing Example 76
8.7.2.1. Circuit Startup 76
8.7.2.2. Check Buffers Available (X) 76
8.7.2.3. Buffers Become Available 76
8.7.2.4. Repeat Window Operator 77
8.7.2.5. Increment Window Operator 77
8.7.2.6. Send FCIND(operator) 77
8.7.2.7. An FCACK Arrives from Sender 77
8.7.2.8. A Non-FCACK Frame Arrives from Sender 77
8.8 Adaptive Pacing Example Flow Diagrams 77
8.8.1 Example Flows from the Above Implementation 77
8.8.2. Example Halve Window Flow 79
8.8.3. Example Reset Window Flows 79
8.9. Other Considerations 81
8.9.1. Protocol Violations 81
Acknowledgments 82
References 82
Security Considerations 82
Editor's Address 83

1. Introduction

Data Link Switching (DLSw) is a forwarding mechanism for the IBM SNA (Systems Network Architecture) and IBM NetBIOS (Network Basic Input Output Services) protocols. This memo documents the Switch-to-Switch Protocol (SSP) that is used between Data Link Switches. This protocol does not provide full routing, but instead provides switching at the SNA Data Link layer (i.e. layer 2 in the SNA ~~and Open Systems Interconnection (OSI)~~ architectures~~)~~ and encapsulation in TCP/IP for transport over the Internet. This RFC documents the frame formats and protocols for multiplexing data between Data Link Switches. The initial implementation of SSP uses TCP as the reliable transport between Data Link Switches. However, other transport connections such as OSI TP4 could be used in the future.

A Data Link Switch (abbreviated also as DLSw in this document) can support SNA (Physical Unit (PU) 2, PU 2.1 and PU 4) systems and optionally NetBIOS systems attached to IEEE 802.2 compliant Local Area Networks, as well as SNA (PU 2 (primary or secondary) and PU2.1) systems attached to IBM Synchronous Data Link Control (SDLC) links. For the latter case, the SDLC attached systems are provided with a LAN appearance within the Data Link Switch (each SDLC PU is presented to the SSP protocol as a unique MAC/SAP address pair). For the Token-Ring LAN attached systems, the Data Link Switch appears as a source-routing bridge. Token-Ring Remote systems that are accessed through the Data Link Switch appear as systems attached to an adjacent ring. This ring is a virtual ring that is manifested within each Data Link Switch.

1.1. Backwards Compatibility with RFC 1434

This document defines significant changes to RFC 1434 and does not state details on how to interoperate with RFC-1434 or "enhanced" implementations (e.g. those that added enter and exit busy flow control). It is up to the implementer to refer to RFC 1434 and/or any other vendor's documentation in order to interoperate with a given vendor's implementation, if interoperability with pre-AIW DLSw RIG standards is desired.

2. Overview

Data Link Switching was developed to provide support for SNA and NetBIOS in multi-protocol routers. Since SNA and NetBIOS are basically connection oriented protocols, the Data Link Control procedure that they use on the LAN is IEEE 802.2 Logical Link Control (LLC) Type 2. Data Link Switching also accommodates SNA protocols over WAN (Wide Area Network) links via the SDLC protocol.

IEEE 802.2 LLC Type 2 was designed with the assumption that the network transit delay would be predictable (i.e., a local LAN). Therefore the LLC Type 2 elements of procedure use a fixed timer for detecting lost frames. When bridging is used over wide area lines (especially at lower speeds), the network delay is larger and it can vary greatly based upon congestion. When the delay exceeds the time-out value LLC Type 2 attempts to retransmit. If the frame is not actually lost, only delayed, it is possible for the LLC Type 2 procedures to become confused. And as a result, the link may be eventually taken down if the delay exceeds the T1 timer times N2 retry count.

Given the use of LLC Type 2 services, Data Link Switching addresses the following bridging problems:

DLC Time-outs
DLC Acknowledgments over the WAN
Flow and Congestion Control
Broadcast Control of Search Packets
Source-Route Bridging Hop Count Limits

NetBIOS also makes extensive use of datagram services that use connectionless LLC Type 1 service. In this case, Data Link Switching addresses the last two problems in the above list.

The principal difference between Data Link Switching and bridging is that for connection -oriented data DLSw terminates the Data Link Control whereas bridging does not. The following figure illustrates this difference based upon two end systems operating with LLC Type 2 services.

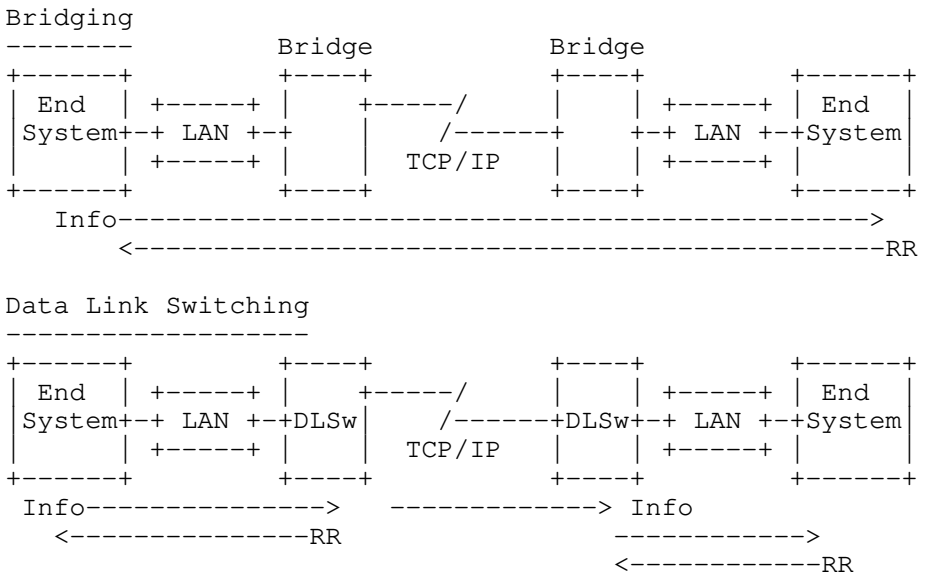


Figure 1. Data Link Switching Contrasted to Bridging

In traditional bridging, the Data Link Control is end-to-end. Data Link Switching terminates the LLC Type 2 connection at the switch. This means that the LLC Type 2 connections do not cross the wide area network. The DLSw multiplexes LLC connections onto a TCP connection to another DLSw. Therefore, the LLC connections at each end are totally independent of each other. It is the responsibility of the Data Link Switch to deliver frames that it has received from a LLC connection to the other end. TCP is used between the Data Link Switches to guarantee delivery of frames.

As a result of this design, LLC time-outs are limited to the local LAN (i.e., they do not traverse the wide area). Also, the LLC Type 2 acknowledgments (RR's) do not traverse the WAN, thereby reducing traffic across the wide area links. For SDLC links, polling and poll response occurs locally, not over the WAN. Broadcast of search frames is controlled by the Data Link Switches once the location of a target system is discovered. Finally, the switches can now apply back pressure to the end systems to provide flow and congestion control.

Only one copy of an Link Protocol Data Unit (LPDU) is sent between Data Link Switches in SSP messages (XIDFRAME and INFOFRAME). Retries of the LPDU are absorbed by Data Link Switch that receives it. The Data Link Switch that transmits the LPDU received in an SSP message to a local DLC, will perform retries in a manner appropriate for the local DLC. This may involve running a reply timer and maintaining a poll retry count. The length of the timer and the number of retries is an implementation choice based on user configuration parameters and the DLC type.

Data Link Switching uses LAN addressing to set up connections between SNA systems. SDLC attached devices are defined with MAC and SAP addresses to enable them to communicate with LAN attached devices. For NetBIOS systems, Data Link Switching uses the NetBIOS name to forward datagrams and to set up connections for NetBIOS sessions. For circuit establishment, SNA systems send TEST (or in some cases, XID) frames to the null (0x00) SAP. NetBIOS systems have an address resolution procedure, based upon the Name Query and Name Recognized frames, that is used to establish an end-to-end circuit.

Since Data Link Switching may be implemented in multi-protocol routers, there may be situations where both bridging and switching are enabled. SNA frames can be identified by their link SAP. Typical SAP values for SNA are 0x04, 0x08, and 0x0C. NetBIOS always uses a link SAP value of 0xF0.

3. Transport Connection

Data Link Switches can be in used in pairs or by themselves.

A Single DLSw internally switches one data link to another without using TCP (DLC(1) to DLC(2) in the figure below). This RFC does not go into details on how to implement this feature and it is not a requirement to support this RFC.

A paired DLSw multiplexes data links over a reliable transport using a Switch-to-Switch Protocol (SSP).

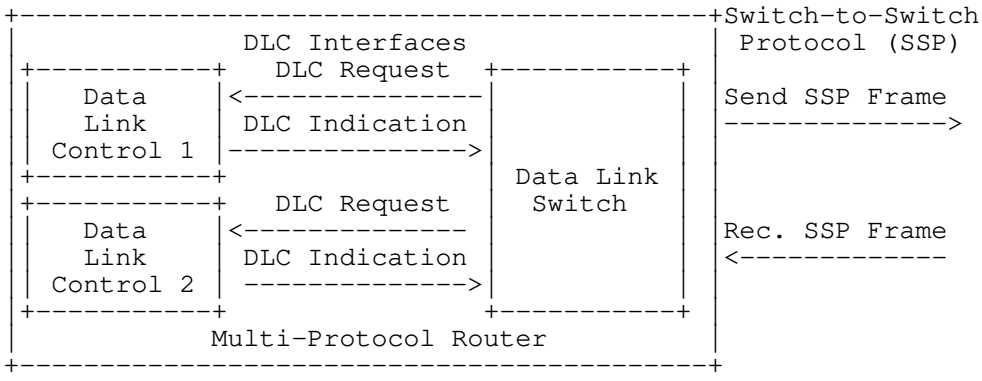


Figure 2. DLSw System Diagram

Before Data Link Switching can occur between two routers, they must establish two TCP connections between them. Each Data Link Switch will maintain a list of DLSw capable routers and their status (active/inactive). After the TCP connection is established, SSP messages are exchanged to establish the capabilities of the two Data Link Switches. Once the exchange is complete, the DLSw will employ SSP control messages to establish end-to-end circuits over the transport connection. Within the transport connection, DLSw SSP messages are exchanged. The message formats and types for these SSP messages are documented in the following sections.

The default parameters associated with the TCP connections between Data Link Switches are as follows:

```
Socket Family     AF_INET         (Internet protocols)
Socket Type       SOCK_STREAM     (stream socket)
Read Port Number  2065
Write Port Number 2067
```

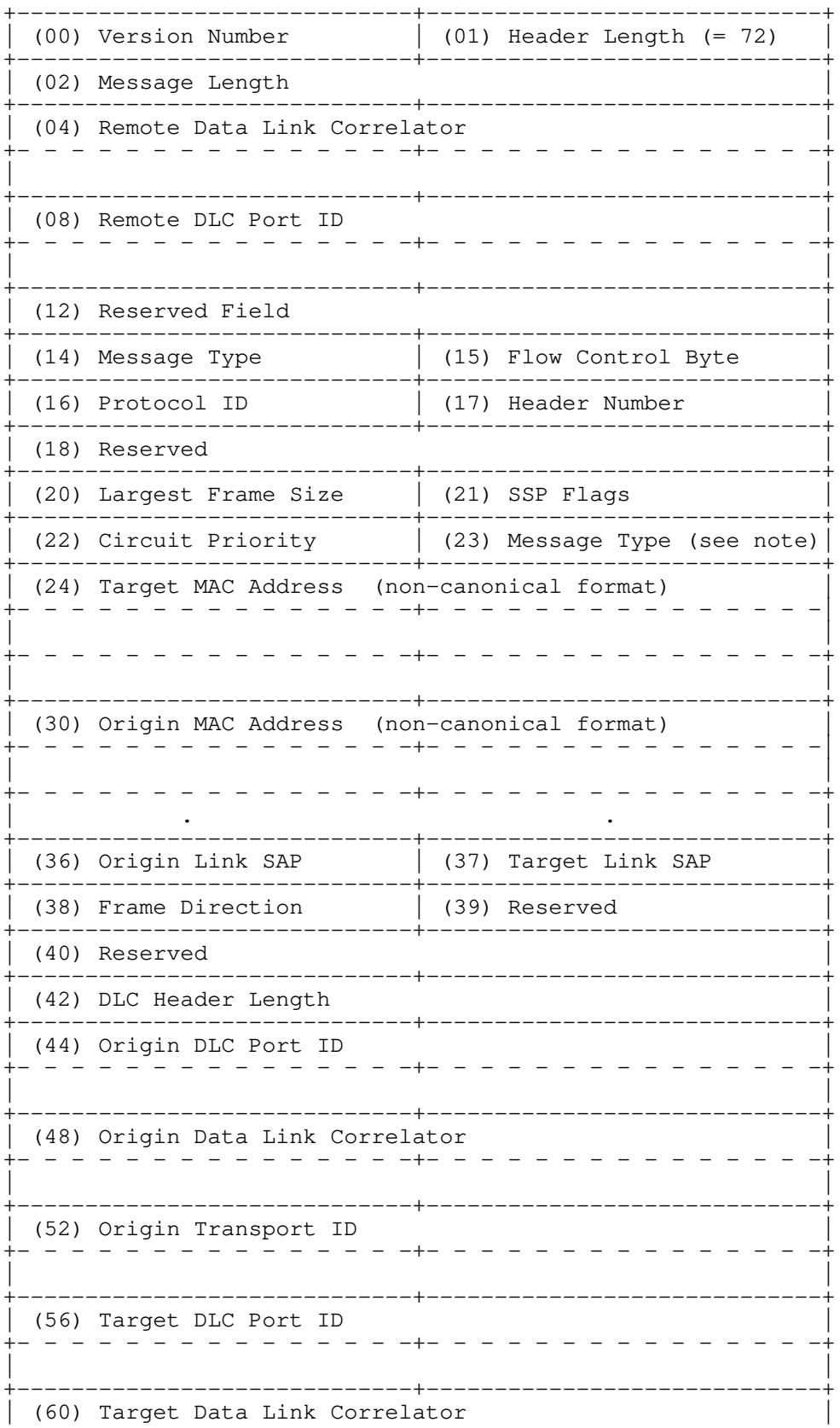
Two or more Data Link Switches may be attached to the same LAN, consisting of a number of token-ring segments interconnected by source-routing bridges. In this case, a TCP connection is not defined between bridges attached to the same LAN. This will allow using systems to select one of the possible Data Link Switches in a similar manner to the selection of a bridge path through a source- routed bridged network. The virtual ring segment in each Data Link Switch attached to a common LAN must be configured with the same ring number. This will prevent LAN frames sent by one Data Link Switch from being propagated through the other Data Link Switches.

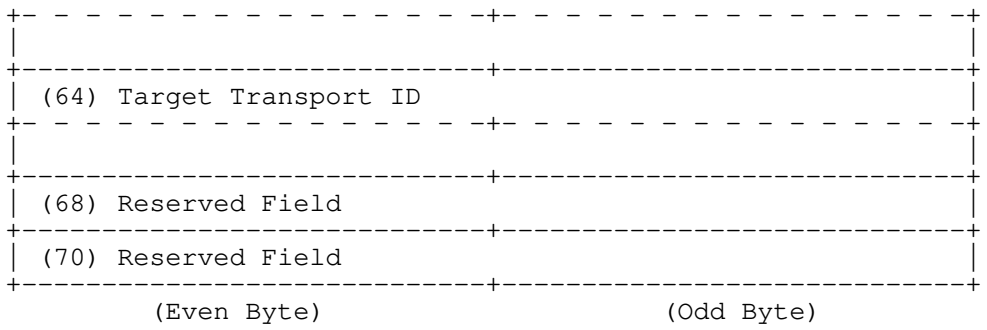
3.1. SSP Frame Formats

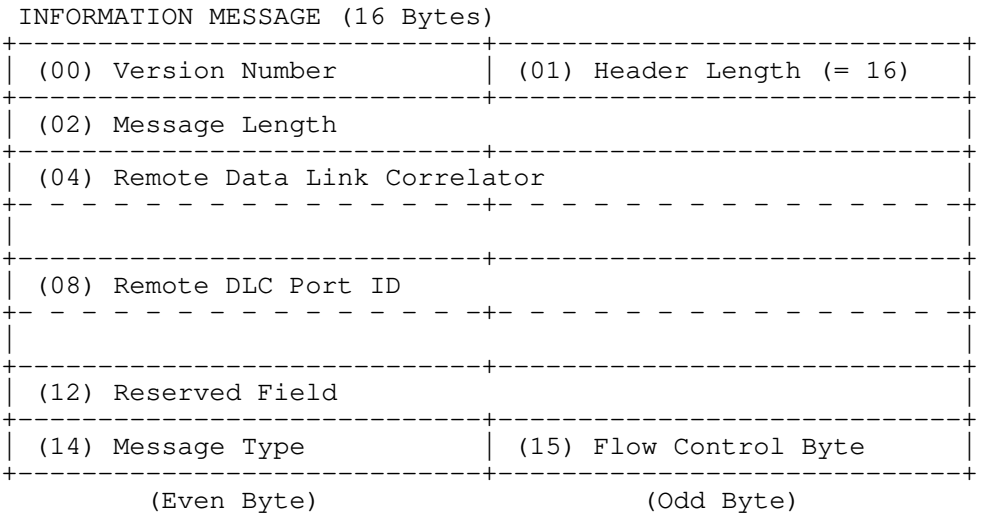
The following diagrams show the two message header formats exchanged between Data Link Switches, Control and Information. The Control message header is used for all messages except Information Frames (INFOFRAME) and Independent Flow Control Messages (IFCM), which are sent in Information header

format. The INFOFRAME, KEEPALIVE and IFCM message headers are 16 bytes long, and the control message header is 72 bytes long. The fields in the first sixteen bytes of all message headers are the same.

CONTROL MESSAGES (72 Bytes)
(zero based offsets below shown in decimal (xx))




(64) Target Transport ID
(68) Reserved Field
(70) Reserved Field
(Even Byte)
(Odd Byte)



The first sixteen bytes of control and information message headers contain identical fields. A brief description of some of the fields in an SSP message are shown below (if not defined below, the fields and/or their values are described in subsequent sections).

The Version Number field (offset 0) is set to 0x31 (ASCII '1'), indicating a decimal value of 49. This is used to indicate DLSw version 1.

The Header Length field (offset 1) is 0x48 for control messages, indicating a decimal value of 72 bytes, and 0x10 for information and Independent Flow Control messages, indicating a decimal value of 16 bytes.

The Message Length field (offset 2) defines the number of bytes within the data field following the header.

The Flow Control Byte field (offset 15) is described in section 8.

The Header Number field (offset 17) is 0x01, indicating a value of one.

The Circuit Priority field (offset 22) is described in section 4.

The Frame Direction field (offset 38) is set to 0x01 for frames sent from the origin DLSw to the target DLSw, and is set to 0x02 for frames sent from the target DLSw to the origin DLSw.

> Note: The Remote Data Link Correlator and Remote DLC Port ID are set equal to the Target Data Link Correlator and Target DLC Port ID if the Frame Direction field is set to 0x01, and are set equal to the Origin Data Link Correlator and Origin DLC Port ID if the Direction Field is set to 0x02.

The Protocol ID field is set to 0x42, indicating a decimal value of 66.

The DLC Header Length is set to zero for SNA and is set to 0x23 for NetBIOS datagrams, indicating a length of 35 bytes. This includes the Access Control (AC) field, the Frame Control (FC) field, Destination MAC Address (DA), the Source MAC Address (SA), the Routing Information (RI) field (padded to 18 bytes), the Destination link SAP (DSAP), the Source link SAP (SSAP), and the LLC control field (UI).

NOTE: The values for the Message Type field are defined in section 3.5. Note that this value is specified in two different fields (offset 14 and 23 decimal) of the control message header. Only the first field is to be used when parsing a received SSP message. The second field is to be ignored by new implementations on

reception. The second field was left in for backwards compatibility with RFC 1434 implementations and this field may be used in future versions if needed.

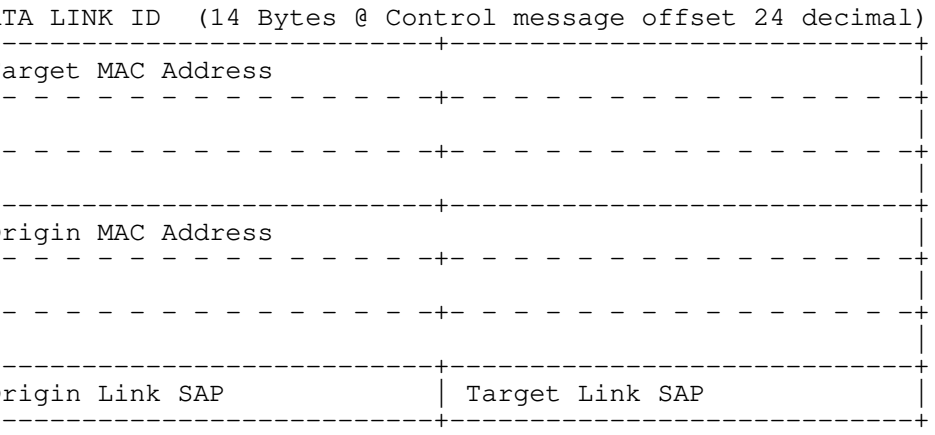
The SSP Flags field contains additional information related to the SSP message. The flags are defined as follows (bit 7 being the most significant bit and bit 0 the least significant bit of the octet):

```
Bit(s)
76543210    Name    Meaning
--------    -----   -------
x.......    SSPex   1 = explorer message (CANUREACH and ICANREACH)
.xxxxxxx            not used
```

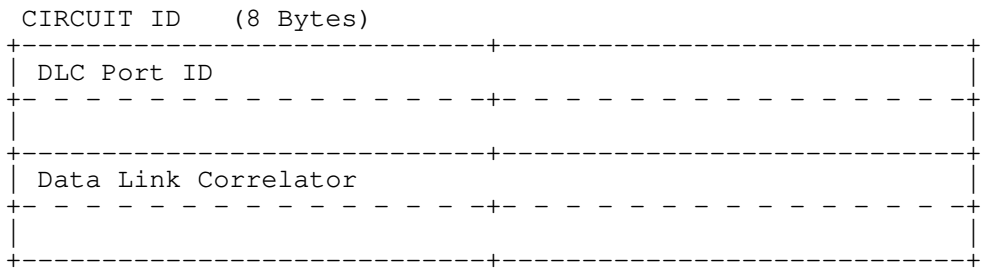
Reserved fields are set to zero upon transmission and should be ignored upon receipt.

3.2. Address Parameters

A data link is defined as a logical association between the two end stations using Data Link Switching. It is identified by a Data Link ID (14 bytes) consisting of the pair of attachment addresses associated with each end system. Each attachment address is represented by the concatenation of the MAC address (6 bytes) and the LLC address (1 byte). Each attachment address is classified as either "Target" in the context of the Destination MAC/SAP addresses of an explorer frame sent in the first frame used to establish a circuit, or "Origin" in the context of the Source MAC/SAP addresses. All MAC addresses are expressed in non-canonical (Token-Ring) format.



An end-to-end circuit is identified by a pair of Circuit ID's. A Circuit ID is a 64 bit number that identifies the DLC circuit within a single DLSw. It consists of a DLC Port ID (4 bytes), and a Data Link Correlator (4 bytes). The Circuit ID must be unique in a single DLSw and is assigned locally. The pair of Circuit ID's along with the Data Link IDs, uniquely identify a single end-to-end circuit. Each DLSw must keep a table of these Circuit ID pairs, one for the local end of the circuit and the other for the remote end of the circuit. In order to identify which Data Link Switch originated the establishment of a circuit, the terms, "Origin" DLSw and "Target" DLSw, will be employed in this document.



The Origin Transport ID and the Target Transport ID fields in the message header are used to identify the individual TCP/IP port on a Data Link Switch. The values have only local significance. However, each Data Link Switch is required to reflect the values contained in these two fields, along with the associated values for DLC Port ID and the Data Link Correlator, when returning a message to the other Data Link Switch.

The following figure shows the use of the addressing parameters during the establishment of an end-to-end connection. The CANUREACH, ICANREACH, and REACH_ACK message types all carry the Data Link ID, consisting of the MAC and Link SAP addresses associated with the two end stations. The CANUREACH and ICANREACH messages are qualified by the SSPex flag into CANUREACH_ex, ICANREACH_ex (explorer messages) and CANUREACH_cs, ICANREACH_cs (circuit start). The CANUREACH_ex is used to find a remote MAC and Link SAP address without establishing an SSP circuit. Upon receipt of a CANUREACH_cs message, the target DLSw starts a data link for each port, thereby obtaining a Data Link Correlator. If the target station can be reached, an ICANREACH_cs message is returned to the origin DLSw containing the Target Circuit ID parameter. Upon receipt, the origin DLSw starts a data link and returns the Origin Circuit ID to the target DLSw within the REACH_ACK message. (Note for a full list of message types, see section 3.5.)

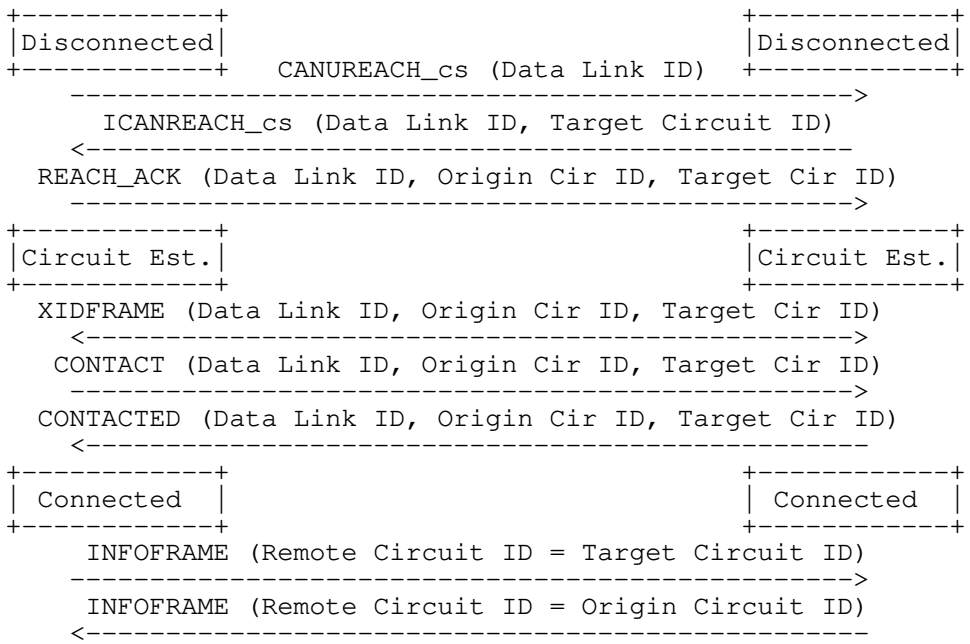


Figure 3. DLSw Circuits and Connections

During the exchange of the XIDFRAME, CONTACT, and CONTACTED messages, the pair of Circuit ID parameters is included in the message format along with the DATA LINK ID parameter. Once the connection has been established, the INFOFRAME messages are exchanged with the shorter header. This header contains only the Circuit ID associated with the remote DLSw. The Remote Data Link Correlator and

the Remote DLC Port ID are set equal to the Data Link Correlator and the DLC Port ID that are associated with the origin or target Data Link Switch, dependent upon the direction of the packet.

3.3. Correlators

The local use, and contents of the Data Link Correlator, Port ID and Transport ID fields in SSP messages is an implementation choice. These fields have local significance only. The values received from a partner DLSw must not be interpreted by the DLSw that receives them and should be echoed "as is" to a partner DLSw in subsequent messages. All implementations must obey the following rules in this section (3.3) on the assignment and fixing of these correlator fields for each transport connection or circuit:

The Transport ID fields are learned from the first SSP message exchanged with a DLSw partner (the Capabilities exchange). This field should not be varied by a DLSw after the capabilities exchange and must be reflected to the partner DLSw in every SSP control message.

The Target Data Link Correlator, Target Port ID and Target Transport ID must remain the same once the Target DLSw has sent the ICANREACH_cs for a given circuit. The Origin DLSw must store the values specified in the ICANREACH_cs and use these on all subsequent SSP messages for this circuit.

The Origin DLSw must allow these fields to vary until the ICANREACH_cs is received. Each SSP message issued for a circuit must reflect the values specified by the Target DLSw in the last SSP message for this circuit received by the Origin DLSw. Binary zero should be used if no such message has yet been received for a given circuit (apart from the Target Transport ID which will have been learnt as specified above).

The Origin Data Link Correlator, Origin Port ID and Origin Transport ID must remain the same once the Origin DLSw has issued the REACH_ACK for a given circuit. The Target DLSw must store the values specified in the REACH_ACK and use these on all subsequent SSP messages for this circuit.

The Target DLSw must allow these fields to vary until the REACH_ACK is received. Each SSP message issued for a circuit must reflect the values specified by the Origin DLSw in the last SSP message for this circuit received by the Target DLSw. Binary zero should be used if no such message has yet been received for a given circuit (apart from the Origin Transport ID which will have been learnt as specified above).

For the purposes of correlator exchange, explorer messages form a separate circuit. Both DLSw partners must reflect the last received correlator values as specified above. However correlators learned on explorer messages need not be carried over to a subsequent circuit setup attempt. In particular, the Origin DLSw may elect to use the same values for the Origin Data Link Correlator and Origin Port ID when it issues a CANUREACH_cs after receiving an ICANREACH_ex or NETBIOS_NR_ex. However the Target DLSw must not assume that the CANUREACH_cs will specify any of the Target Data Link Correlator or Target Port ID that were exchanged on the explorer messages~~frames~~.

Received SSP messages that require a valid Remote Circuit ID but cannot be associated with an existing circuit should be rejected with a HALT_DL_NOACK message. This is done to prevent a situation where one DLSw partner has a circuit defined while the other partner does not. The exception would be a HALT_DL_NOACK message with an invalid Remote Circuit ID. The HALT_DL_NOACK message is typically used in error situations where a response is not appropriate.

The SSP messages requiring a valid Remote Circuit ID are all messages except the following: CANUREACH_ex, CANUREACH_cs, ICANREACH_ex, ICANREACH_cs, NETBIOS_NQ_cs, NETBIOS_NR_cs, DATAFRAME, NETBIOS_ANQ, NETBIOS_ANR, KEEPALIVE and CAP_EXCHANGE.

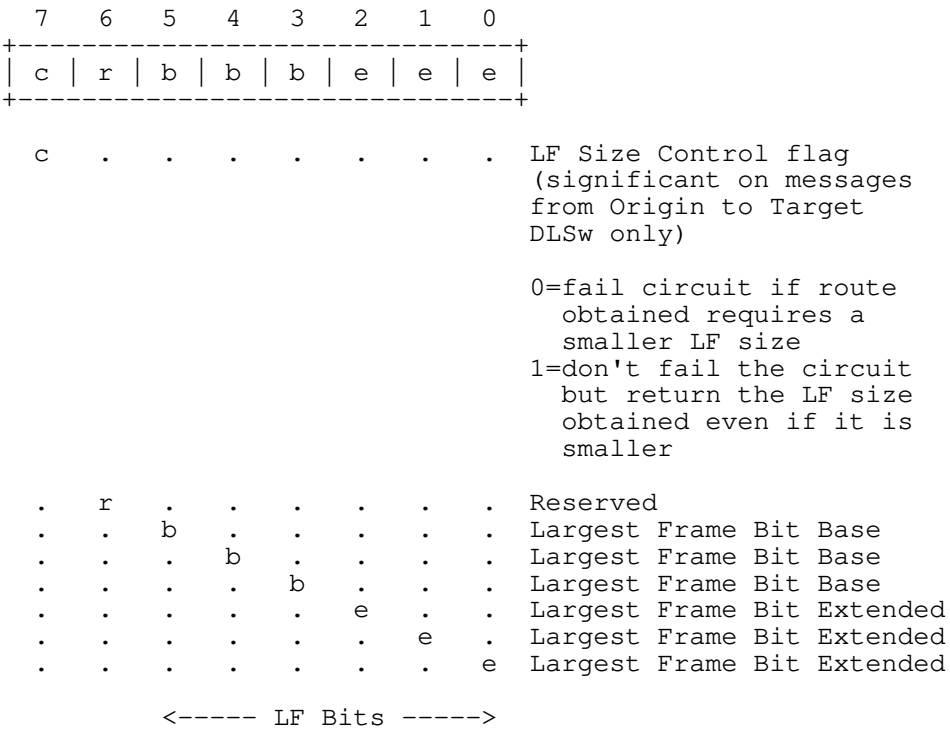
3.4. Largest Frame Size Field

The Largest Frame Size (LF Size) field in the SSP Control Header is used to carry the LF Size bits across the DLSw connection. This should be used to ensure that the two end-stations always negotiate a frame size to be used on a circuit that does not require the Origin and Target DLSw partners to re-segment frames.

This field is valid on CANUREACH_ex, CANUREACH_cs, ICANREACH_ex, ICANREACH_cs, NETBIOS_NQ_ex and NETBIOS_NR_ex messages only. The contents of this field should be ignored on all other frames.

Every DLSw forwarding a SSP frame to its DLSw partner must ensure that the contents of this frame reflect the minimum capability of the route to its local end-station or any limit imposed by the DLSw itself.

The bit-wise definition of this field is as follows (bit 7 is the most significant bit, bit 0 is the least significant bit):



The Origin DLSw "Size Control" flag informs a Target DLSw that chooses to reply to *_cs messages on the basis of cached information that it may safely return a smaller LF Size on the ICANREACH_cs frame if it has had to choose an alternative route on which to initialize the circuit. If this bit is set to 1, the Origin DLSw takes responsibility for ensuring that the end-stations negotiate a suitable frame size for the circuit. If this bit is set to 0, the Target DLSw must not reply to the CANUREACH_cs if it cannot obtain a route to the Target end station that support an LF Size at least as large as that specified in the CANUREACH_cs frame.

3.5. Message Types

The following table lists the protocol data units that are exchanged between Data Link Switches. All values not listed are reserved for potential use in follow-on releases.

Command	Description	Type	flags/notes
CANUREACH_ex	Can U Reach Station-explorer	0x03	SSPex
CANUREACH_cs	Can U Reach Station-circuit start	0x03	
ICANREACH_ex	I Can Reach Station-explorer	0x04	SSPex
ICANREACH_cs	I Can Reach Station-circuit start	0x04	
REACH_ACK	Reach Acknowledgment	0x05	
DGRMFRAME	Datagram Frame	0x06	(note 1)
XIDFRAME	XID Frame	0x07	
CONTACT	Contact Remote Station	0x08	
CONTACTED	Remote Station Contacted	0x09	
RESTART_DL	Restart Data Link	0x10	
DL_RESTARTED	Data Link Restarted	0x11	
ENTER_BUSY	Enter Busy	0x0C	(note 2)
EXIT_BUSY	Exit Busy	0x0D	(note 2)
INFOFRAME	Information (I) Frame	0x0A	
HALT_DL	Halt Data Link	0x0E	
DL_HALTED	Data Link Halted	0x0F	
NETBIOS_NQ_ex	NETBIOS Name Query-explorer	0x12	SSPex
NETBIOS_NQ_cs	NETBIOS Name Query-circuit setup	0x12	(note 3)
NETBIOS_NR_ex	NETBIOS Name Recognized-explorer	0x13	SSPex
NETBIOS_NR_cs	NETBIOS Name Recog-circuit setup	0x13	(note 3)
DATAFRAME	Data Frame	0x14	(note 1)
HALT_DL_NOACK	Halt Data Link with no Ack	0x19	
NETBIOS_ANQ	NETBIOS Add Name Query	0x1A	
NETBIOS_ANR	NETBIOS Add Name Response	0x1B	
KEEPALIVE	Transport Keepalive Message	0x1D	(note 4)
CAP_EXCHANGE	Capabilities Exchange	0x20	
IFCM	Independent Flow Control Message	0x21	
TEST_CIRCUIT_REQ	Test Circuit Request	0x7A	
TEST_CIRCUIT_RSP	Test Circuit Response	0x7B	

Note 1: Both the DGRMFRAME and DATAFRAME messages are used to carry information received by the DLC entity within UI frames. The DGRMFRAME message is addressed according to a pair of Circuit IDs, while the DATAFRAME message is addressed according to a Data Link ID, being composed of a pair of MAC addresses and a pair of link SAP addresses. The latter is employed prior to the establishment of an end-to-end circuit when Circuit IDs have yet to be established or during circuit restart when Data Links are reset.

Note 2: These messages are not used for the DLSw Standard but may be used by older DLSw implementations. They are listed here for informational purposes. These messages were added after publication of RFC 1434 and were deleted in this standard (adaptive pacing is now used instead).

Note 3: These messages are not normally issued by a Standard DLSw, which uses the NB_*_ex messages as shown in section 5.4.. However if a Standard DLSw attempts to interoperate with older DLSw implementations, these messages correspond to the NETBIOS_NQ and NETBIOS_NR messages used in RFC1434 both to locate the resource and to setup a circuit. This document does not attempt to provide a complete specification of the use of these messages.

Note 4: A KEEPALIVE message may be sent by a DLSw to a partner DLSw in order to verify the TCP connection (or other future SSP carrying protocol) is still functioning. If received by a DLSw, this message is discarded and ignored. Use of this message is optional.

For the exchange of NetBIOS control messages, the entire DLC header is carried as part of the message unit. This includes the MAC header, with the routing information field padded to 18 bytes, and the LLC header. The following message types are affected: NETBIOS_NQ, NETBIOS_NR, NETBIOS_ANQ, NETBIOS_ANR, and DATAFRAME when being used by NetBIOS systems. The routing information in the DLC header is not used by the remote Data Link Switch upon receiving the above five messages.

Any SSP message types not defined above if received by a DLSw are to be ignored (i.e. no error action is to be performed). A Data Link Switch should quietly drop any SSP message with a Message Type that is not recognized or not supported. Receipt of such a message should not cause the termination of the transport connection to the message sender.

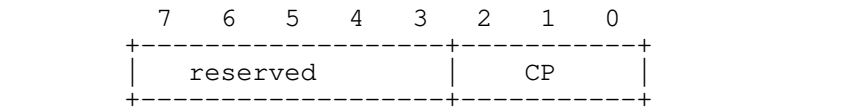
4. Circuit Priority

At circuit start time, each circuit end point will provide priority information to its circuit partner. The initiator of the circuit will choose which circuit priority will be effective for the life of the circuit. If Priority is not implemented by the Data Link Switch, then "Unsupported~~Unspecified~~" priority is used.

4.1. Frame format~~:~~

Circuit priority will be valid in the CANUREACH_cs, ICANREACH_cs, and REACH_ACK frames only. The relevant header field is shown below. The Circuit Priority value is a byte value at offset 22 in an SSP Control Message.

The following describes the format of the Circuit Priority byte.




```
CP: Circuit Priority bits
        000 - Unsupported       (note 1)
        001 - Low Priority      (note 2)
        010 - Medium Priority   (note 2)
        011 - High Priority     (note 2)
        100 - Highest Priority  (note 2)
        101 to 111 are reserved for future use
```

Note 1: Unsupported~~Unspecified~~ means that the Data Link Switch that originates the circuit does not implement priority. Actions taken on Unsupported~~Unspecified~~ priority are vendor specific.

~~Note 2:These values were chosen to correspond to the SNA INN FID4 TH TPF and NTWK_PRTY fields.~~

4.2. Circuit Startup~~:~~

The sender of a CANUREACH_cs is responsible for setting the CP bits to reflect the priority it would like to use for the circuit being requested. The mechanism for choosing an appropriate value is implementation dependent. The sender of an ICANREACH_cs frame will set the CP bits to reflect the priority it would like to use for the circuit being requested, with the mechanism for choosing the appropriate value being implementation dependent. The receiver of the ICANREACH_cs will ~~select from the possible priorities~~ select from the priorities in the CANUREACH_cs and ICANREACH_cs frames, and will set the value in the CP field of the REACH_ACK frame that follows to the value to be used for this circuit. This priority will be used for the life of the circuit. A CANUREACH_cs or ICANREACH_cs with the circuit priority value set to Unsupported (CP=000) indicates that the sender does not support the circuit priority function.

Flow:

```
   DLSw A                DLSw B

CANUREACH_cs (CP=011) ----->          Circuit initiator requests
                                      high Priority.

     <--------- ICANREACH_cs (CP=010) Circuit target requests
                                      medium priority.

REACH_ACK (CP=010) -------->          Circuit initiator sets
                                      the priority for this
                                      circuit to medium. The
                                      circuit initiator could
                                      choose either high or
                                      medium in this example.
~~any valid value.~~
```

5. DLSw State Machine

The following state tables describe the states for a single connection through the Data Link Switch. State information is kept for each connection. The initial state for a connection is DISCONNECTED. The steady state is either CIRCUIT_ESTABLISHED or CONNECTED. In the former state, an end-to-end circuit has been established allowing the support of Type 1 LLC between the end systems. The latter state exists when an end-to-end connection has been established for the support of Type 2 LLC services between the end systems.

For SNA, circuit establishment is via the use of IEEE 802.2 Test or XID frames. SNA devices send these frames to the null SAP in order to determine the source route information in support of bridging. Normally SNA devices use SAP 0x04, 0x08, or 0x0C (most SNA LLC2 devices that have a single PU per MAC address use a default of 0x04). Typically the SAP would be used to determine if the Test frames should be sent to the DLSw code in the router. If both bridging and DLSw are enabled, this allows the product to ensure that SNA frames are not both bridged and switched. Note that although typically SNA uses a DSAP and SSAP of 0x04, it allows for other SAPs to be configured and supports unequal SAPs. This allows multiple PUs to share connections between two given MAC addresses (each PU to PU session uses one LLC2 connection).

For NetBIOS, circuit establishment is via the Name Query and Name Recognized frames. These frames are used for both address resolution and source route determination. NetBIOS devices use SAP 0xF0.

5.1. Data Link Switch States

The Switch-to-Switch Protocol is formally defined through the state machines described in this chapter. The following table lists the eleven possible states for the main circuit FSM. A separate state machine instance is employed for each end-to-end circuit that is maintained by the Data Link Switch.

State Name	Description
CIRCUIT_ESTABLISHED	The end-to-end circuit has been established. At this time LLC Type 1 services are available from end-to-end.
CIRCUIT_PENDING	The target DLSw is awaiting a REACH_ACK response to an ICANREACH_cs message.
CIRCUIT_RESTART	The DLSw that originated the reset is awaiting the restart of the data link and the DL_RESTARTED response to a RESTART_DL message.
CIRCUIT_START	The origin DLSw is awaiting a ICANREACH_cs in response to a CANUREACH_cs message.

CONNECTED	The end-to-end connection has been established thereby allowing LLC Type 2 services from end-to-end in addition to LLC Type 1 services.
CONNECT_PENDING	The origin DLSw is awaiting the CONTACTED response to a CONTACT message.
CONTACT_PENDING	The target DLSw is awaiting the DLC_CONTACTED confirmation to a DLC_CONTACT signal (i.e., DLC is waiting for a UA response to an SABME command).
DISCONNECTED	The initial state with no circuit or connection established, the DLSw is awaiting either a CANUREACH_cs, or an ICANREACH_cs.
DISCONNECT_PENDING	The DLSw that originated the disconnect is awaiting the DL_HALTED response to a HALT_DL message.
HALT_PENDING	The remote DLSw is awaiting the DLC_DL_HALTED indication following the DLC_HALT_DL request (i.e., DLC is waiting for a UA response to a DISC command), due to receiving a HALT_DL message.
HALT_PENDING_NOACK	The remote DLSw is awaiting the DLC_DL_HALTED indication following the DLC_HALT_DL request (i.e., DLC is waiting for a UA response to a DISC command), due to receiving a HALT_DL_NOACK message.
RESTART_PENDING	The remote DLSw is awaiting the DLC_DL_HALTED indication following the DLC_HALT_DL request (i.e., DLC is waiting for a UA response to a DISC command), and the restart of the data link.
RESOLVE_PENDING	The target DLSw is awaiting the DLC_DL_STARTED indication following the DLC_START_DL request (i.e., DLC is waiting for a Test response as a result of sending a Test command).

~~Table 2. Data Link Switch States~~

The DISCONNECTED state is the initial state for a new circuit. One end station starts the connection via an XID or SABME command (i.e., DLC_XID~~DLC_RESOLVE_C~~ or DLC_CONTACTED). Upon receipt, the Data Link Switches exchange a set of CANUREACH_cs, ICANREACH_cs and REACH_ACK messages. Upon completion of this three-legged exchange both Data Link Switches will be in the CIRCUIT_ESTABLISHED state. Three pending states also exist during this exchange. The CIRCUIT_START state is entered by the origin Data Link Switch after it has sent the CANUREACH_cs

message. The RESOLVE_PENDING state is entered by the target Data Link Switch awaiting a Test response to a Test Command. And lastly, the CIRCUIT_PENDING state is entered by the target DLSw awaiting the REACH_ACK reply to an ICANREACH_cs message.

The CIRCUIT_ESTABLISHED state allows for the exchange of LLC Type 1 frames such as the XID exchanges between SNA stations that occurs prior to the establishment of a connection. Also, datagram traffic (i.e., UI frames) may be sent and received between the end stations. These exchanges use the XIDFRAME and DGRMFRAME messages sent between the Data Link Switches.

In the CIRCUIT_ESTABLISHED state, the receipt of a SABME command (i.e., DLC_CONTACTED) causes the origin DLSw to issue a CONTACT message, to send an RNR supervisory frame (i.e., DLC_ENTER_BUSY) to the origin station, and to enter the CONNECT_PENDING state awaiting a CONTACTED message. The target DLSw, upon the receipt of a CONTACT message, will issue a SABME command (i.e., DLC_CONTACT) and enter the Contact Pending state. Once the UA response is received (i.e., DLC_CONTACTED), the target DLSw sends a CONTACTED message and enters the CONNECTED state. When received, the origin DLSw enters the CONNECTED state and sends an RR supervisory frame (i.e., DLC_EXIT_BUSY).

The CONNECTED state is the steady state for normal data flow once a connection has been established. Information frames (i.e., INFOFRAME messages) are simply sent back and forth between the end points of the connection. This is the path that should be optimized for performance.

The connection is terminated upon the receipt of a DISC frame or under some other error condition detected by DLC (i.e., DLC_ERROR). Upon receipt of this indication, the DLSw will halt the local data link, send a HALT_DL message to the remote DLSw, and enter the DISCONNECT_PENDING State. When the HALT_DL frame is received by the other DLSw, the local DLC is halted for this data link, a DL_HALTED message is returned, and the DISCONNECTED state is entered. Receipt of this DL_HALTED message causes the other DLSw to also enter the DISCONNECTED state.

The CIRCUIT_RESTART state is entered if one of the Data Link Switches receives a SABME command (i.e., DLC_RESET) after data transfer while in the CONNECTED state. This causes a DM command to be returned to the origin station and a RESTART_DL message to be sent to the remote Data Link Switch. This causes the remote data link to be halted and then restarted. The remote DLSw will then send a DL_RESTARTED message back to the first DLSw. The receipt of the DL_RESTARTED message causes the first DLSw to issue a new CONTACT message, assuming that the local DLC has been contacted (i.e., the origin station has resent the SABME command). This is eventually responded to by a CONTACTED message. Following this exchange, both Data Link Switches will return to the CONNECTED state. If the local DLC has not been contacted, the receipt of a DL_RESTARTED command causes the Data Link Switch to enter the CIRCUIT_ESTABLISHED state awaiting the receipt of a SABME command (i.e., DLC_CONTACTED signal).

The HALT_PENDING, HALT_PENDING_NOACK and RESTART_PENDING states correspond to the cases when the Data Link Switch is awaiting responses from the local station on the adjacent LAN (e.g., a UA response to a DISC command). Also in the RESTART_PENDING state, the Data Link Switch will attempt to restart the data link prior to sending a DL_RESTARTED message. For some implementations, the start of a data link involves the exchange of a Test command/response on the adjacent LAN (i.e., DLC_START_DL). For other implementations, this additional exchange may not be required.

5.2. State Transition Tables

This section provides a detailed representation of the Data Link Switch, as documented by a single state machine. Many of the transitions are dependent upon local signals between the Data Link Switch entity and one of the DLC entities. These signals and their definitions are given in the following tables.

DLC Events:

Event Name	Description
DLC_CONTACTED	Contact Indication: DLC has received an SABME command or DLC has received a UA response as a result of sending an SABME command.
DLC_DGRM	Datagram Indication: DLC has received a UI frame.
DLC_ERROR	Error condition indicated by DLC: Such a condition occurs when a DISC command is received or when DLC experiences an unrecoverable error.
DLC_INFO	Information Indication: DLC has received an Information (I) frame.
DLC_DL_HALTED	Data Link Halted Indication: DLC has received a UA response to a DISC command.
DLC_DL_STARTED	Data Link Started Indication: DLC has received a Test response from the null SAP.
DLC_RESET	Reset Indication: DLC has received an SABME command during the time a connection is currently active and has responded with DM.
DLC_RESOLVE_C	Resolve Command Indication: DLC has received a Test command addressed to the null SAP, or an XID command addressed to the null SAP.
DLC_RESOLVED	Resolve request: DLC has received a TEST response frame (or equivalent for non-LAN DLCs) but has not reserved the resources required for a circuit yet.
DLC_XID	XID Indication: DLC has received an XID command or response to a non-null SAP.

Other Events:

Event Name	Description
XPORT_FAILURE	Failure of the transport connection used by the circuit.
CS_TIMER_EXP	The CIRCUIT_START timer (started when the circuit went into CIRCUIT_START state) has expired.

Table 3. Local DLC Events

DLC Actions:

Action Name	Description
DLC_CONTACT	Contact Station Request: DLC will send a SABME command or a UA response to an outstanding SABME command.
DLC_DGRM	Datagram Request: DLC will send a UI frame.
DLC_ENTER_BUSY	Enter Link Station Busy: DLC will send an RNR supervisory frame.
DLC_EXIT_BUSY	Exit Link Station Busy: DLC will send an RR supervisory frame.
DLC_HALT_DL	Halt Data Link Request: DLC will send a DISC command.
DLC_INFO	Information Request: DLC will send an I frame.
DLC_RESOLVE	Resolve request: DLC should issue a TEST (or appropriate equivalent for non-LAN DLCs) but need not reserve the resources required for a circuit yet.
DLC_RESOLVE_R	Resolve Response Request: DLC will send a Test response or XID response from the null SAP.
DLC_START_DL	Start Data Link Request: DLC will send a Test command to the null SAP.
DLC_XID	XID Request: DLC will send an XID command or an XID response.

Other Actions:

Action Name	Description
START_CS_TIMER	Start the CIRCUIT_START timer.

Table 4. Local DLC Actions

DLC_RESOLVE_R and DLC_START_DL actions require the DLC to reserve the resources necessary for a link station as they are used only when a circuit is about to be started. The DLC_RESOLVE action is used for topology explorer traffic and does not require such resources to be reserved, though a DLC implementation may choose not to distinguish this from the DLC_START_DL action. See section 5.4 for details of the actions and events for explorer frames.

The Data Link Switch is described by a state transition table as documented in the following sections. Each of the states is described below in terms of the events, actions, and next state for each transition. If a particular event is not listed for a given state, no action and no state transition should occur for that event. Any significant comments concerning the transitions within a given state are given immediately following the table representing the state.

A separate state machine instance is maintained by the Data Link Switch for each end-to-end circuit. The number of circuits that may be supported by each Data Link Switch is a local implementation option.

The CANUREACH_ex, ICANREACH_ex, NETBIOS_NQ_ex, and NETBIOS_NR_ex are SSP messages that are not associated with a particular circuit. The processing of these messages is covered in section 5.4.

5.2.1. DISCONNECTED State

Event	Action(s)	Next State
Receive CANUREACH_cs	DLC_START_DL	RESOLVE_PENDING
Receive DATAFRAME	DLC_DGRM	
DLC_XID	If source route bridged frame with broadcast indicated: Send CANUREACH_ex else: Send CANUREACH_cs START_CS_TIMER	If CANUREACH_cs was sent: CIRCUIT_START
DLC_DGRM	If NETBIOS NAME_QUERY: Send NETBIOS_NQ_ex else: Send DATAFRAME	
DLC_CONTACTED	Send CANUREACH_cs	CIRCUIT_START

It is assumed that each Data Link Switch will build a set of topology tables giving the identity of each Data Link Switch that can reach a specific MAC address or a specific NetBIOS name. This table can be built using the explorer frames, as per the Explorer FSM in section 5.4. As a consequence, the amount of search traffic can be kept to a minimum.

Upon receipt of a TEST command, broadcast XID~~, SABME command (in case of no XID exchange)~~ or NetBIOS NAME_QUERY, the Data Link Switch checks the topology table for the target MAC/SAP or NetBIOS name. If there is no matching entry in the table, the Data Link Switch uses the explorer FSMs in section 5.4 to locate the target MAC/SAP or NetBIOS name.

When the first non-broadcast XID or SABME flows, the Data Link Switch issues a CANUREACH_cs to attempt to start a circuit. The CANUREACH_cs message is sent to only those Data Link Switches that are known to be able to reach the given MAC address. The mechanism by which a topology table entry is determined to be out-of-date and is deleted from the table is implementation specific.

The DISCONNECTED state is exited upon the sending or receipt of a CANUREACH_cs message by a prospective target Data Link Switch. In the former case, the Data Link Switch will issue a Test command to the target station (i.e. DLC_START_DL signal is presented to DLC).

Note 1- Some implementations will also send a Test command to the station that originated the resolve procedure (i.e., a DLC_START_DL will be issued) in order to start the data link.

5.2.2. RESOLVE_PENDING State

Event	Action(s)	Next State
Receive DATAFRAME	DLC_DGRM	

DLC_DL_STARTED	If LF value of DLC_DL_STARTED is greater than or equal to LF Size of CANUREACH_cs or LF Size Control bit set:
	  Send ICANREACH_cs
	else:
	  Send DLC_HALT_DL
		If LF value of DLC_DL_STARTED is greater than or equal to LF Size of CANUREACH_cs or LF Size Control bit set:
		  CIRCUIT_PENDING
		else:
		  HALT_PENDING_NOACK

DLC_ERROR		DISCONNECTED
DLC_DGRM	Send DATAFRAME	

The RESOLVE_PENDING state is entered upon receipt of a CANUREACH_cs message by the target DLSw. A data link is started, causing a Test command to be sent by the DLC.

Several CANUREACH_cs messages can be received in the RESOLVE_PENDING state. The Data Link Switch may update its topology information based upon the origin MAC address information in each CANUREACH_cs message.

Upon the receipt of a DLC_DL_STARTED signal in the RESOLVE_PENDING state, the Data Link Switch may update its topology table base upon the remote MAC address information. The ICANREACH_cs message must be returned to the first partner DLSw from which a CANUREACH_cs was received for this circuit, or an implementation may optionally reply to all partners from which the CANUREACH_cs was received.

The RESOLVE_PENDING state is exited once the data link has been started (i.e., a DLC_DL_STARTED signal is received as a result of a Test response received by the DLC). The target Data Link Switch then enters the CIRCUIT_PENDING state.

5.2.3. CIRCUIT_START State

Event	Action(s)	Next State

Receive CANUREACH_cs for circuit in opposite direction	If origin MAC addr in CANUREACH_cs is greater than origin MAC addr of circuit:
	  DLC_START_DL
	else:
	  no action taken
		If DLC_START_DL issued:
		  RESOLVE_PENDING

Receive ICANREACH_cs	If LF Size Control bit set and LF Size is not negotiable:
	  Send HALT_DL_NOACK
	else:
	  Send REACH_ACK,
	  Send appropriate SSP message based on the event that generated CANUREACH_cs
	  (see Note)
		If LF Size Control bit set and LF Size is not negotiable:
		  DISCONNECTED
		else if Connected:
		  CONNECT_PENDING
		else:
		  CIRCUIT_ESTABLISHED

DLC_DGRM	Send DATAFRAME	
DLC_ERROR		DISCONNECTED
CS_TIMER_EXP		DISCONNECTED
XPORT_FAILURE		DISCONNECTED

The CIRCUIT_START state is entered by the origin Data Link Switch when a DLC_XID or DLC_CONTACTED signal has been received from the DLC.

The CIRCUIT_START state is exited upon receipt of an ICANREACH_cs message. A REACH_ACK message is returned to the target Data Link Switch. If the CIRCUIT_START state was entered due to a DLC_XID signal, an XIDFRAME message containing the XID is sent to the target Data Link Switch. If the CIRCUIT_START state was entered due to a DLC_CONTACTED signal, a CONTACT message is sent to the target Data Link Switch.

Note: For example, DLC_XID - send XIDFRAME, DLC_CONTACTED - send CONTACT message

5.2.4. CIRCUIT_PENDING State

Event	Action(s)	Next State
Receive CONTACT	DLC_CONTACT	CONTACT_PENDING
Receive HALT_DL	DLC_HALT_DL	HALT_PENDING
Receive HALT_DL_NOACK	DLC_HALT_DL	HALT_PENDING_NOACK
Receive REACH_ACK	If Connected: Send CONTACT	If Connected: CONNECT_PENDING, else: CIRCUIT_ESTABLISHED
Receive XIDFRAME	DLC_XID	
Receive DGRMFRAME	DLC_DGRM	
Receive DATAFRAME	DLC_DGRM	
DLC_CONTACTED	If UA is sent in response to SABME: DLC_ENTER_BUSY else: no action taken	
DLC_ERROR		DISCONNECTED
~~DLC_ERROR~~	~~Send HALT_DL~~	~~DISCONNECT_PENDING~~
DLC_XID	Drop or hold until REACH_ACK received	
DLC_DGRM	Send DATAFRAME	
XPORT_FAILURE	DLC_HALT_DL	HALT_PENDING_NOACK

The CIRCUIT_PENDING state is entered by the target Data Link Switch following the sending of an ICANREACH_cs message. In this state it is awaiting the reception of a REACH_ACK message from the origin Data Link Switch.

If the target Data Link Switch happens to receive a SABME command from the target station while in the CIRCUIT_PENDING state (i.e., a DLC_CONTACTED signal received from the DLC), the reception of the REACH_ACK message causes the target Data Link Switch to enter the CONNECT_PENDING state and to send a CONTACT message to the origin Data Link Switch.

If no such SABME is received, the receipt of the REACH_ACK causes the Data Link Switch to enter CIRCUIT_ESTABLISHED state.

5.2.5. CONNECT_PENDING State

Event	Action(s)	Next State
Receive CONTACTED	If UA was sent in response to SABME: DLC_EXIT_BUSY else: DLC_CONTACT	CONNECTED
Receive HALT_DL	DLC_HALT_DL	HALT_PENDING
Receive HALT_DL_NOACK	DLC_HALT_DL	HALT_PENDING_NOACK
Receive DGRMFRAME	DLC_DGRM	
Receive DATAFRAME	DLC_DGRM	
Receive ICANREACH_cs	Send HALT_DL_NOACK	
DLC_RESET	Send RESTART_DL	CIRCUIT_RESTART
DLC_ERROR	Send HALT_DL	DISCONNECT_PENDING
DLC_DGRM	Send DGRMFRAME	
XPORT_FAILURE	DLC_HALT_DL	HALT_PENDING_NOACK

The CONNECT_PENDING state is entered when a DLC_CONTACTED signal has been received from the DLC (i.e., a SABME command has been received). A CONTACT message it then issued. The state is exited upon the receipt of a CONTACTED message. If a DLC_RESET signal is received, the local data link is restarted and a RESTART_DL message is sent to the remote DLSw.

An ICANREACH_cs received after the transition to CONNECT_PENDING state indicates that more than one CANUREACH_cs was sent at circuit establishment time and the target station was found by more than one Data Link Switch partner. A HALT_DL_NOACK is sent to halt the circuit started by the Data Link Switch partner that originated each such ICANREACH_cs.

Note: Some implementations will also send a Test command in order to restart the data link to the station that sent the SABME command (i.e., a DLC_START_DL will be issued).

5.2.6. CIRCUIT_ESTABLISHED State

Event	Action(s)	Next State
Receive CONTACT	DLC_CONTACT	CONTACT_PENDING
Receive HALT_DL	DLC_HALT_DL	HALT_PENDING
Receive HALT_DL_NOACK	DLC_HALT_DL	HALT_PENDING_NOACK
Receive XIDFRAME	DLC_XID	
Receive DGRMFRAME	DLC_DGRM	
Receive DATAFRAME	DLC_DGRM	
Receive ICANREACH_cs	Send HALT_DL_NOACK	

DLC_CONTACTED	Send CONTACT
	If UA is sent in response to SABME:
	  DLC_ENTER_BUSY
	else:
	  no action taken
		CONNECT_PENDING

DLC_ERROR	Send HALT_DL	DISCONNECT_PENDING
DLC_DGRM	Send DGRMFRAME	
DLC_XID	Send XIDFRAME	
XPORT_FAILURE	DLC_HALT_DL	HALT_PENDING_NOACK

The CIRCUIT_ESTABLISHED state is entered by the origin Data Link Switch from the CIRCUIT_START state, and by the target Data Link Switch from the CIRCUIT_PENDING state. The state is exited when a connection is started (i.e., DLC receives a SABME command) or CONTACT is received. The next state is CONTACT_PENDING or CONNECT_PENDING.

An ICANREACH_cs received after the transition to CIRCUIT_ESTABLISHED state indicates that more than one CANUREACH_cs was sent at circuit establishment time and the target station was found by more than one Data Link Switch partner. A HALT_DL_NOACK is sent to halt the circuit started by the Data Link Switch partner that originated each such ICANREACH_cs.

5.2.7. CONTACT_PENDING State

Event	Action(s)	Next State
Receive HALT_DL	DLC_HALT_DL	HALT_PENDING
Receive HALT_DL_NOACK	DLC_HALT_DL	HALT_PENDING_NOACK
Receive RESTART_DL	DLC_HALT_DL	RESTART_PENDING
Receive DGRMFRAME	DLC_DGRM	
Receive DATAFRAME	DLC_DGRM	
DLC_CONTACTED	Send CONTACTED	CONNECTED
DLC_ERROR	Send HALT_DL	DISCONNECT_PENDING
DLC_DGRM	Send DGRMFRAME	
XPORT_FAILURE	DLC_HALT_DL	HALT_PENDING_NOACK

The CONTACT_PENDING state is entered upon the receipt of a CONTACT message, which causes the Data Link Switch to issue a DLC_CONTACT signal to the DLC (i.e., DLC sends a SABME command). This state is then exited upon the receipt of a DLC_CONTACTED signal from the DLC (i.e., a UA response received).

If a RESTART_DL message is received, indicating that the remote Data Link Switch has received a DLC_RESET signal, the local Data Link Switch sends a DISC command frame on the adjacent LAN (i.e., DLC_HALT_DL signal) and enter the RESTART_PENDING state.

An ICANREACH_cs received after the transition to CONTACT_PENDING state indicates that more than one CANUREACH_cs was sent at circuit establishment time and the target station was found by more than one Data Link Switch partner. A HALT_DL_NOACK is sent to halt the data link started by the Data Link Switch partner that originated this ICANREACH_cs.

5.2.8. CONNECTED State

Event	Action(s)	Next State
Receive HALT_DL	DLC_HALT_DL	HALT_PENDING
Receive HALT_DL_NOACK	DLC_HALT_DL	HALT_PENDING_NOACK
Receive RESTART_DL	DLC_HALT_DL	RESTART_PENDING
Receive DGRMFRAME	DLC_DGRM	
Receive INFOFRAME	DLC_INFO	
Receive DATAFRAME	DLC_DGRM	
Receive XIDFRAME	If non-activation XID3: DLC_XID	
Receive ICANREACH_cs	Send HALT_DL_NOACK	
Receive ENTER_BUSY	DLC_ENTER_BUSY	
Receive EXIT_BUSY	DLC_EXIT_BUSY	
Rec TEST_CIRCUIT_REQ	Snd TEST_CIRCUIT_RSP	
DLC_RESET	Send RESTART_DL	CIRCUIT_RESTART
DLC_ERROR	Send HALT_DL	DISCONNECT_PENDING
DLC_DGRM	Send DGRMFRAME	
DLC_INFO	Send INFOFRAME	
DLC_XID	If non-activation XID3: Send XIDFRAME	
XPORT_FAILURE	DLC_HALT_DL	HALT_PENDING_NOACK

The CONNECTED state is entered from the CONNECT_PENDING state upon the receipt of a CONTACTED message or from the CONTACT_PENDING state upon the receipt of a DLC_CONTACTED signal.

The CONNECTED state is exited usually under one of two conditions: a DLC_ERROR signal received from the DLC (e.g., a DISC command received by the local DLC), or a HALT_DL message received from the other Data Link Switch (e.g., a DISC command received by the remote DLC).

A SABME command (i.e., a DLC_RESET signal) received by either Data Link Switch will also cause the two Data Link Switches to leave the CONNECTED state and attempt to restart the circuit. Following the receipt of a SABME, the local Data Link Switch sends a RESTART_DL message to the other Data Link Switch and enters the CIRCUIT_RESTART state. Upon the receipt of the RESTART_DL message, the remote Data Link Switch sends a DISC command (i.e., DLC_HALT_DL signal) and enters the RESTART_PENDING state.

An ICANREACH_cs received after the transition to CONNECTED state indicates that more than one CANUREACH_cs was sent at circuit establishment time and the target station was found by more than one Data Link Switch partner. A HALT_DL_NOACK is sent to halt the circuit started by the Data Link Switch partner that originated each such ICANREACH_cs.

Note: Some implementations will also send a Test command in order to restart the data link to the station that sent the SABME command (i.e., a DLC_START_DL will be issued).

5.2.9. CIRCUIT_RESTART State

Event	Action(s)	Next State
Receive DL_RESTARTED	If Connected: Send CONTACT	If Connected: CONNECT_PENDING, else: CIRCUIT_ESTABLISHED
Receive HALT_DL_NOACK	DLC_HALT_DL	HALT_PENDING_NOACK
Receive DGRMFRAME	DLC_DGRM	
DLC_ERROR	Send HALT_DL	DISCONNECT_PENDING
DLC_DGRM	Send DGRMFRAME	
XPORT_FAILURE	DLC_HALT_DL	HALT_PENDING_NOACK

The CIRCUIT_RESTART state is entered if a DLC_RESET signal is received from the local DLC. This was caused by the receipt of a SABME command while a connection was currently active. A DM response will be issued to the SABME command and the Data Link Switch will attempt to restart the end-to- end circuit.

The CIRCUIT_RESTART state is exited through one of two transitions. The next state depends upon the time the local DLC has reached the contacted state (i.e., a DLC_CONTACTED signal is presented) relative to the receipt of the DL_RESTARTED message. This signal is caused by the origin station resending the SABME command that initially caused the Data Link Switch to enter the CIRCUIT_RESTART state. The two cases are as follows:

1) DL_RESTARTED message received before the DLC_CONTACTED signal- In this case, the CIRCUIT_ESTABLISHED state is entered.

2) DL_RESTARTED message received after the DLC_CONTACTED signal- In this case, the CONNECT_PENDING state is entered.

5.2.10. DISCONNECT_PENDING State

Event	Action(s)	Next State
Receive DL_HALTED		DISCONNECTED
Receive HALT_DL	Send DL_HALTED	
Receive HALT_DL_NOACK		DISCONNECTED
Receive DATAFRAME	DLC_DGRM	
DLC_DGRM	Send DATAFRAME	
XPORT_FAILURE		DISCONNECTED

The DISCONNECT_PENDING state is entered when a DLC_ERROR signal is received from the local DLC. Upon receipt of this signal, a HALT_DL message is sent. Once an DL_HALTED message is received, the state is exited, and the Data Link Switch enters the DISCONNECTED state.

5.2.11. RESTART_PENDING State

Event	Action(s)	Next State
Receive HALT_DL_NOACK		HALT_PENDING_NOACK
Receive DGRMFRAME	DLC_DGRM	
DLC_DL_HALTED	Send DL_RESTARTED	CIRCUIT_ESTABLISHED
DLC_ERROR	Send HALT_DL	DISCONNECT_PENDING
DLC_DGRM	Send DGRMFRAME	
XPORT_FAILURE	DLC_HALT_DL	HALT_PENDING_NOACK

The RESTART_PENDING state is entered upon the receipt of a RESTART_DL message from the remote DLSw while the local Data Link Switch is in either the CONTACT_PENDING state or the CONNECTED state, which causes the local DLSw to issue a DISC command to the DLC. Upon the receipt of the UA response (DLC_DL_HALTED), the data link is restarted, a DL_RESTARTED message is returned to the remote DLSw, and the CIRCUIT_ESTABLISHED state is entered.

Note: Some implementations will send a Test command in order to restart the data link to the target station (i.e., a DLC_START_DL will be issued) prior to sending the DL_RESTARTED message.

5.2.12. HALT_PENDING State

Event	Action(s)	Next State
Receive HALT_DL_NOACK		HALT_PENDING_NOACK
Receive DATAFRAME	DLC_DGRM	
DLC_DL_HALTED	Send DL_HALTED	DISCONNECTED
DLC_ERROR	Send DL_HALTED	DISCONNECTED
DLC_DGRM	Send DATAFRAME	
XPORT_FAILURE		HALT_PENDING_NOACK

The HALT_PENDING state is entered upon the receipt of a HALT_DL message. This causes the local DLC to issue a DISC command. Upon the receipt of the UA response (DLC_DL_HALTED), a DL_HALTED message is returned to the remote DLSw and the DISCONNECTED state is entered.

5.2.13. HALT_PENDING_NOACK State

Event	Action(s)	Next State
Receive DATAFRAME	DLC_DGRM	
DLC_DL_HALTED		DISCONNECTED
DLC_ERROR		DISCONNECTED
DLC_DGRM	Send DATAFRAME	

The HALT_PENDING_NOACK state is entered upon the receipt of a HALT_DL_NOACK message. This causes the local DLC to issue a DISC command. Upon the receipt of the UA response (DLC_DL_HALTED), the DISCONNECTED state is entered.

5.3. NetBIOS Datagrams

The NetBIOS protocols use a number of UI frames for directory services and the transmission of datagrams. Most of these frames are directed to a group MAC address (GA) with the routing information field indicating spanning tree explorer (STE) (a.k.a. Single Route Broadcast). The NB_Add_Name_Response and NB_Name_Recognized frames are directed to a specific MAC address with the routing information field indicating an all routes explorer frame (ARE) (a.k.a. All Routes Broadcast) The NB_Status_Response frame, is directed to a specific MAC address with the routing information field indicating a specifically routed frame (SRF). The handling of these frames is summarized in the following table.

Event	Action(s)	Comment
DLC_DGRM for NETBIOS group address: NB_Add_Name_Query	Send NETBIOS_ANQ	Transmitted to all remote DLSw
DLC_DGRM for a specific address: NB_Add_Name_Response	Send NETBIOS_ANR	Transmitted to specific DLSw
DLC_DGRM for a specific address: NB_Status_Response	Send DATAFRAME	Transmitted to all remote DLSw
DLC_DGRM for NETBIOS group address: NB_Name_in_Conflict NB_Add_Group_Name_Query NB_Datagram, NB_Datagram_Broadcast NB_Status_Query NB_Terminate_Trace	Send DATAFRAME	Transmitted to all remote DLSw

Table 5. NetBIOS DLC Frames

The above actions do not apply in the following states: CIRCUIT_ESTABLISHED, CONTACT_PENDING, CONNECT_PENDING, CONNECTED, and CIRCUIT_PENDING. The handling of the remaining two UI frames used by NetBIOS systems, NB_Name_Query and NB_Name_Recognized, are documented as part of the DLSw state machine in the previous section (i.e., DISCONNECTED and RESOLVE_PENDING states). Furthermore, the handling of NetBIOS datagrams (i.e., NB_Datagram) sent to a specific MAC address is also governed by the DLSw state machine.

Note: Some implementations also issue Test frames during the exchange of the NetBIOS, NB_Name_Query and NB_Name_Recognized. This exchange of protocol data units occurs during the start of a data link and is used to determine the routing information. Most other implementations of NetBIOS will use the NB_Name_Query/NB_Name_Recognized exchange to determine routes in conjunction with resolving the NetBIOS names. These differences are not reflected in the SSP protocols.

The handling of the NetBIOS specific SSP messages is given in the following table.

Event	Action(s)	Comment
NETBIOS_ANQ	DLC_DGRM: NB_Add_Name_Query	Routed STE (NETBIOS Group Address)
NETBIOS_ANR	DLC_DGRM: NB_Add_Name_Response	Routed ARE (Specific MAC Address)
NETBIOS_NQ_ex	DLC_DGRM: NB_Name_Query	Routed STE (NETBIOS Group Address)
NETBIOS_NQ_cs	DLC_DGRM: NB_Name_Query	Routed STE (NETBIOS Group Address)
NETBIOS_NR_ex	DLC_DGRM: NB_Name_Recognized	Routed ARE (Specific MAC Address)
NETBIOS_NR_cs	DLC_DGRM: NB_Name_Recognized	Routed ARE (Specific MAC Address)
DATAFRAME	DLC_DGRM	If NB_Status_Response: Routed ARE (Specific MAC Address) Else: Routed STE (NETBIOS Group Address)

Table 6. NetBIOS SSP Messages

The above actions apply to all DLSw states. The handling of NetBIOS datagrams sent within DGRMFRAME messages is governed by the DLSw state machine. The DGRMFRAME message type is employed instead of the DATAFRAME message type once the end-to-end circuit has been established. At that time, the message is addressed according to the pair of Circuit IDs in the message header instead of relying upon the MAC address information in the token ring header.

5.4. Explorer Traffic

The CANUREACH_ex, ICANREACH_ex, NETBIOS_NQ_ex, and NETBIOS_NR_ex SSP messages explore the topology of the DLSw cloud and the networks attached to it. These explorer frames are used to determine the DLSw partners through which a MAC or NetBIOS name can be accessed. This information may optionally be cached to reduce explorer traffic in the DLSw cloud.

If a DLSw is aware from cached information that a given MAC address or NetBIOS name is accessible through a given partner DLSw, it should direct all circuit setup attempts to that partner. If the circuit setup fails, or no such data is available in the MAC or name cache database, the DLSw may fallback to issuing the setup attempt to all DLSw partners on the assumption that the cached data is now out of date. The mechanism for determining when to use such a fallback is implementation defined.

DLSw implementations may also use a local MAC cache to enable responses to CANUREACH_ex requests to be issued without the need for TEST frame exchange (or equivalent) until the CANUREACH_cs is received. Again, the fallback mechanism for determining when such local cache data is out-of-date is implementation defined.

The use of either cache is an optional function in DLSw. An implementation may choose to always issue explorer frames or to use either or both types of cache.

The following sections describe the FSMs used for explorer frames. The DLC events and actions are a subset of those described in section 5.2 for the main circuit FSM.

5.4.1. CANUREACH/ICANREACH Explorer FSM

The FSM described below is used to handle explorer frames routed by MAC address. There is one instance of this FSM for each Data Link ID (Target and Origin MAC/SAP pair) for which explorer traffic is flowing. The states in this FSM are as follows.

State Name	Description
RESET	The initial state.
SENT_EX	Local DLSw has issued an explorer message
RECEIVED_EX	Local DLSw has received an explorer message

5.4.1.1. RESET State

Event	Action(s)	Next State
Receive CANUREACH_ex	If replying from cache, send ICANREACH_ex else if allowed to test availability, issue DLC_RESOLVE. Optionally update cache.	If DLC_RESOLVE sent, RECEIVED_EX
Receive ICANREACH_ex	Optionally update cache	RESET
DLC_RESOLVE_C	Send CANUREACH_ex	SENT_EX

RESET is the initial state for the CANUREACH/ICANREACH explorer FSM. This state is exited when a DLC_RESOLVE_C request is received from the DLC or a CANUREACH_ex is received from a remote DLSw.

A DLSw implementation may optionally reply from to CANUREACH_ex messages on the basis of cached topology information, in which case the DLC_RESOLVE exchange (i.e. TEST) is not required. If cache is not used, or no match is found, and the DLC permits the use of TEST, DLC_RESOLVE is issued to locate the target MAC and the state changes to RECEIVED_EX. If no cache entry is available and TEST is not allowed by the DLC, a received CANUREACH_ex frame is ignored.

5.4.1.2. SENT_EX State

Event	Action(s)	Next State
Receive ICANREACH_ex	DLC_RESOLVE_R Optionally update cache	RESET
DLC_RESOLVE_C		SENT_EX

SENT_EX is entered when the DLSw has issued a CANUREACH_ex message to locate a MAC address. This state is exited when a remote DLSw returns a matching ICANREACH_ex, or after an implementation defined timeout. DLC_RESOLVE events received in this state correspond to TEST retries by the origin DLC station and are absorbed.

An implementation may choose whether to handle explorer frame crossover either by using entirely separate FSM instances and simply allowing both ends to issue TEST frames, or by detecting a reverse CANUREACH_ex frame here and issuing an ICANREACH_ex message and DLC_RESOLVE_R action.

5.4.1.3. RECEIVED_EX State

Event	Action(s)	Next State
Receive CANUREACH_ex	Optionally update cache	RECEIVED_EX
Receive ICANREACH_ex		RECEIVED_EX
DLC_RESOLVED	Send ICANREACH_ex Optionally update cache	RESET

RECEIVED_EX is entered when the DLSw has received a CANUREACH_ex from a remote DLSw and has issued a DLC_RESOLVE to locate the MAC address. This state is exited when the DLC_RESOLVED response is received, or after an implementation defined timeout.

If the target MAC is located, the DLSw must reply to the first received CANUREACH_ex that caused the move to this state. If additional CANUREACH_ex messages are received in this state from other remote DLSw partners, the DLSw may optionally reply to these messages too but it is not required to do so.

An implementation may choose whether to handle explorer frame crossover either by using entirely separate FSM instances and simply allowing both ends to issue TEST frames, or by detecting such a reverse DLC_RESOLVE_C event here and issuing an ICANREACH_ex message and DLC_RESOLVE_R action.

5.4.2. NETBIOS_NQ/NR Explorer FSM

The FSM described below is used to handle explorer frames routed by NetBIOS names There is one instance of this FSM for each unique combination of Source Name, Destination Name, Data 2 field and Response Correlator.

State Name	Description
RESET	The initial state.
SENT_EX	Local DLSw has issued an explorer message
RECEIVED_EX	Local DLSw has received an explorer message
SENT_REC_EX	An explorer frame has been both sent and received for the same (potential) NetBIOS circuit.

5.4.2.1. RESET State

Event	Action(s)	Next State
Receive NETBIOS_NQ_ex	DLC_DGRM(NAME_QUERY) Optionally update cache.	RECEIVED_EX
Receive NETBIOS_NR_ex	Optionally update cache	RESET
DLC_DGRM (NAME_QUERY)	Send NETBIOS_NQ_ex	SENT_EX

The RESET state is the initial state for the NETBIOS_NQ/NR explorer FSM. It is exited when the DLC receives either a NETBIOS_NQ_ex or a DLC_DGRM containing a NetBIOS NAME_QUERY frame. If a NETBIOS_NQ_ex message is received, the NAME_QUERY is propagated to the DLC and this FSM moves to state RECEIVED_EX. If a NetBIOS NAME_QUERY frame is received, the NETBIOS_NQ_ex is propagated either to the appropriate DLSw partners (see below), and this FSM moves to state SENT_EX.

Unlike SNA traffic where the CANUREACH_ex/ICANREACH_ex exchange can be omitted if the MAC location is already cached, NETBIOS_NQ_ex/NETBIOS_NR_ex frames must always be issued during NetBIOS session setup in order that the NetBIOS session numbers are exchanged correctly between the DLC end stations. If the location of a NetBIOS name is known from cached data, the NETBIOS_NQ_ex need only be issued to the cached DLSw partners. Otherwise the NETBIOS_NQ_ex should be issued to all partners that support NetBIOS.

5.4.2.2. SENT_EX State

Event	Action(s)	Next State
Receive NETBIOS_NQ_ex	DLC_DGRM(NAME_QUERY) Optionally update cache	SENT_REC_EX
Receive NETBIOS_NR_ex	DLC_DGRM(NAME_RECOG) Optionally update cache	RESET
DLC_DGRM (NAME_QUERY) (different local session number than existing searches)	Send NETBIOS_NQ_ex Optionally update cache	SENT_EX

SENT_EX is entered when the local DLSw issues a NETBIOS_NQ_ex to its remote DLSw partners. This state is exited when a NETBIOS_NR_ex is received from a remote DLSw, or if a matching NETBIOS_NQ_ex is received from a remote DLSw (i.e. a NETBIOS_NQ_ex crossover case). If the local NetBIOS end station issues a NAME_QUERY with a different session number from any previous NAME_QUERY for this search, the NAME_QUERY is propagated to the DLSw partners to ensure that the exchange of NetBIOS session numbers is handled correctly.

5.4.2.3. RECEIVED_EX State

Event	Action(s)	Next State
Receive NETBIOS_NQ_ex	DLC_DGRM(NAME_QUERY) Optionally update cache	RECEIVED_EX
Receive NETBIOS_NR_ex		RECEIVED_EX
DLC_DGRM (NAME_QUERY)	Send NETBIOS_NQ_ex Optionally update cache	SENT_REC_EX
DLC_DGRM (NAME_RECOG)	Send NETBIOS_NR_ex Optionally update cache	RESET

RECEIVED_EX is entered when the local DLSw receives a NETBIOS_NQ_ex message from a remote DLSw. This state is exited when a NAME_RECOGNIZED NetBIOS frame is received from the DLC, completing the query, or when a matching NAME_QUERY is received from DLC (i.e. NAME_QUERY crossover).

5.4.2.4. SENT_REC_EX State

Event	Action(s)	Next State
Receive NETBIOS_NQ_ex	DLC_DGRM(NAME_QUERY) Optionally update cache	SENT_REC_EX
Receive NETBIOS_NR_ex	DLC_DGRM(NAME_RECOG) Optionally update cache	RECEIVED_EX
DLC_DGRM (NAME_QUERY) (different local session number than existing searches)	Send NETBIOS_NQ_ex Optionally update cache	SENT_REC_EX
DLC_DGRM (NAME_RECOG)	Send NETBIOS_NR_ex Optionally update cache	SENT_EX

This state is required if an implementation wishes to manage NQ/NR crossover cases from a single FSM instance by detecting 'opposite' NAME_QUERY attempts between the same two NetBIOS names. If separate FSM instances are used instead, this state is not required and the transitions to it from other states can be removed.

SENT_RCV_EX is exited when the NAME_QUERY search in either direction is resolved. If the local NetBIOS end station issues a NAME_QUERY with a different session number from any previous NAME_QUERY it has issued for this search, the NAME_QUERY is propagated to the DLSw partners to ensure that the exchange of NetBIOS session numbers is correctly handled.

5.4.2.5. NetBIOS Session Numbers

NetBIOS NAME_QUERY and NAME_RECOGNIZED frames exchange NetBIOS session numbers between the end stations. For correct NetBIOS operation over DLSw, it is important that all SSP NETBIOS_NQ_ex frames received by a DLSw cause NetBIOS NAME_QUERY frames to flow on the LAN with the new session number from the NETBIOS_NQ_ex. These frames cannot be replied to from a cache of locally available NetBIOS names in the same way that MAC addresses and CANUREACH_ex messages can be handled.

Also, NAME_QUERY messages are normally retried several times on the LAN. The generation and absorption of such frames is outside the scope of the FSM defined above.

6. Protocol Flow Diagrams

The Switch-to-Switch Protocol is used to setup and take down circuits between a pair of Data Link Switches. Once a circuit is established, the end stations on the local networks can employ LLC Type 1 (connectionless UI frames) protocols end-to-end. In addition, the end systems can establish an end-to-end connection for support of LLC Type 2 (connection oriented I frames) protocols (Type 2 I frames go end-to-end, supervisory frames are handled locally).

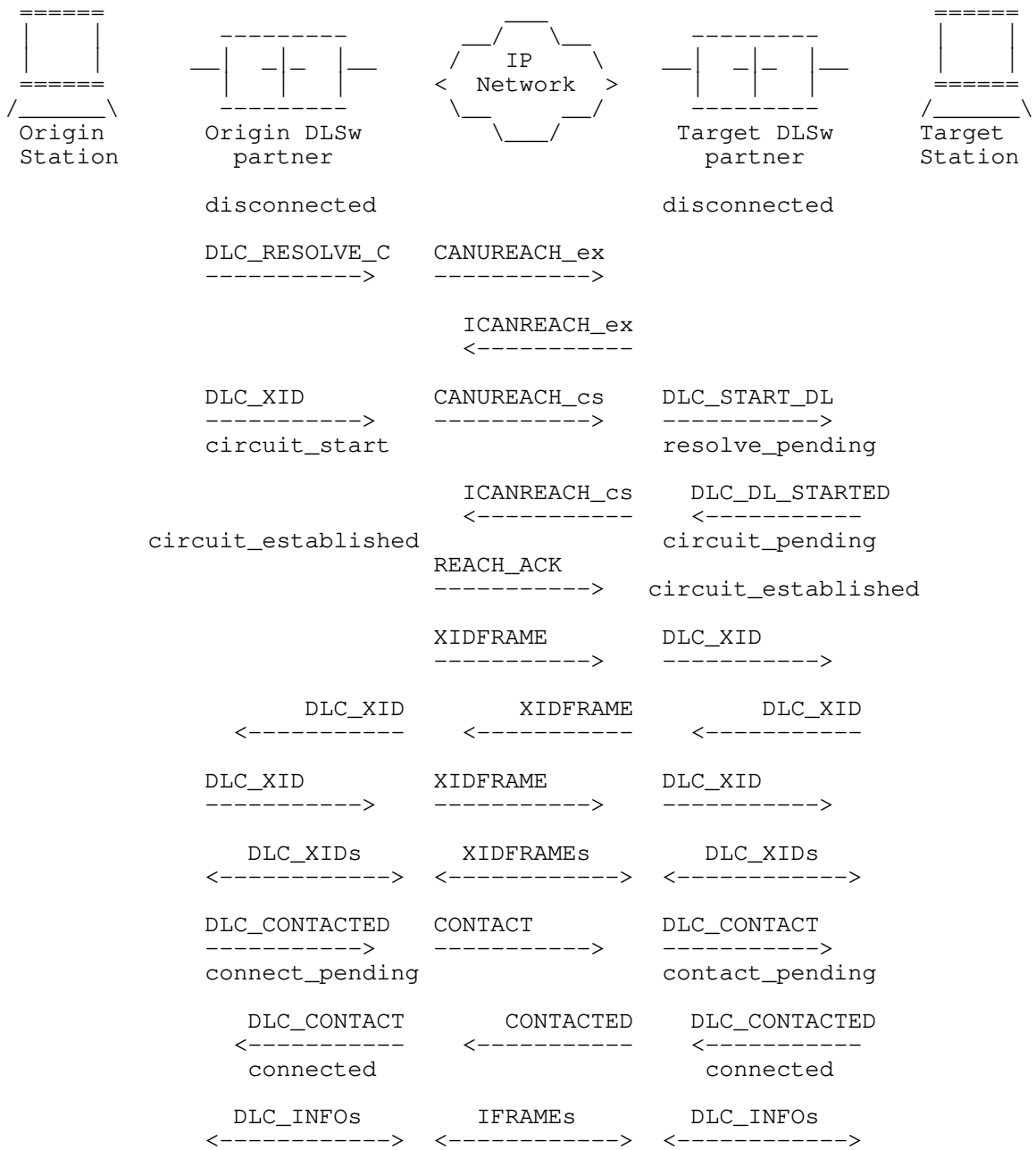
The term, Data Link, is used in this document to refer to both a "logical data link" when supporting Type 1 LLC services, and a "data link connection" when supporting Type 2 LLC services. In both cases, the Data Link is identified by the Data Link ID defined in section 3.2.

NOTE: THIS SECTION CONTAINS EXAMPLES ONLY. IT CANNOT AND DOES NOT SHOW ALL POSSIBLE VARIATIONS AND OPTIONS ON PROTOCOL FLOWS FOR SNA/SDLC, SSP, AND LLC PROTOCOLS.

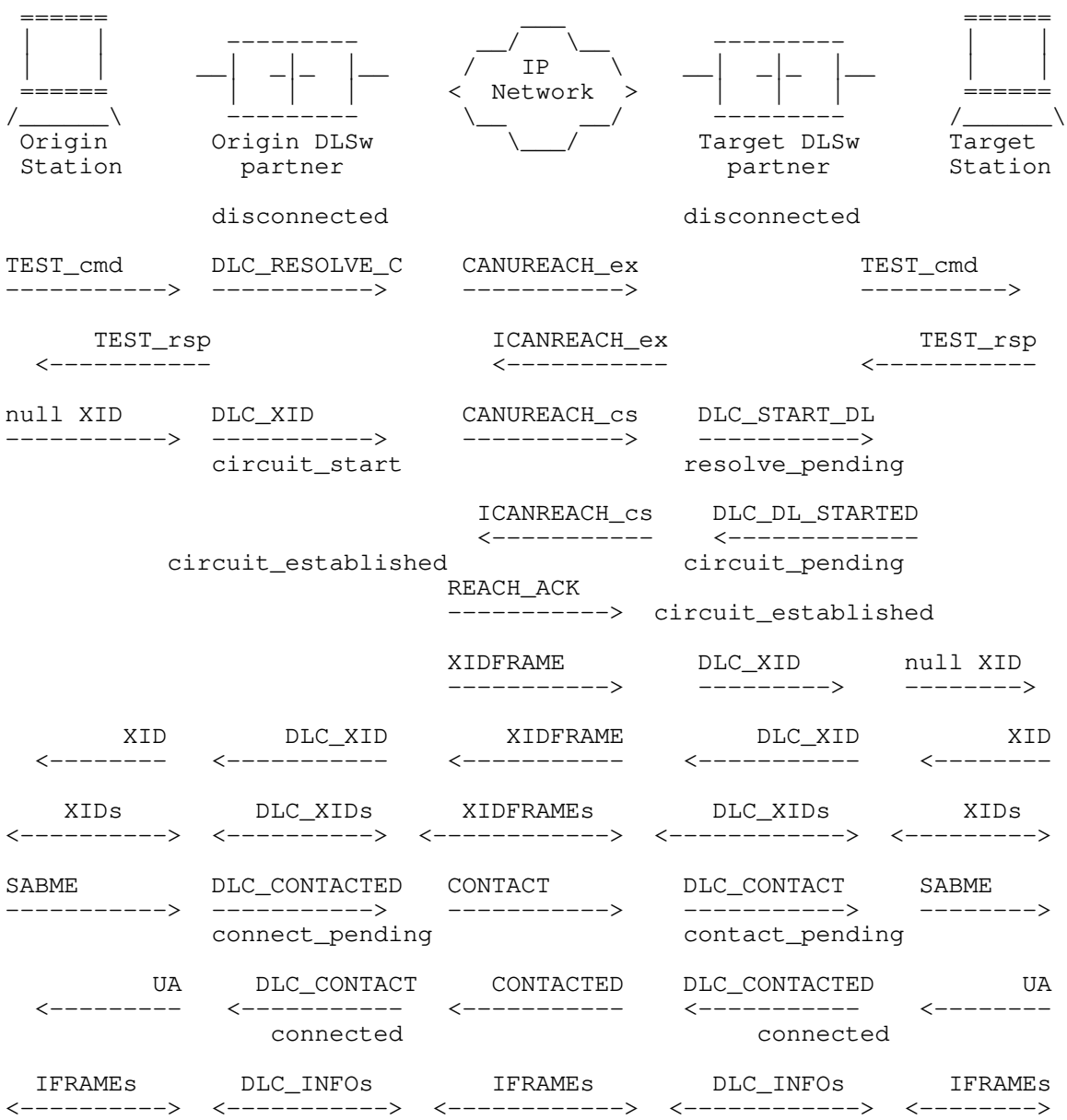
6.1. Connect Protocols

The two basic startup flows from a pure FSM perspective are shown below. The first flow is a startup involving XIDs and the second is one without XIDs.

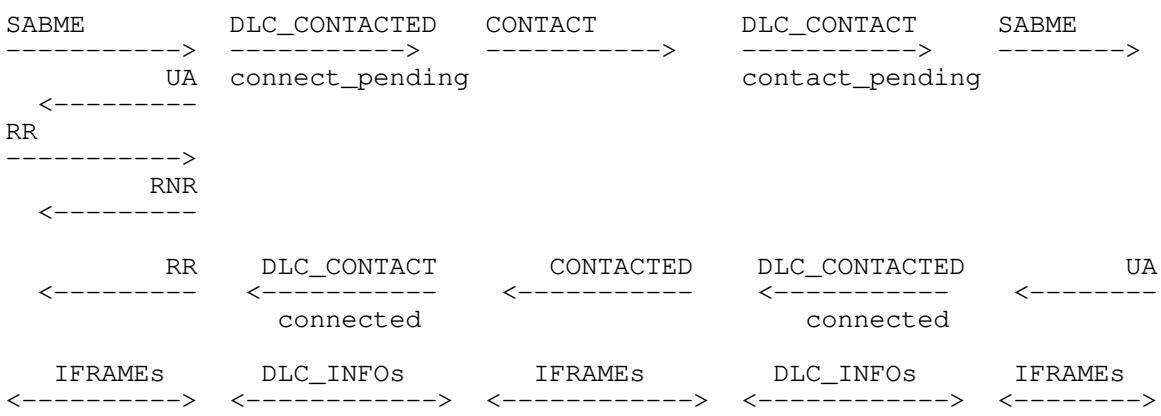
Flow #1 - DLSw Startup With XIDs



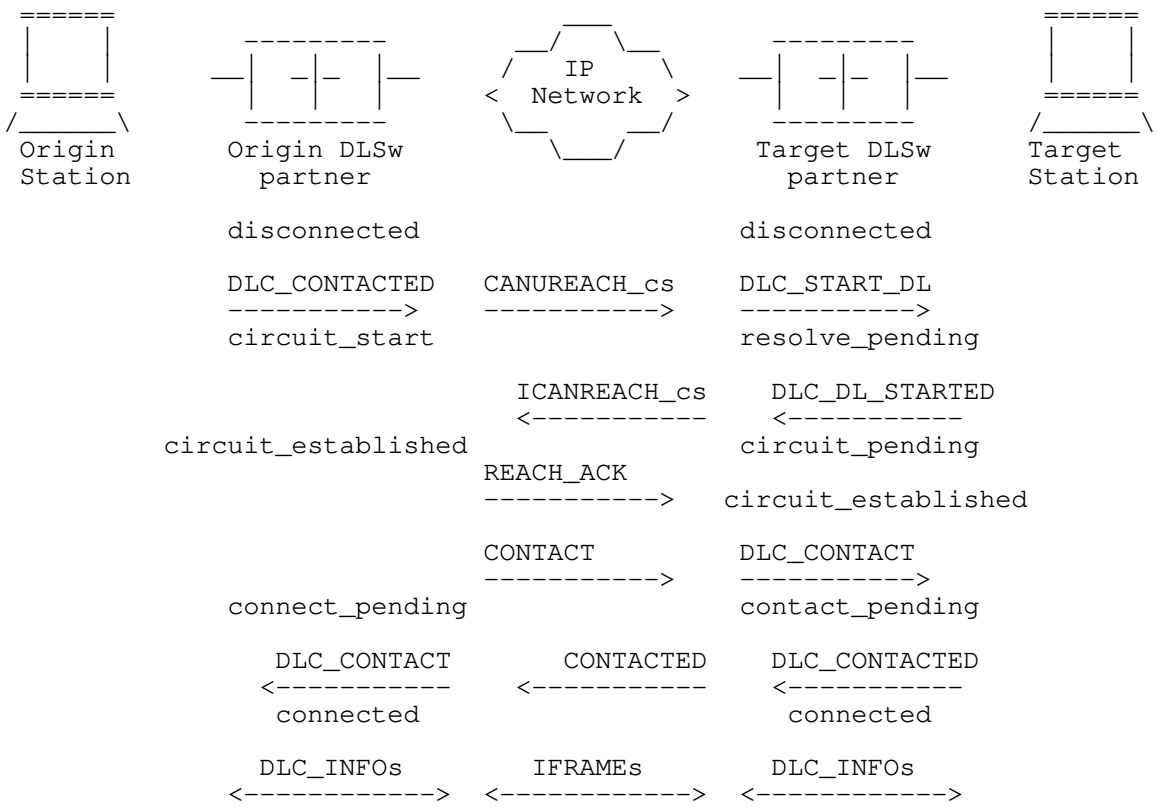
Mapping LAN events to the DLC events and actions on Flow #1 produces the following flows shown below:



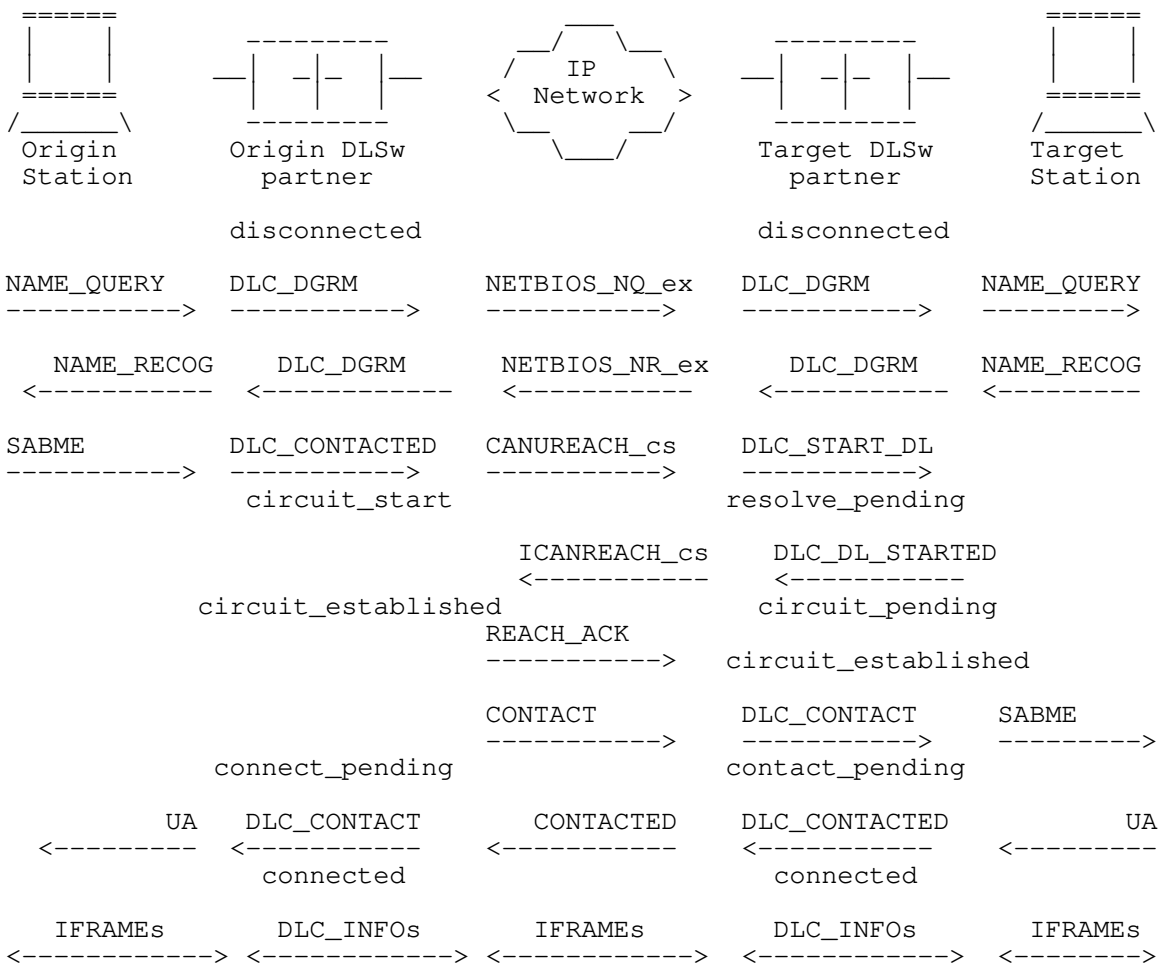
Those implementations that prefer to respond to the SABME immediately could use the same events to do that:



Flow #2 - DLSw Startup Without XIDs (circuit setup)

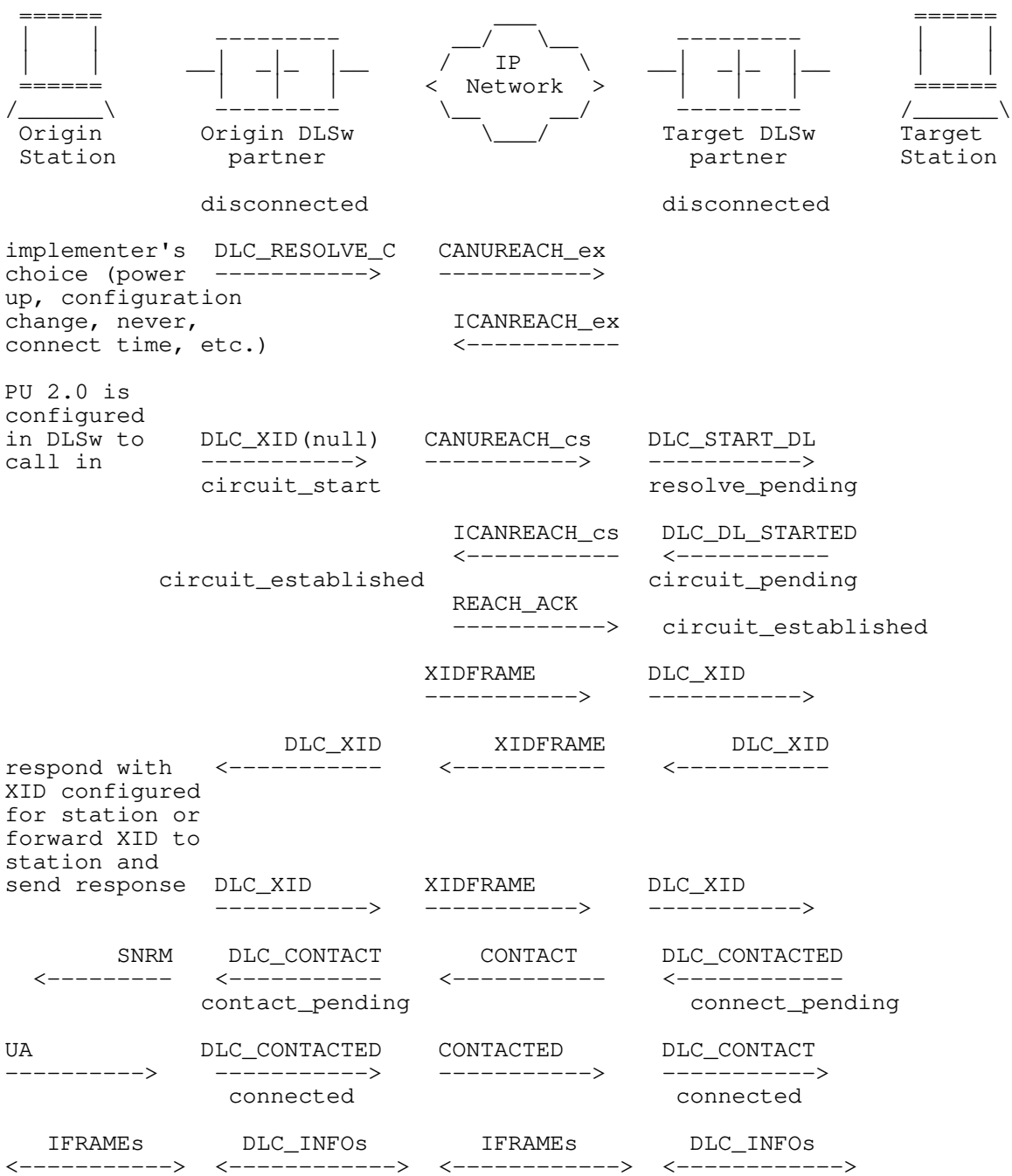


Mapping LAN events to the DLC events and actions on Flow #2 (and adding a NETBIOS_NQ and NETBIOS_NR_ex) produces:

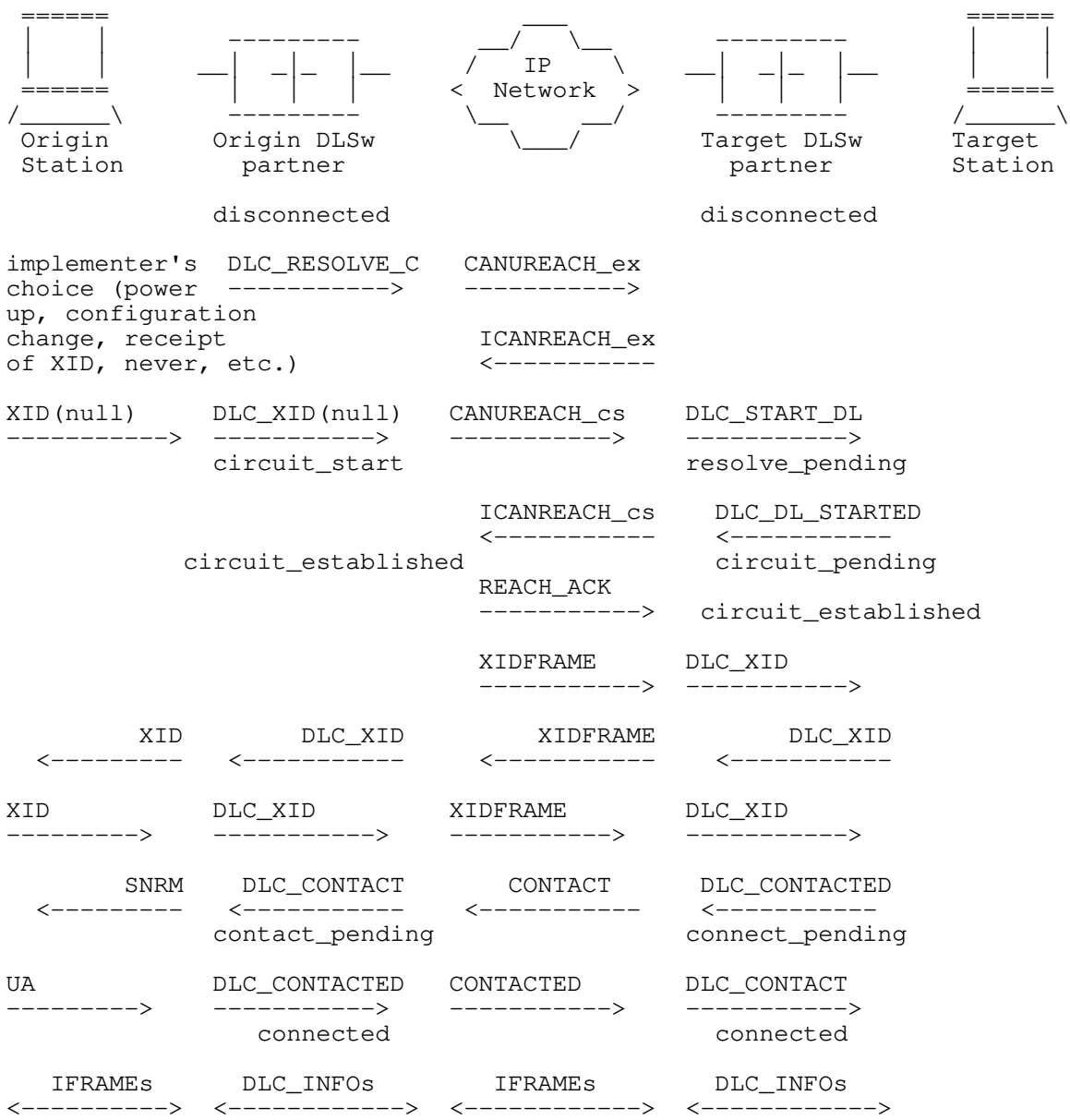


In keeping with a paradigm of 'DLSw is a big 802.2 LAN', all other DLC types (SDLC for now, QLLC, channel, or whatever in the future) would be handled by a 'DLC transformation layer' that would transform the specific protocol's events into the appropriate DLSw DLC events and DLSw DLC actions into the appropriate protocol actions. The XIDs that flow in the SSP XIDFRAME should stay 802.2ish (i.e. ABM bit set) and leave it up to the DLC transformation layer to suit the XID to its particular DLC type.

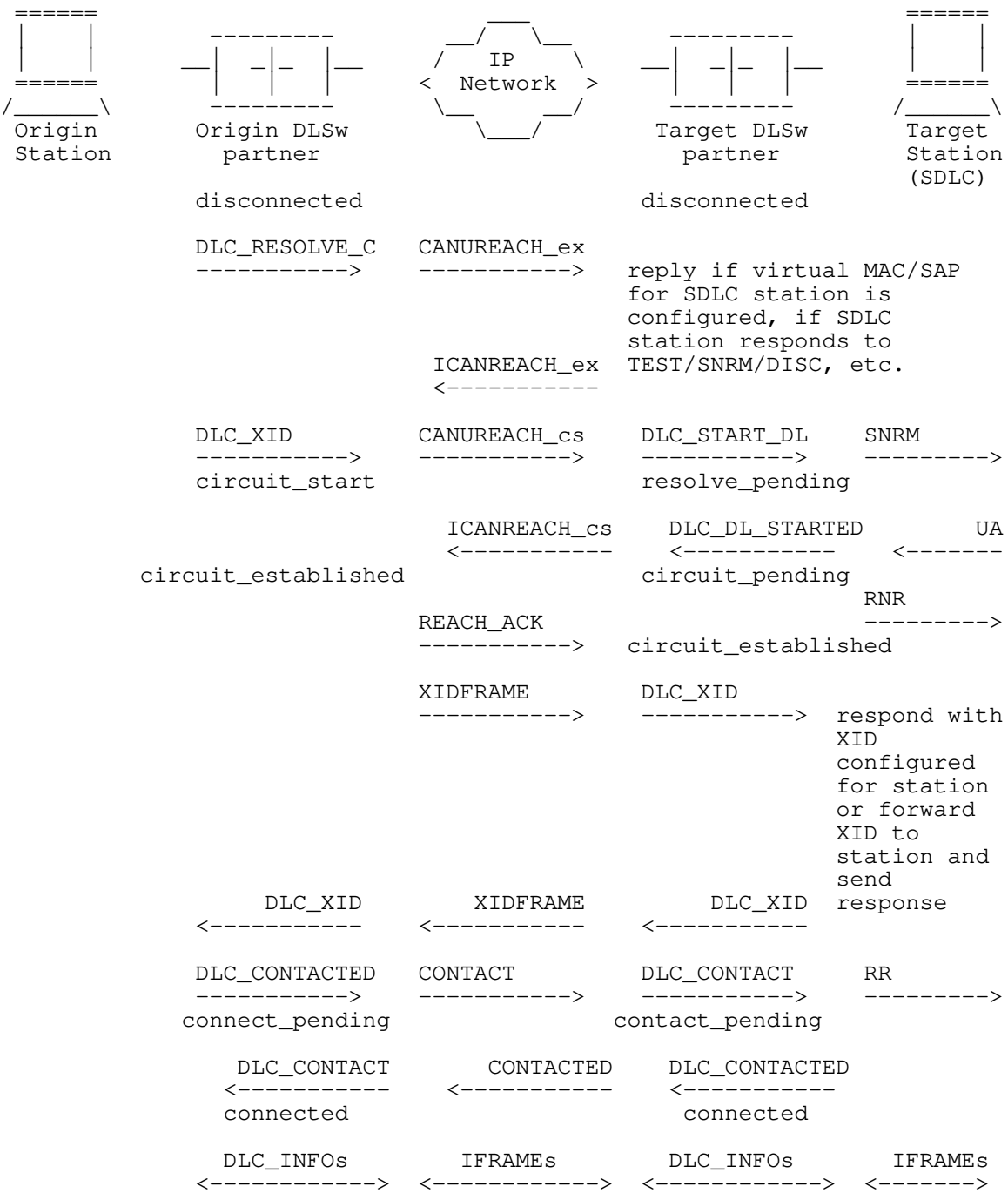
Here is an example of a leased SDLC PU 2.0 device as the origin station. It should use Flow #2 since it is not known if the other side is a LAN, a switched line or a leased line.



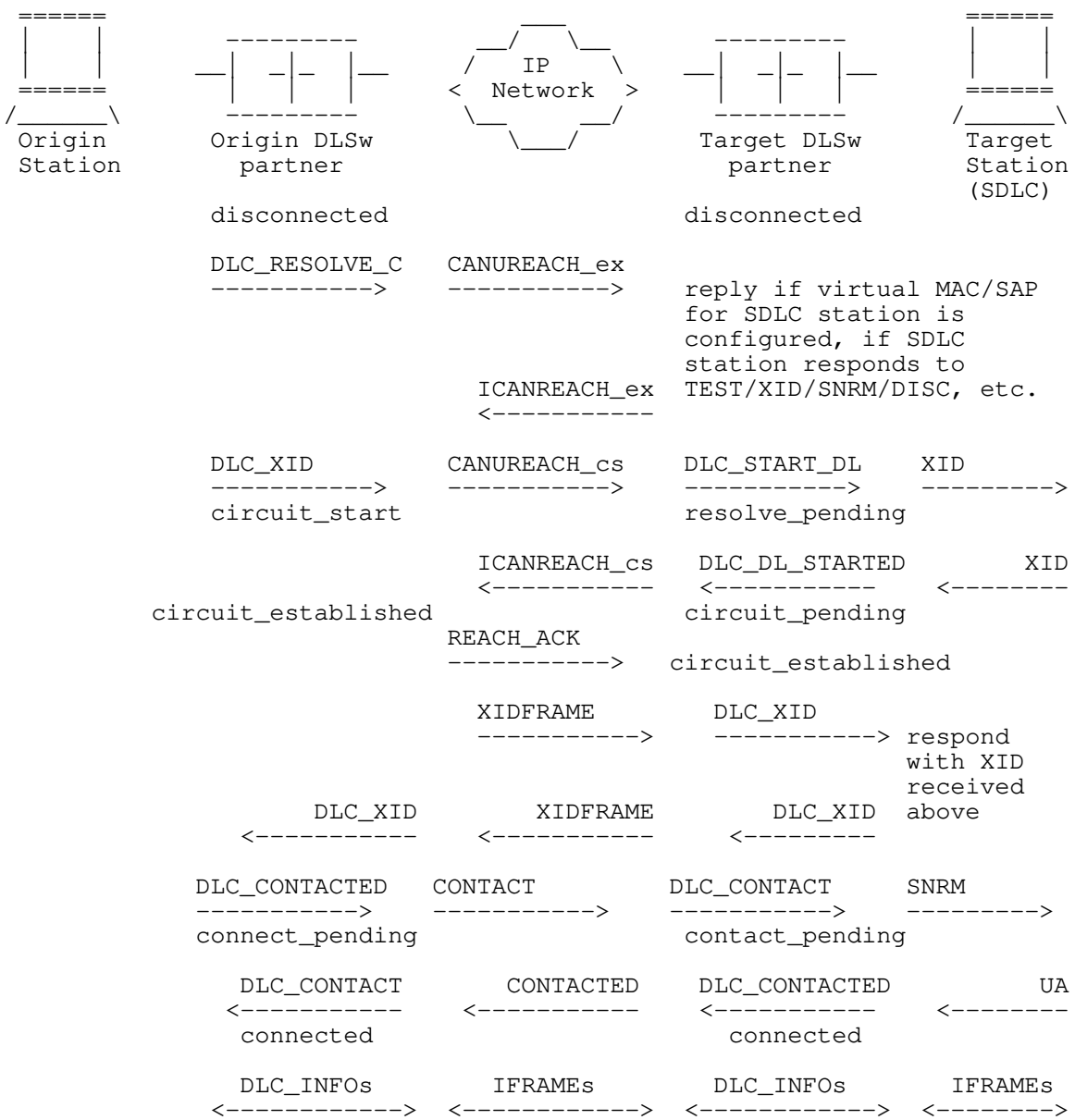
Here is an example of a switched SDLC PU 2.0 device as the origin station.



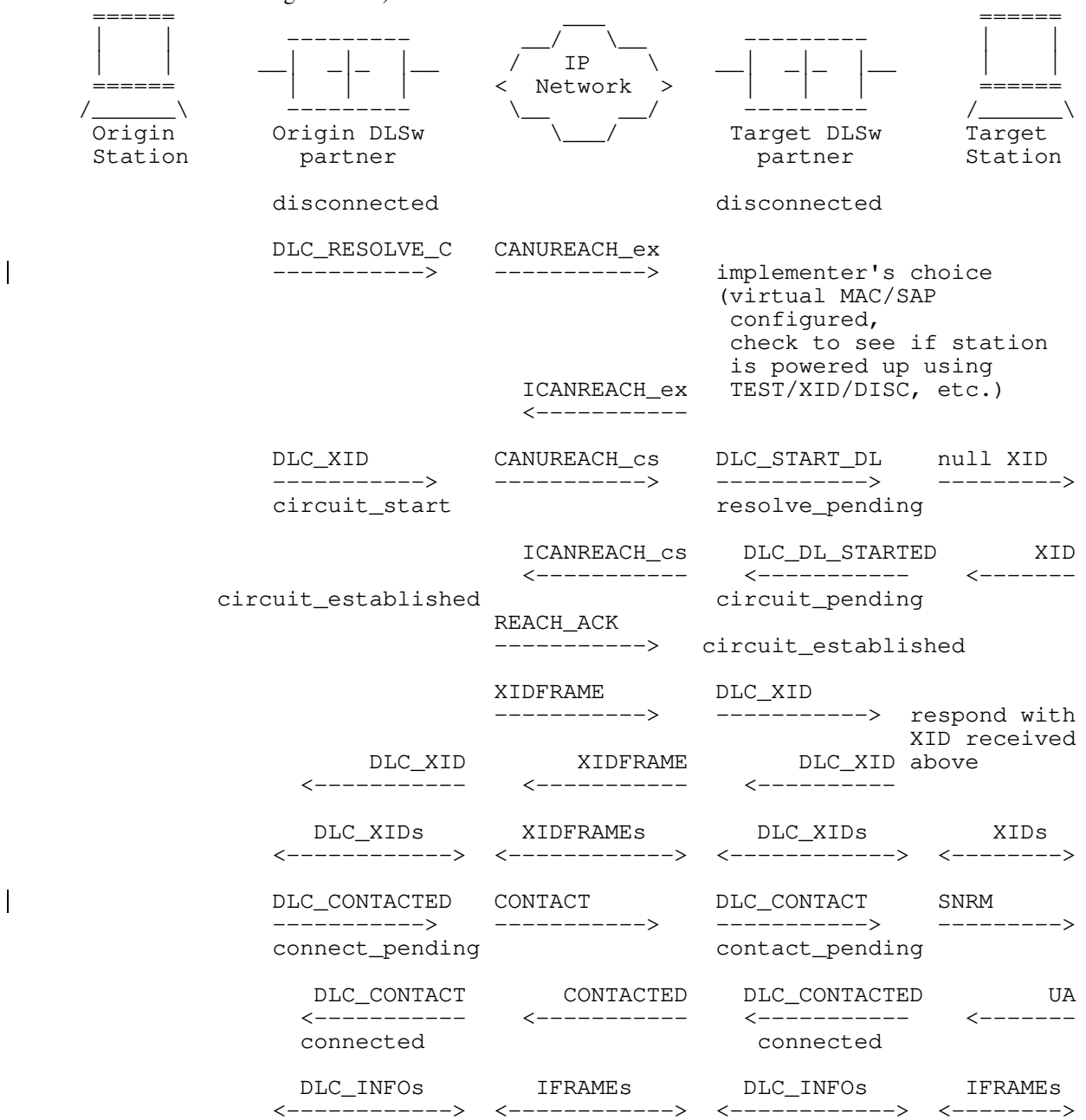
Here is an example of a leased SDLC PU 2.0 device as the target station.



Here is an example of a switched SDLC PU 2.0 device as the target station.



Here is an example of an SDLC T2.1 device as the target station. (SDLC T2.1 origin station would look just like the LAN T2.1 origin station)



6.2. Link Restart Protocols

The following figure depicts the protocol flows that result from restarting the end-to-end connection. This causes the Data Link Switches to terminate the existing connection and to enter the Circuit Established state awaiting the start of a new connection.

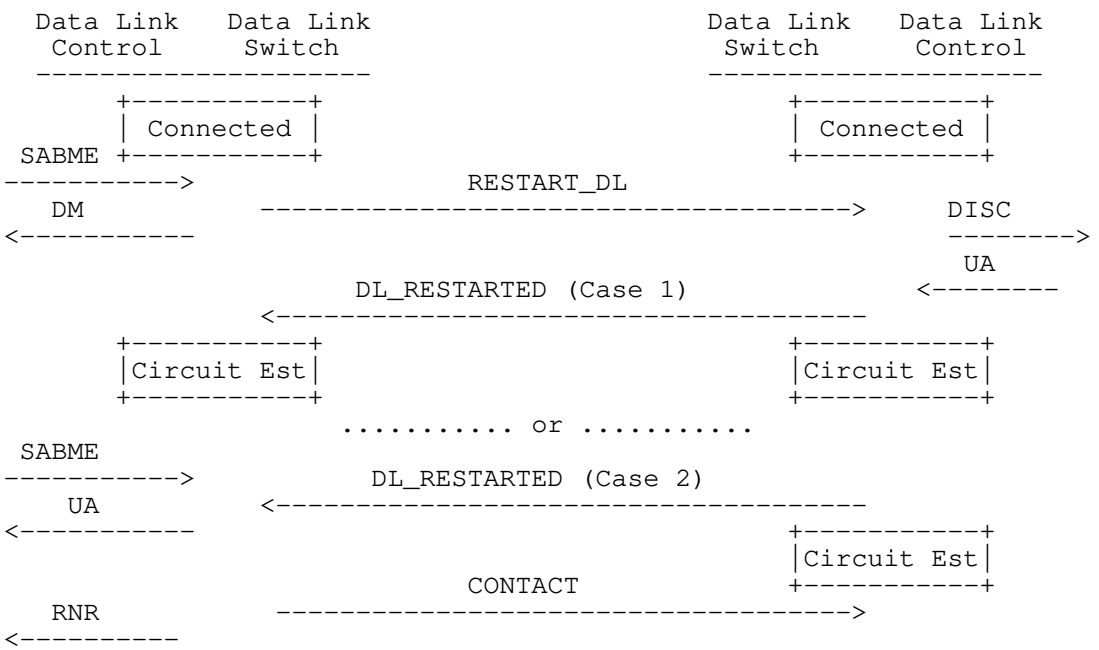


Figure 5. DLSw Link Restart Message Protocols

Upon receipt of a SABME command from the origin station, the origin DLSw will send a RESTART_DL message to the target DLSw. A DM response is also returned to the origin station and the data link is restarted.

Upon receipt of the RESTART_DL message, the target DLSw will issue a DISC command to the target station. The target station is expected to return a UA response. The target DLSw will then restart its data link and send an DL_RESTARTED message back to the origin DLSw. During this exchange of messages, both Data Link Switches change states from Connected state to Circuit Established state.

If the origin station now resends the SABME command, the origin DLSw will send a CONTACT message to the target DLSw. If the SABME command is received prior to the receipt of the DL_RESTARTED message (case 2 in the figure), the CONNECT message is delayed until the DL_RESTARTED message is received. The resulting protocol flows at this point parallel those given above for the connect sequence.

6.3. Disconnect Protocols

The following figure depicts the protocol flows that result from the end system terminating an existing connection. Not only is the connection terminated, but the circuit between the Data Link Switches is taken down.

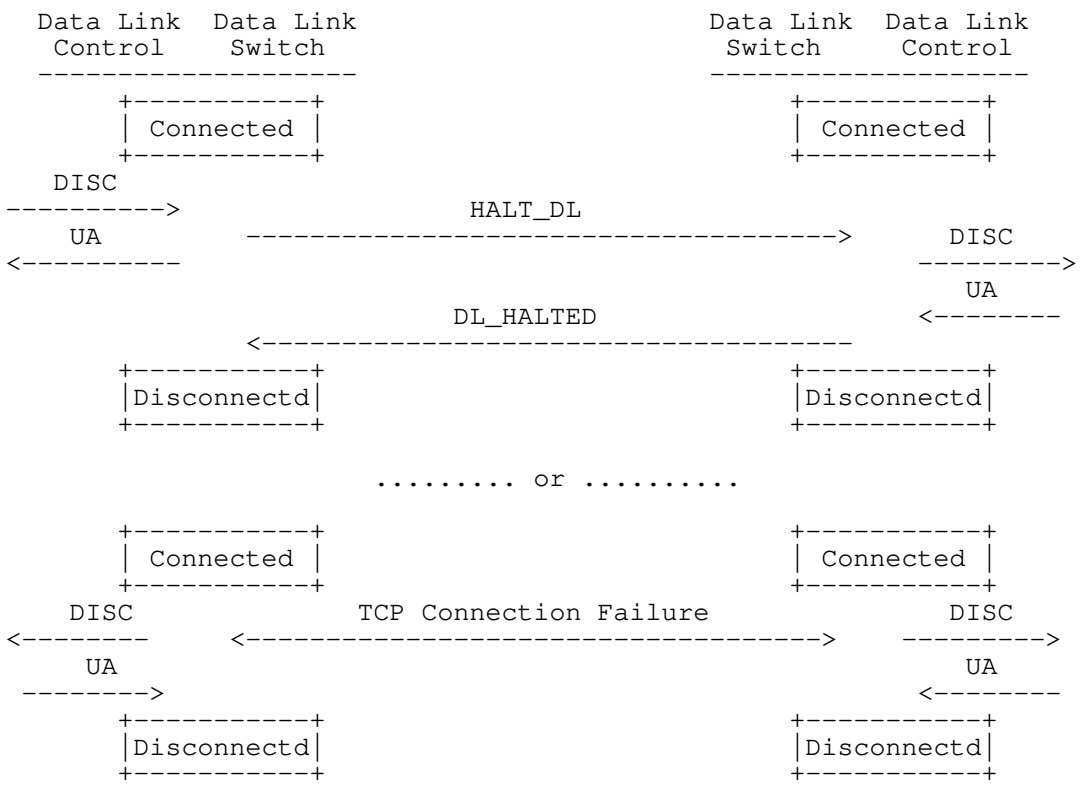


Figure 6. DLSw Disconnect Message Protocols

Upon receipt of a DISC command from the origin station, the origin DLSw will reply with a UA response and issue a HALT_DL message to the target DLSw. Upon receipt of the HALT_DL message, the target DLSw will send a DISC command to the target station. The target station will then respond with a UA response, causing the target DLSw to return a DL_HALTED message to the origin DLSw. During this exchange of messages, both Data Link Switches change states from the Connected state to the Disconnected state.

If the TCP connection between two Data Link Switches fails, all connections that are currently multiplexed on the failed TCP connection will be taken down. This implies that both Data Link Switches will send DISC commands to all the local systems that are associated with the failed connections. Upon sending the DISC command, the Data Link Switch will enter the DISCONNECTED state for each circuit.

7. Capabilities Exchange Formats/Protocol

The Data Link Switching Capabilities Exchange is a special DLSw Switch-to-Switch control message that describes the capabilities of the sending data link switch. This control message is sent after the switch-to-switch connection is established and optionally during run time if certain operational parameters have changed and need to be communicated to the partner switch.

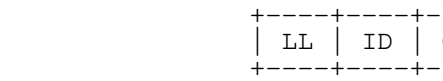
The actual contents of the Capabilities Exchange is in the data field following the SSP message header. The Capabilities Exchange itself is formatted as a single General Data Stream (GDS) Variable with multiple type "LT" structured subfields.

The SSP Message Header has the following fields set for the Capabilities Exchange:

Offset	Field	Value
0x00	Version Number	0x31
0x01	Header Length	0x48 (decimal 72)
0x02	Message Length	same as LL in GDS Variable
0x14	Message Type	0x20 (CAP_EXCHANGE)
0x16	Protocol Id	0x42
0x17	Header Number	0x01
0x23	Message Type	0x20 (CAP_EXCHANGE)
0x38	Direction	0x01 for CapEx request 0x02 for CapEx response

Other fields in the SSP header are not referenced and should be set to zero.

The DLSw Capabilities Exchange Request has the following overall format:



0-1	Length, in binary, of the DLSw Capabilities Exchange Request GDS Variable. The value of LL is the sum of the length of all fields in the GDS Variable (i.e. length of LL + length of ID + length of Control Vectors).
2-3	GDS Id: 0x1520
4-n	Control Vectors consisting of type LT structured subfields (i.e. the DLSw Capabilities Exchange Structured Subfields)

Type LT structured subfields consist of a 1-byte length field (the "L"), a 1-byte type field (the "T") and n-bytes of data. The length field includes itself as well as the structured subfield. The structured subfield consists of the type field and data so the length is n + 2. This imposes a length restriction of 253 bytes on all data contained in a structured subfield.

7.1. Control Vector Id Range

Control Vector identifiers (i.e. Type) in the range of 0x80 through 0xCF are reserved for use by the Data Link Switching standard.

Control Vector identifiers (i.e. Type) in the range of 0xD0 through 0xFD are used for vendor-specific purposes.

Currently defined vectors are:

Vector Description	Hex Value
Vendor Id Control Vector	0x81
DLSw Version Control Vector	0x82
Initial Pacing Window Control Vector	0x83
Version String Control Vector	0x84
Mac Address Exclusivity Control Vector	0x85
Supported SAP List Control Vector	0x86
TCP Connections Control Vector	0x87
NetBIOS Name Exclusivity Control Vector	0x88
MAC Address List Control Vector	0x89
NetBIOS Name List Control Vector	0x8A
Vendor Context Control Vector	0x8B
Reserved for future use	0x8C - 0xCF
Vendor Specific	0xD0 - 0xFD

7.2. Control Vector Order and Continuity

Since their contents can greatly affect the parsing of the Capabilities Exchange GDS Variable, the required control vectors must occur first and appear in the following order: Vendor Id, DLSw Version Number, Initial Pacing Window, Supported SAP List. The remainder of the Control Vectors can occur in any order.

Control Vectors that can be repeated within the same message (e.g. MAC Address List Control Vector and NetBIOS Name List Control Vector) are not necessarily adjacent. It is advisable, but not required, to have the Exclusivity Control Vector occur prior to either of the above two vectors so that the use of the individual MAC addresses or NetBIOS names will be known prior to parsing them.

Both the Vendor Context and Vendor Specific control vectors can be repeated. If there are multiple instances of the Vendor Context control vector, the specified context remains in effect for all Vendor Specific control vectors until the next Vendor Context control vector is encountered in the Capabilities Exchange.

7.3. Run-Time Capabilities Exchange

Capabilities exchange always occurs when the SSP connection is started between two DLSw switches. Capabilities Exchange can also occur at run-time, typically when a configuration change is made.

Support for run-time Capabilities Exchange is optional. If a node does not support receiving/using Run-Time Capabilities Exchange and receives one, it should discard it quietly (not send back a negative response). If a node supports receipt of run-time capabilities, it should send a positive or negative response as appropriate. The receiver of a negative response to a run-time capabilities message is not required to take down its TCP connections with the offended partner.

Run-time Capabilities Exchange can consist of one or more of the following control vectors. Note that the control vectors required at start-up are not present in a run-time Capabilities Exchange.

1. MAC Address Exclusivity CV,
2. NetBIOS Name Exclusivity CV,
3. MAC Address List CV,
4. NetBIOS Name List CV,
5. Supported SAP List CV,
6. Vendor Context CV,
7. Vendor Specific CVs

A run-time capabilities exchange is a replacement operation. As such, all pertinent MAC addresses and NetBIOS names must be specified in the run-time exchange. In addition, run-time changes in capabilities will not effect existing link station circuits.

7.4. Capabilities Exchange Filtering Responsibilities

Recipients of the SAP, MAC, and NetBIOS lists are not required to actually use them to filter traffic, etc., either initially or at run-time.

7.5. DLSw Capabilities Exchange Structured Subfields

The Capabilities Exchange Subfields are listed in the table below and are described in the following sections: The following table summarizes the DLSw Capabilities Exchange Control Vectors.

ID	Required @ Startup	Length	Repeatable*	Allowed @ Runtime	Order	Content
0x81	Y	0x05	N	N	1	Vendor ID
0x82	Y	0x04	N	N	2	DLSw Version
0x83	Y	0x04	N	N	3	Initial pacing window
0x84	N	>=0x02	N	N	5+	Version String
0x85	N	0x03	N	Y	5+	MAC Address Exclusivity
0x86	Y	0x12	N	Y	4	Supported SAP List
0x87	N	0x03	N	N	5+	TCP Connections
0x88	N	0x03	N	Y	5+	NetBIOS Name Exclusivity
0x89	N	0x0E	Y	Y	5+	MAC Address List
0x8A	N	<=0x13	Y	Y	5+	NetBIOS Name List
0x8B	N	0x05	Y	Y	5+	Vendor Context
0xD0	N	varies	Y	Y	5+	Vendor Specific

*Note: "Repeatable" means a Control Vector is repeatable within a single message.

7.5.1. Vendor Id (0x81) Control Vector

The Vendor Id control vector identifies the manufacturer's IEEE assigned Organizationally Unique Identifier (OUI) of the Data Link Switch sending the DLSw Capabilities Exchange. The OUI is sent in non-canonical (Token-Ring) format. This control vector is required and must be the first control vector.

Offset	Length	Value	Contents
0	1	0x05	Length of the Vendor Id structured subfield
1	1	0x81	key = 0x81 that identifies this as the Vendor Id structured subfield
2-4	3		the 3-byte Organizationally Unique Identifier (OUI) for the vendor (non-canonical format)

7.5.2. DLSw Version (0x82) Control Vector

The DLSw Version control vector identifies the particular version of the DLSw standard supported by the sending Data Link Switch. This control vector is required and must follow the Vendor Id Control Vector.

Offset	Length	Value	Contents
0	1	0x04	Length of the Version String structured subfield
1	1	0x82	key = 0x82 that identifies this as the DLSw Version structured subfield
2	1		the hexadecimal value representing the DLSw standard Version number of the sending Data Link Switch. 0x01 (indicates version 1 - closed pages)
3	1		the hexadecimal value representing the DLSw standard Release number of the sending Data Link Switch. 0x00 (indicates release 0)

7.5.3. Initial Pacing Window (0x83) Control Vector

The Initial Pacing Window control vector specifies the initial value of the receive pacing window size for the sending Data Link Switch. This control vector is required and must follow the DLSw Version Control Vector.

Offset	Length	Value	Contents
0	1	0x04	Length of the Initial Pacing Window structured subfield
1	1	0x83	key = 0x83 that identifies this as the Initial Pacing Window structured subfield
2-3	2		the pacing window size, specified in byte normal form..

7.5.4. Version String (0x84) Control Vector

The Version String control vector identifies the particular version number of the sending Data Link Switch. The format of the actual version string is vendor-defined. This control vector is optional.

Offset	Length	Value	Contents
0	1	0xn	Length of the Version String structured subfield
1	1	0x84	key = 0x84 that identifies this as the Version String structured subfield
2-n	n-2		the ASCII string that identifies the software version for the sending DLSw.

7.5.5. MAC Address Exclusivity (0x85) Control Vector

The MAC Address Exclusivity control vector identifies how the MAC Address List control vector data is to be interpreted. Specifically, this control vector identifies whether the MAC addresses in the MAC Address List control vectors are the only ones accessible via the sending Data Link Switch.

If a MAC Address List control vector is specified and the MAC Address Exclusivity control vector is missing, then the MAC addresses are not assumed to be the only ones accessible via this switch.

A node may specify that it supports no local MAC addresses by including in its capabilities the MAC Address List Exclusivity CV (with byte 2 == 0x01), and not including any instances of the MAC Address List CV.

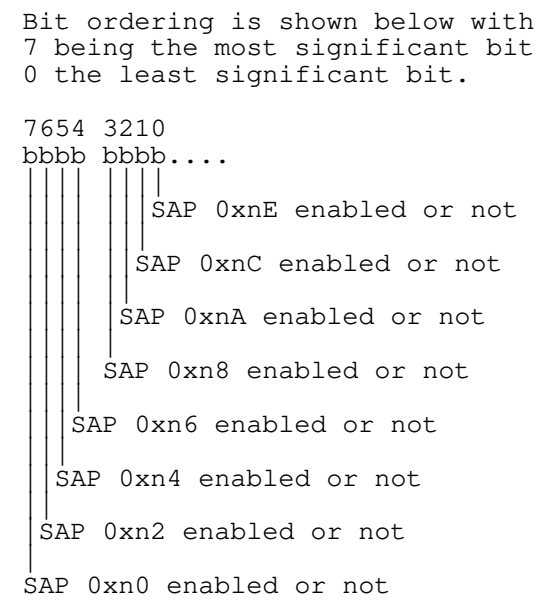
Offset	Length	Value	Contents
0	1	0x03	Length of the Exclusivity structured subfield
1	1	0x85	key = 0x85 that identifies this as the MAC address Exclusivity structured subfield
2	1		an indicator of the relationship of the MAC addresses to the sending Data Link Switch. 0x00 the MAC addresses specified in this Capabilities Exchange can be accessed via this switch but are not the exclusive set (i.e. other entities are accessible in addition to the ones specified) 0x01 the MAC addresses specified in this Capabilities Exchange are the only ones accessible via this switch.

7.5.6. SAP List Support (0x86) Control Vector

The SAP List Support control vector identifies support for Logical Link Control SAPs (DSAPs and SSAPs) by the sending Data Link Switch. This is used by the DLSw that sent the SAP List Support control vector to indicate which SAPs can be used to support SNA and optionally NetBIOS traffic. This may be used by the DLSw that receives the SAP list to filter explorer traffic (TEST, XID, or NetBIOS UI frames) from the DLSw state machine. For SNA, a DLSw should set bits for all SAP values (SSAP or DSAP) that may be used for SNA traffic. For NetBIOS support, the bit for SAP 0xF0 should be set (if not supported then the same bit should be cleared).

Each bit in the SAP control vector data field represents a SAP as defined below. This vector is required and must follow the Initial Pacing Window Control Vector.

Offset	Length	Value	Contents
0	1	0x12	Length of the Supported SAP List structured subfield
1	1	0x86	key = 0x86 that identifies this as the Supported SAP List structured subfield
2-17	16		the 16-byte bit vector describing all even numbered SAPs enabled. Each Bit within the 16 byte bit vector will indicate whether an even numbered SAP is enabled (b'1') or disabled (b'0'). Each Byte within the 16 byte bit vector will be numbered from 0 - F. (Most significant byte first). Byte 0 1 2 3 ... F XX XX XX XX ... XX The bits in each byte indicate whether an even numbered SAP is enabled (b'1') or disabled (b'0'). (Most significant bit first) Bits 7 6 5 4 ... 0 SAP 0 2 4 6 ... E By combining the byte label with the enabled bits, all supported SAPs can be determined. In the following diagram, 'n' would equal 0 through F depending on which byte was being interpreted.



An example of using all User Definable SAPs of 0x04 to 0xEC for SNA Data Link Switching and SAP 0xF0 for NetBIOS Data Link Switching would be as follows:

Offset	SAPs	Binary	Hex
0	4,8,C	0010 1010	0x2A
1	10,14,18,1C	1010 1010	0xAA
2	20,24,28,2C	1010 1010	0xAA
3	30,34,38,3C	1010 1010	0xAA
4	40,44,48,4C	1010 1010	0xAA
5	50,54,58,5C	1010 1010	0xAA
6	60,64,68,6C	1010 1010	0xAA
7	70,74,78,7C	1010 1010	0xAA
8	80,84,88,8C	1010 1010	0xAA
9	90,94,98,9C	1010 1010	0xAA
A	A0,A4,A8,AC	1010 1010	0xAA
B	B0,B4,B8,BC	1010 1010	0xAA
C	C0,C4,C8,CC	1010 1010	0xAA
D	D0,D4,D8,DC	1010 1010	0xAA
E	E0,E4,E8,EC	1010 1010	0xAA
F	F0	1000 0000	0x80

7.5.7. TCP Connections (0x87) Control Vector

The TCP Connections control vector indicates the support of an alternate number of TCP Connections for the Data Link Switching traffic. The base implementation of Data Link Switching supports two TCP Connections, one for each direction of data traffic.

This control vector is optional. If it is omitted in a DLSw Capabilities Exchange, then two TCP Connections are assumed. It is further assumed that if a Data Link Switch can support one TCP circuit~~session~~, it can support two TCP Connections.

If TCP Connections CV values agree and the number of connections is one, then the DLSw with the higher IP address must tear down the TCP connections on its local port 2065.

The format of the TCP Connections Control Vector is shown below:

Offset	Length	Value	Contents
0	1	0x03	Length of the TCP Connections structured subfield
1	1	0x87	key = 0x87 that identifies this as the TCP Connections structured subfield
2	1		an indicator of the support for an alternate number of TCP Connections by the sending Data Link Switch. 0x01 the number of TCP Connections may be brought down to one after Capabilities Exchange is completed. 0x02 the number of TCP Connections will remain at two for the duration of the DLSw connection.

7.5.8. NetBIOS Name Exclusivity (0x88) Control Vector

The NetBIOS Name Exclusivity control vector identifies how the NetBIOS Name List control vector data is to be interpreted. Specifically, this control vector identifies whether the NetBIOS Names in the NetBIOS Name List control vectors are the only ones accessible via the sending Data Link Switch.

If a NetBIOS Name List control vector is specified and the NetBIOS Name Exclusivity control vector is missing, then the NetBIOS Names are not assumed to be the only ones accessible via this switch.

A node may specify that it supports no local NetBIOS names by including in its capabilities the NetBIOS Name List Exclusivity CV (with byte 2 == 0x01), and not including any instances of the NetBIOS Name List CV.

Offset	Length	Value	Contents
0	1	0x03	Length of the Exclusivity structured subfield
1	1	0x88	key = 0x88 that identifies this as the NetBIOS Name Exclusivity structured subfield
2	1		an indicator of the relationship of the NetBIOS Names to the sending Data Link Switch. 0x00 the NetBIOS Names specified in this Capabilities Exchange can be accessed via this switch but are not the exclusive set (i.e. other entities are accessible in addition to the ones specified) 0x01 the NetBIOS Names specified in this Capabilities Exchange are the only ones accessible via this switch.

7.5.9. MAC Address List (0x89) Control Vector

The MAC Address List control vector identifies one or more MAC addresses that are accessible through the sending Data Link Switch. This control vector specifies a single MAC address value and MAC address mask value to identify the MAC address or range of MAC addresses. MAC addresses and masks are in non-canonical (Token-Ring) format in this control vector.

This control vector is optional and can be repeated if necessary.

Note 1: If a particular MAC address, <mac-addr>, satisfies the following algorithm, then <mac-addr> is assumed to be accessible via the sending Data Link Switch:

```
<mac-addr> & <mac-addr-mask> == <mac-addr-value>

where:  <mac-addr-value> is the MAC Address
                         Value specified in
                         this control vector

        <mac-addr-mask>  is the MAC Address
                         Mask specified in
                         this control vector
```

Note 2: If an individual MAC Address is desired, then <mac-addr-value> should be the individual MAC address and <mac-addr-mask> should be 0xFFFFFFFFFFFF.

Offset	Length	Value	Contents
0	1	0x0E	Length of the MAC Address List structured subfield
1	1	0x89	key = 0x89 that identifies this as the MAC Address List structured subfield
2-7	6		the 6-byte MAC Address Value, <mac-addr-value> in the above formula
8-13	6		the 6-byte MAC Address Mask, <mac-addr-mask> in the above formula

7.5.10. NetBIOS Name List (0x8A) Control Vector

The NetBIOS Name List control vector identifies one or more NetBIOS names that are accessible through the sending Data Link Switch. This control vector specifies a single NetBIOS name in ASCII. However, the NetBIOS name can consist of "don't care" and "wildcard" characters to match on a number of NetBIOS names. If an individual character position in the NetBIOS name in this control vector contains a '?', then the corresponding character position in real NetBIOS name is a "don't care". If a NetBIOS name in this control vector ends in '*', then the remainder of real NetBIOS names is a "don't care". '*' is only considered a wildcard if it appears at the end of a name.

All blanks or nulls at the end of NetBIOS names in this control vector are ignored. NetBIOS names which have fewer than 16 bytes and which do not end with '*' are not assumed to have a trailing '*'; the "wildcard" character must be explicit.

NetBIOS group names can exist across several LANs/networks. As such, NetBIOS group names received in a NetBIOS Name List Control Vector can not be treated the same as NetBIOS individual names. The Individual/Group Flag allows Data Link Switches to distinguish between the two.

This control vector is optional and can be repeated if necessary.

Offset	Length	Value	Contents
0	1	0xn	Length of the NetBIOS Name List structured subfield (maximum = 0x13)
1	1	0x8A	key = 0x8A that identifies this as the NetBIOS Name List structured subfield
2	1		Individual/Group Flag 0x00 - Individual NetBIOS Name 0x01 - Group NetBIOS Name
3-n	n-3		the NetBIOS name with possible embedded '?' and terminating '*'.

7.5.11. Vendor Context (0x8B) Control Vector

The Vendor Context control vector identifies the manufacturer's IEEE assigned Organizationally Unique Identifier (OUI) of the Data Link Switch sending the DLSw Capabilities Exchange. The OUI is sent in non-canonical (Token-Ring) format.

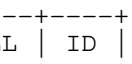
This control vector is optional and is used to provide the context for any Vendor Specific control vectors that follow in the Capabilities Exchange. If there are multiple instances of the Vendor Context control vector, the specified context remains in effect for all Vendor Specific control vectors until the next Vendor Context control vector is encountered.

Offset	Length	Value	Contents
0	1	0x05	Length of the Vendor Context structured subfield
1	1	0x8B	key = 0x8B that identifies this as the Vendor Context structured subfield
2-4	3		the 3-byte Organizationally Unique Identifier (OUI) for the vendor (non-canonical format)

7.6. Capabilities Exchange Responses

There are two kinds of DLSw Capabilities Exchange Responses: positive and negative. A positive response is returned to the sending Data Link Switch if there were no errors encountered in the DLSw Capabilities Exchange Request. A negative response is returned if there is at least one error encountered.

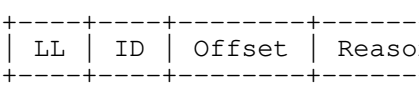
A positive DLSw Capabilities Exchange Response has the following overall format:



0-1 Length, in binary, of the DLSw Capabilities Exchange Response GDS Variable. The value of LL in this case is 0x0004.

2-3 GDS Id: 0x1521

A negative DLSw Capabilities Exchange Response has the following overall format:



0-1 Length, in binary, of the DLSw Capabilities Exchange Response GDS Variable. The value of LL is the sum of the length of all fields in the GDS Variable (i.e. length of LL + length of ID + length of Offsets/Reasons).

2-3 GDS Id: 0x1522

4-5 Offset into the DLSw Capabilities Exchange Request of the error. Offset should always point to the start of the GDS Variable or a specific control vector.

6-7 Reason code that uniquely identifies the error. Specific values for the reason code are:

0x0001	invalid GDS length for a DLSw Capabilities Exchange Request. (The value of Offset is ignored.)
0x0002	invalid GDS id for a DLSw Capabilities Exchange Request. (The value of Offset is ignored.)
0x0003	Vendor Id control vector is missing. (The value of Offset is ignored.)
0x0004	DLSw Version control vector is missing. (The value of Offset is ignored.)
0x0005	Initial Pacing Window control vector is missing. (The value of Offset is ignored.)
0x0006	length of control vectors doesn't correlate to the length of the GDS variable
0x0007	invalid control vector id
0x0008	length of control vector invalid
0x0009	invalid control vector data value
0x000A	duplicate control vector (for non-repeating control vectors)
0x000B	out-of-sequence control vector (for repeating control vector)
0x000C	DLSw Supported SAP List control vector is missing. (The value of Offset is ignored.)

Note: Multiple Offset, Reason pairs can be returned with one pair for each error encountered.

8. Pacing/Flow Control

This section describes the required Pacing and Flow Control mechanisms used by a Data Link Switch.

While it is beyond the scope of this document to specify a policy for how an implementation maps SSP flow control to the native data link flow control at the edges, the following paragraphs describe a general philosophical overview of how the mechanism is to be applied.

There are two types of flows which are covered by the flow control mechanism: connection-oriented and connectionless. In the first, connection-oriented flows, the implementer is to map the native flow control mechanism of the two data links at the boundaries to the SSP flow control mechanism thus presenting an end-to-end flow control mechanism which "pushes back" all the way to the originating station in either direction.

However, in the case of connectionless traffic, this is not possible at the data link level because there is no native flow control mechanism for connectionless data links. At first glance it is tempting to allow connectionless traffic to flow the DLSw cloud unthrottled. However, the rationale for subjecting these flows to flow control within the DLSw cloud is to "push" the discarding of frames (should this become necessary) back to the ingress of the DLSw cloud. This "early discarding" of excessive DATAGRAMs should allow the cloud to remain deterministic without wasting network bandwidth.

8.1. Basic Overview

Each circuit~~session~~ consists of two data flows, one in each direction. Each data flow has its own independent flow control mechanism. For each data flow there is an entity that originates traffic, referred to as the sender, and a target entity which receives the traffic, referred to as the receiver.

A sender may only send data when its receiver has granted explicit permission to send a discrete~~discreet~~ number of data units. Data units are defined as either a DGRMFRAME or an INFOFRAME.

The receiver grants permission to send data units by sending a Flow Control Indicator (FCIND- defined later). The sender must acknowledge all FCINDs by sending a Flow Control Acknowledgment (FCACK- defined later).

A sending implementation must maintain these values:

1. GrantedUnits - The number of units (frames) which the sender currently has permission to send.

2. CurrentWindow - This is a discrete~~discreet~~ number of units, controlled by the receiver, which is basis for granting additional units.

3. InitialWindowSize - Global for all circuits on a transport connection. Learned in capabilities exchange when the transport connection is established. It specifies an initial value for CurrentWindow when each circuit is established.

A receiving implementation must maintain these values:

1. CurrentWindow - This is a discrete~~discreet~~ number of units, controlled by the receiver, which is basis for granting additional units.

2. InitialWindowSize - Global for all circuits on a transport connection. Sent in capabilities exchange when the transport connection is established. It specifies an initial value for CurrentWindow when each circuit is established.

3. FCACKOwed - The sender owes an FCACK. If true, no FCIND may be sent.

8.2. Frame Format

The Flow control Byte is contained at offset 15 in both the Information and Control SSP messages. From a flow control perspective, the flow control information in the two frames are handled identically.

The following diagram describes the format of the Flow Control Byte (Bit 7 is the most significant and Bit 0 is the Least significant bit of the octet):

```
bit   7   6   5   4   3   2   1   0
    +---+---+---+---+---+---+---+---+
    |FCI|FCA| reserved  |    FCO    |
    +---+---+---+---+---+---+---+---+

FCI : Flow Control Indicator
FCA : Flow Control Ack
FCO : Flow Control Operator Bits

      000 - Repeat Window Operator
      001 - Increment Window Operator
      010 - Decrement Window Operator
      011 - Reset Window Operator
      100 - Halve Window Operator
      101 - Reserved
      110 - Reserved
      111 - Reserved
```

A frame with the FCI bit set is referred to as a Flow Control Indication (FCIND). An FCIND is used to manage the flow in the opposite direction of the frame which bears it.

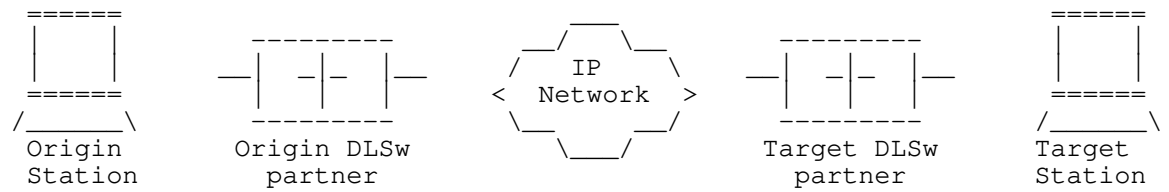
A frame with the FCA bit set is referred to as a Flow Control Acknowledgment (FCACK). An FCACK is used to manage the flow in the same direction of the frame which bears it.

NOTE: A frame may be both a FCIND and an FCACK.

A frame bearing an FCIND or FCACK may also contain data for the flow in the direction it is traveling. In such a frame, the FCIND or FCACK are said to be piggy-backed. A non-piggy-backed FCIND is called an Independent Flow Control Indication (IFCIND) and a non-piggy-backed FCACK is called an Independent Flow Control Acknowledgment (IFCACK). IFCIND and IFCACK messages are sent in a Independent Flow Control SSP message (type 0x21).

NOTE: A frame may be both an IFCIND and an IFCACK.

It is desirable to carry information in control messages so as to reduce the need to send a flow control only message. The diagram below shows the messages that may carry valid flow control information:




```
May have valid
 FCI/FCA/FCO    Data carrying

      N              N          CANUREACH_cs
      Y*             N          CANUREACH_cs
                                ----------->
      Y*             N            ICANREACH_cs
      Y              N            ICANREACH_cs
                                  <-----------
      Y              N          REACH_ACK
                                ----------->
      Y              Y            XIDFRAMEs
                                <------------>
      Y              Y            DGRMFRAMEs
                                <------------>
      Y              N          CONTACT
                                ----------->
      Y              N               CONTACTED
                                  <-----------
      Y              Y             INFOFRAMEs
                                <------------>
      Y              N          RESTART_DL
                                ----------->
      Y              N               DL_RESTARTED
                                  <-----------
      Y              N          CONTACT
                                ----------->
      Y              N               CONTACTED
                                  <-----------
      N              N          HALT_DL
                                ----------->
      N              N               DL_HALTED
                                  <-----------
```

*Note: ICANREACH_cs cannot carry FCA, as there could not be an outstanding FCI.

~~*Note: CANUREACH_cs cannot carry FCA, as there could not be an outstanding FCI.~~

8.3. Granting Permission to Send Data

A receiver grants a sender permission to send units of data by sending FCIND. Each FCIND is further qualified by a flow control operator, which is encoded in the FCO bits of the FCIND header. With one exception (the Reset Window operator) all operators may be either piggy-backed or carried in a IFCIND.

The five flow control operators are outlined below:

8.3.1. Repeat Window Operator

This operator is processed as follows:

```
(CurrentWindow unchanged)
```

```
GrantedUnits += CurrentWindow
```

8.3.2. Increment Window Operator

This operator is processed as follows:

```
CurrentWindow++
GrantedUnits += CurrentWindow
```

8.3.3. Decrement Window Operator

This operator is processed as follows:

```
CurrentWindow--
GrantedUnits += CurrentWindow
```

NOTE: This operator may only be sent if CurrentWindow is greater than one.

8.3.4. Reset Window Operator

This operator is processed as follows:

```
CurrentWindow = 0;
GrantedUnits  = 0;
```

NOTE: This operator may only flow on an independent pacing indication (may NOT be piggy-backed).

NOTE: After sending this operator, the only legal subsequent operator is Increment Window.

8.3.5. Halve Window Operator

This operator shall be processed as follows:

```
IF CurrentWindow > 1 THEN
   CurrentWindow = CurrentWindow / 2
ENDIF
GrantedUnits += CurrentWindow
```

Note: The divide by two operation is an unsigned integer divide (round down) or bit shift right operation.

~~EDITOR'S NOTE: WE HAVE DEFINED THIS OPERATOR, BUT THERE IS NO TEXT OR PSEUDO-CODE INDCATING HOW TO USE THIS.~~

8.4. Acknowledging a Flow Control Operator

Each sender must acknowledge each FCIND with an FCACK which is piggy-backed on the next frame in the opposite direction in all cases except the Reset Window Operator.

The receiver may have no more than one unacknowledged FCIND outstanding at any time with one exception: A Reset Window Operator may be sent while another FCIND is pending acknowledgment.

NOTE: The FCI and FCO bits of the FCACK are used independently by the flow in the opposite direction

8.4.1. Acknowledging a Reset Window Operator

Since this operator revokes all previously granted units, the sender must acknowledge this FCIND using an IFCACK (Independent Flow Control Acknowledgment). This is the only case where IFCACK is used.

Should a sender receive a non-reset FCIND followed by a Reset Window FCIND before acknowledging the first, it only acknowledges the Reset Window.

NOTE: The FCI and FCO bits on these frames are used independently by the flow in the opposite direction.

8.5. Capabilities Exchange Initial Window Size

When two nodes establish a transport connection, they engage in a capabilities exchange (this is a requirement). Refer to the Capabilities Exchange section 7 for further details. The two nodes are required to exchange the following parameter:

```
InitialWindowSize -  This indicates to the partner what
                     the sending flow entity initializes
                     its CurrentWindow value to for each
                     multiplexed circuit~~session~~ subsequently
                     established on that transport
                     connection.  This value must be
                     non-zero.
```

8.6. Circuit Startup

Process as follows:

```
CurrentWindow = InitialWindowSize
GrantedUnits  = 0
```

NOTE: The InitialWindow Size variable has a scope of one per DLSw transport connection, while CurrentWindow and Granted units are maintained on a per circuit basis. At circuit~~session~~ startup, a sender may not send data units until the receiver grants explicit permission with an FCIND message. This grant may be an independent FCIND message or the FCIND may be piggy-backed on any of the following message types listed in section 8.2.
~~(to reduce the startup flows):~~

~~CANUREACH_es~~
~~ICANREACH_es~~
~~REACH_ACK~~
~~XIDFRAMEs~~
~~DGRMFRAMEs~~
~~CONTACT~~
~~CONTACTED~~

8.7. Example Receiving Implementations

The following two examples illustrate receiving implementations of varying degrees of complexity. These are not meant to be complete implementations but rather serve to illustrate the protocol.

NOTE: The examples are independent of the buffering model (buffers may be deterministicly or statistically committed)

NOTE: The examples assume a process model where each event processes to completion without being preempted by another event.

8.7.1. Fixed Pacing Example

Consider the following variables, in addition to InitialWindowSize and CurrentWindow and FCACKOwed:

```
GrantDelayed     - Boolean
GrantedUnits     - Outstanding Units
```

The following section describes how various events are processed in this example implementation:

8.7.1.1. Circuit Startup

```
CurrentWindow    = InitialWindowSize
FCACKOwed        = FALSE
GrantDelayed     = FALSE
GrantedUnits     = 0
Repeat Window Operator
```

8.7.1.2. Check Buffers Available

```
Can my implementation afford to grant CurrentWindow just now?
```

8.7.1.3. Buffers Become Available

```
IF Check Buffers Available THEN
   Send FCIND( Repeat Window)
   GrantDelayed = FALSE
ELSE
   Wait on buffers to become available (LIFO)
ENDIF
```

8.7.1.4. Repeat Window Operator

```
IF Check Buffers Available THEN
    Send FCIND( Repeat Window)
ELSE
   GrantDelayed = TRUE
   Wait on buffers to become available (FIFO)
ENDIF
```

8.7.1.5. Send FCIND(operator)

```
GrantedUnits += CurrentWindow
FCACKOwed     = TRUE
Encode and Transmit FCIND piggybacked or as IFCIND
```

8.7.1.6. A Frame Arrives from Sender

```
GrantedUnits--;
IF frame is FCACK THEN
   IF FCACKOwed THEN
      FCACKOwed = FALSE
   ELSE
      Protocol Violation
   ENDIF
ENDIF
IF NOT GrantDelayed THEN
   IF GrantedUnits <= CurrentWindow THEN
       IF FCACKOwed THEN
         Protocol Violation
       ELSE
         Repeat Window Operator
       ENDIF
   ENDIF
ENDIF
```

8.7.2. Adaptive Pacing Example

The following example illustrates a receiving implementation that adjusts the window size and granted units based on buffer availability and transport utilization.

NOTE: This example ignores other factors which might compel the receiving implementation to adjust the window size (i.e. Outbound queue length, traffic priority, ...)

Consider the following variables, in addition to InitialWindowSize, CurrentWindow and FCACKOwed:

```
GrantDelayed     - Boolean
GrantedUnits     - Outstanding Units
```

8.7.2.1. Circuit Startup

```
CurrentWindow    = InitialWindowSize
FCACK            = FALSE
GrantDelayed     = FALSE
GrantedUnits     = 0
Repeat Window Operator
```

8.7.2.2. Check Buffers Available (X)

```
Can my implementation afford to grant X units just now?
```

8.7.2.3. Buffers Become Available

```
IF Check Buffers Available THEN
   CurrentWindow--;
   Send FCIND( Decrement Window)
   GrantDelayed = FALSE
ELSE
   Wait on buffers to become available (LIFO)
ENDIF
```

8.7.2.4. Repeat Window Operator

```
IF Check Buffers Available (CurrentWindow) THEN
    Send FCIND( Repeat Window)
ELSE
   GrantDelayed = TRUE
   Wait on buffers to become available (FIFO)
ENDIF
```

8.7.2.5. Increment Window Operator

```
IF Check Buffers Available ( CurrentWindow + 1) THEN
    CurrentWindow++
    Send FCIND( Increment Window)
ELSE
    Repeat Window Operator
ENDIF
```

8.7.2.6. Send FCIND(operator)

```
FCACKOwed     = TRUE
GrantedUnits += CurrentWindow
Encode and Transmit FCIND piggybacked or as IFCIND
```

8.7.2.7. An FCACK Arrives from Sender

```
GrantedUnits--;
IF NOT FCACKOwed THEN
   Protocol Violation
ENDIF
FCACKOwed = FALSE;
IF NOT GrantDelayed THEN
   IF GrantedUnits < CurrentWindow THEN
       Increment Window Operator
   ELSE IF GrantedUnits == CurrentWindow THEN
       Repeat Window Operator
   END
ENDIF
```

8.7.2.8. A Non-FCACK Frame Arrives from Sender

```
GrantedUnits--;
IF NOT GrantDelayed THEN
   IF FCACKOwed THEN
      IF GrantedUnits < CurrentWindow THEN
         Protocol Violation
      END
   ELSE
      IF GrantedUnits <= CurrentWindow THEN
         Repeat Window Operator
      ENDIF
   ENDIF
ENDIF
```

8.8 Adaptive Pacing Example Flow Diagrams

8.8.1 Example Flows from the Above Implementation

The following diagram illustrates the use of adaptive pacing (use of Halve Window, and Reset operation are shown in subsequent diagrams).

```
-----SENDER-----                      ----RECEIVER----
Granted   Window                      Window   Granted
  0         2   circuit established    2         0
  2         2   <-------- FCIND(Rpt)   2         2
  1         2   FCACK-------------->   2         1
  4         3   <-------- FCIND(Inc)   3         4
  3         3   FCACK-------------->   3         3
                        +- FCIND(Rpt)   3         6
  2         3   DATA---|----------->   3         5
  1         3   DATA---|----------->   3         4
  4         3   <------+
  3         3   FCACK-------------->   3         3
  6         3   <-------- FCIND(Rpt)   3         6
  5         3   FCACK-------------->   3         5
  4         3   DATA--------------->   3         4
  3         3   DATA--------------->   3         3
                        +- FCIND(Rpt)   3         6
  2         3   DATA---|----------->   3         5
  1         3   DATA---|----------->   3         4
  0         3   DATA---|----------->   3         3
  3         3   <------+
  2         3   FCACK-------------->   3         2
  6         4   <-------- FCIND(Inc)   4         6
  5         4   FCACK-------------->   4         5
  4         4   DATA--------------->   4         4
                                      Waiting on Buffer
                        +- FCIND(Dec)   3         7
  3         4   DATA---|----------->   3         6
  2         4   DATA---|----------->   3         5
  1         4   DATA---|----------->   3         4
  0         4   DATA---|----------->   3         3
  3         3   <------+
  2         3   FCACK-------------->   3         2
                                      Waiting on Buffer
                        +- FCIND(Dec)   2         4
  1         3   DATA---|----------->   2         3
  0         3   DATA---|----------->   2         2
  2         2   <------+
  1         2   FCACK-------------->   2         1
  4         3   <-------- FCIND(Inc)   3         4
  3         3   FCACK-------------->   3         3
  6         3   <-------- FCIND(Rpt)   3         6
  5         3   FCACK-------------->   3         5
  4         3   DATA--------------->   3         4
  3         3   DATA--------------->   3         3
  6         3   <-------- FCIND(Rpt)   3         6
```

8.8.2. Example Halve Window Flow

The following flow illustrates the use of the Halve Window Operator:

```
   -----SENDER-----                      ----RECEIVER----
   Granted   Window                      Window   Granted
     0         2   circuit established    2         0
     2         2   <-------- FCIND(Rpt)   2         2
     1         2   FCACK-------------->   2         1
     4         3   <-------- FCIND(Inc)   3         4
     3         3   FCACK-------------->   3         3
                                          Resource Shortage
     2         3   DATA--------------->   1         2
     1         3   DATA--------------->   1         1
     0         3   DATA--------------->   1         0
     1         1   <-------- FCIND(Hlv)   1         1
     0         1   FCACK-------------->   1         0
```

NOTE: The Halve Window Operator could have been sent before the granted units fell to zero. The implementer may make a choice based on the severity of the condition.

8.8.3. Example Reset Window Flows

The following flow diagram illustrates the ResetWindow operation if the receiver has no FCIND outstanding.

```
-----SENDER-----                      ----RECEIVER----
Granted   Window                      Window   Granted
  0         2   circuit established    2         0
  2         2   <-------- FCIND(Rpt)   2         2
  1         2   FCACK-------------->   2         1
  4         3   <-------- FCIND(Inc)   3         4
  3         3   FCACK-------------->   3         3
                    +- FCIND(Rpt)      3         6
  2         3   DATA---|----------->   3         5
  1         3   DATA---|----------->   3         4
  4         3   <------+
  3         3   FCACK-------------->   3         3
  6         3   <-------- FCIND(Rpt)   3         6
  5         3   FCACK-------------->   3         5
                                       Resource shortage!
  0         0   <-------- FCIND(Rst)   0         5 (note still committed)
  0         0   IFCACK------------->   0         0
                                       Condition eases
  1         1   <-------- FCIND(Inc)   1         1
  0         1   FCACK-------------->   1         0
  2         2   <-------- FCIND(Inc)   2         2
  1         2   FCACK-------------->   3         4
  1         2   DATA--------------->   3         4
```

The next two flows illustrate the Reset Window operation if the receiver has an outstanding FCIND.

```
-----SENDER-----                          ----RECEIVER----
Granted   Window                          Window   Granted
  0          2   circuit established      2          0
  2          2   <-------- FCIND(Rpt)     2          2
  1          2   FCACK-------------->     2          1
  4          3   <-------- FCIND(Inc)     3          4
  3          3   FCACK-------------->     3          3
                        +- FCIND(Rpt)     3          6
  2          3   DATA---|---------->      3          5
                        |                 Resource shortage!
                        |+-FCIND(Rst)     0          5
  1          3   DATA---||---------->     0          4
  4          3   <------+|
  3          3   FCACK---+---------->     0          3 (Not IFCACK!)
  2          3   DATA----|---------->     0          2
  0          0   <-------+
  0          0   IFCACK------------->     0          0
                                          Condition eases
  1          1   <-------- FCIND(Inc)     1          1
  0          1   FCACK-------------->     1          0
  2          2   <-------- FCIND(Inc)     2          2
  1          2   FCACK-------------->     3          4
  1          2   DATA--------------->     3          4
```

```
-----SENDER-----                          ----RECEIVER----
Granted   Window                          Window   Granted
  0          2   circuit established      2          0
  2          2   <-------- FCIND(Rpt)     2          2
  1          2   FCACK-------------->     2          1
  4          3   <-------- FCIND(Inc)     3          4
  3          3   FCACK-------------->     3          3
                        +- FCIND(Rpt)     3          6
  2          3   DATA---|---------->      3          5
                        |                 Resource shortage!
                        |+-FCIND(Rst)     0          5
  1          3   DATA---||---------->     0          4
  4          3   <------+|
  0          0   <-------+
  0          0   IFCACK------------->     0          0
                                          Condition eases
  1          1   <-------- FCIND(Inc)     1          1
  0          1   FCACK-------------->     1          0
  2          2   <-------- FCIND(Inc)     2          2
  1          2   FCACK-------------->     3          4
  1          2   DATA--------------->     3          4
```

8.9. Other Considerations

8.9.1. Protocol Violations

The following events are considered protocol violations:

1. Sender exceeds granted units or does not acknowledge FCIND on first frame after its receipt (the receiver can not discern the difference between the two).

2. Receiver does not follow a Reset Window Operator with an Increment Window Operator.

3. Receiver has two unacknowledged FCINDs (other than Reset Window) outstanding.

4. Receiver sends Decrement Window Operator with a window size of one.

Actions taken in response to protocol violations are left to the implementation of the node which discovers the violation. If an implementation chooses to take down the circuit on which the violation occurred, HALT_DL is the appropriate action.

Acknowledgments

Original RFC 1434 Authors:

Roy C. Dixon, IBM
David M. Kushi, IBM

Chair of APPN Implementers Workshop Data Link Switching Related Interest Group:

Louise Herndon Wells, Internetworking Technology Institute

Working Group Chairs (and significant contributors to this document):

Connect/Disconnect (State Machines): Steve Klein, IBM
Capabilities Exchange: Wayne Clark, Cisco Systems
Flow Control (Adaptive Pacing): Shannon Nix, Metaplex
Priority/Class of Service: Gene Cox, IBM

Other significant contributors:

Peter Gayek, IBM
Paul Brittain, Data Connection Limited

References

1) ISO 8802-2/IEEE Std 802.2 International Standard,
Information Processing Systems, Local Area Networks, Part 2:
Logical Link Control, December 31, 1989

2) ~~The NETBIOS Frames Protocol,~~ IBM Local Area Technical Reference
~~Reference,~~ SC30-3383-03~~, Chapter 5~~, December 1990

3) ISO/IEC DIS 10038 DAM 2, MAC Bridging, Source Routing
Supplement, December 1991

4) ISO 8802-2/IEEE Std 802.1D International Standard,
Information Processing Systems, Local Area Networks, Part 2:
MAC layer Bridging.

Security Considerations

Security issues are not discussed in this memo.

Editor's Address

Alan K. Bartky
Manager of Technology
Sync Research Inc.
7 Studebaker
Irvine, CA 91728-2013

Phone: (714)588-2070
EMail: alan@sync.com

Note: Any questions or comments relative to the contents of this RFC should be sent to the following Internet address:
aiw-dlsw@networking.raleigh.ibm.com. This address will be used to coordinate the handling of responses.

NOTE 1: This is a widely subscribed mailing list and messages sent to this address will be sent to all members of the DLSw mailing list. For specific questions relating to subscribing to the AIW and any of it's working groups send email to:
aiw@networking.raleigh.ibm.com

NOTE 2: These mailing lists and addresses are subject to change.