                 Data Link Switching Client Access Protocol

Status of this Memo

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

Abstract

   This memo describes the Data Link Switching Client Access Protocol
   that is used between workstations and routers to transport SNA/
   NetBIOS traffic over TCP sessions. Any questions or comments should
   be sent to dcap@cisco.com.

Table of Contents

1. Introduction

    Since the Data Link Switching Protocol, RFC 1795, was published, some
    software vendors have begun implementing DLSw on workstations. The
    implementation of DLSw on a large number of workstations raises
    several important issues that must be addressed. Scalability is the
    major concern. For example, the number of TCP sessions to the DLSw
    router increases in direct proportion to the number of workstations
    added. Another concern is efficiency. Since DLSw is a switch-to-
    switch protocol, it is not efficient when implemented on
    workstations.

    DCAP addresses the above issues. It introduces a hierarchical
    structure to resolve the scalability problems. All workstations are
    clients to the router (server) rather than peers to the router. This
    creates a client/server model. It also provides a more efficient
    protocol between the workstation (client) and the router (server).

2. Overview

2.1. DCAP Client/Server Model

```
        +-----------+                +-----------+        +---------+
        | Mainframe |                | IP Router +- ppp -+ DLSw    |
        +--+--------+                +-----+-----+        | Work    |
           |                               |              | Station |
           |                               |              +---------+
        +--+--+      +-------------+       |
        | FEP +- TR -+ DLSw Router +-- IP Backbone
        +-----+      +-------------+       |
                                           |
                                           |
                                           |
                            +-----------+        +---------+
                            | IP Router +- ppp -+ DLSw    |
                            +-----+-----+        | Work    |
                                                 | Station |
                                                 +---------+

                   |            DLSw Session            |
                   +-----------------------------------+
```
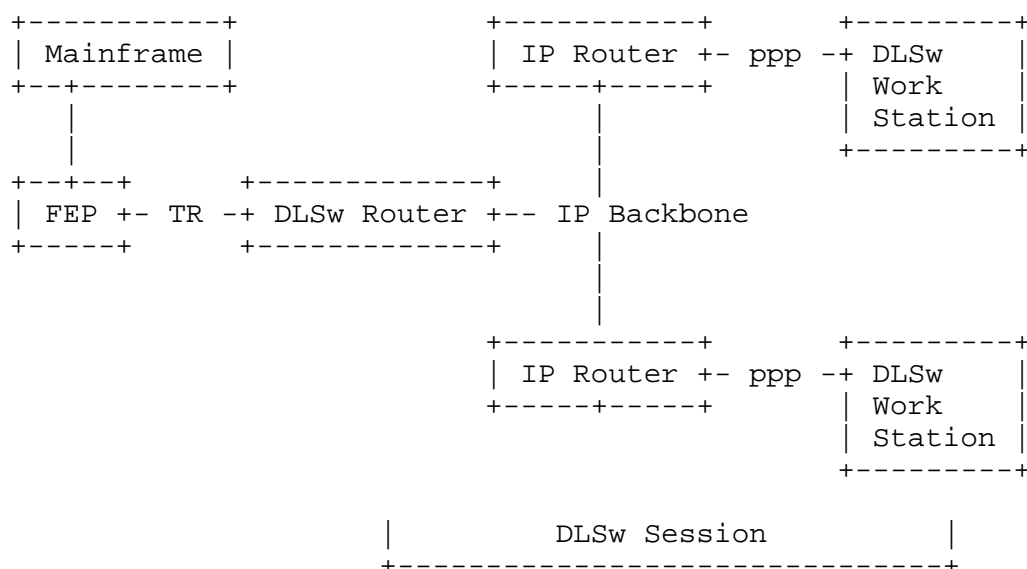   Figure 2-1. Running DLSw on a large number of workstations creates a
                        scalability problem.

    Figure 2-1 shows a typical DLSw implementation on a workstation. The
    workstations are connected to the central site DLSw router over the
    IP network. As the network grows, scalability will become an issue
    as the number of TCP sessions increases due to the growing number of
    workstations.

```
+-----------+                                       +--------+
| Mainframe |                                       | DCAP   |
+--+--------+                               +-----+ Client |
   |                                        |     +--------+
   |                                      ppp
   |                                        |
+--+--+      +--------+                  +------+------+
| FEP +- TR -+  DLSw  +-- IP Backbone --+ DLSw Router |
+-----+      | Router |                 | DCAP Server |
             +--------+                  +------+------+
                                                |
                                              ppp
                                                |     +--------+
                                          +-----+ DCAP   |
                                                | Client |
                                                +--------+

             |     DLSw Session      | | DCAP Session |
             +-----------------------+ +--------------+
```
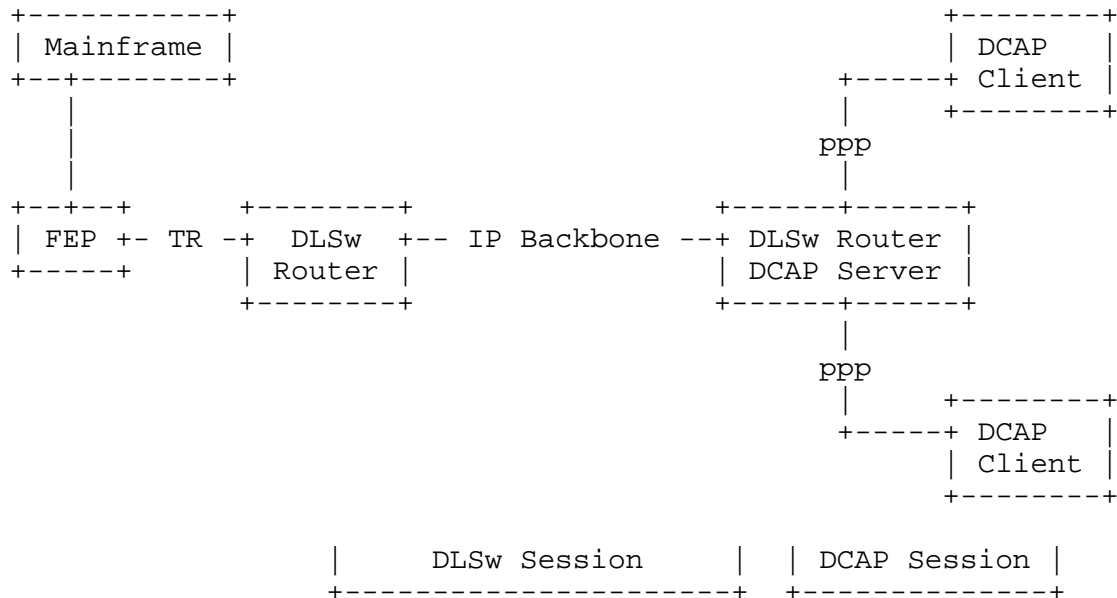   Figure 2-2. DLSw Client Access Protocol solves the scalability
                            problem.

   In a large network, DCAP addresses the scalability problem by
   significantly reducing the number of peers that connect to the
   central site router. The workstations (DCAP clients) and the router
   (DCAP server) behave in a Client/Server relationship. Workstations
   are attached to a DCAP server. A DCAP server has a single peer
   connection to the central site router.

2.2. Dynamic Address Resolution

   In a DLSw network, each workstation needs a MAC address to
   communicate with a FEP attached to a LAN. When DLSw is implemented on
   a workstation, it does not always have a MAC address defined. For
   example, when a workstation connects to a router through a modem via
   PPP, it only consists of an IP address. In this case, the user must
   define a virtual MAC address. This is administratively intensive
   since each workstation must have an unique MAC address.

   DCAP uses the Dynamic Address Resolution protocol to solve this
   problem. The Dynamic Address Resolution protocol permits the server
   to dynamically assign a MAC address to a client without complex
   configuration.

   For a client to initiate a session to a server, the workstation sends
   a direct request to the server. The request contains the destination
   MAC address and the destination SAP. The workstation can either
   specify its own MAC address, or request the server to assign one to

it. The server's IP address must be pre-configured on the
workstation. If IP addresses are configured for multiple servers at a
workstation, the request can be sent to these servers and the first
one to respond will be used.

For a server to initiate a session to a client, the server sends a
directed request to the workstation. The workstation must pre-
register its MAC address at the server. This can be done either by
configuration on the server or registration at the server (both MAC
addresses and IP addresses will be registered).

2.3. TCP Connection

The transport used between the client and the server is TCP. A TCP
session must be established between the client and the server before
a frame can be sent. The default parameters associated with the TCP
connections between the client and the server are as follows:

    Socket Family   AF_INET   (Internet protocols)
    Socket Type   SOCK_STREAM   (stream socket)
    Port Number   1973

There is only one TCP connection between the client and the server.
It is used for both read and write operations.

A race condition occurs when both client and server try to establish
the TCP session with each other at the same time. The TCP session of
the initiator with the lower IP address will be used. The other TCP
session will be closed.

2.4   Multicast and Unicast (UDP)

Multicast and unicast with UDP support are optional. In the reset of
this session, when multicast and unicast are referenced, UDP is used.
Two multicast addresses are reserved for DCAP. The server should
listen for 224.0.1.49 and the client should listen for 224.0.1.50.
Not all DCAP frames can be sent via multicast or unicast. The
DATA_FRAME can be sent via either multicast or unicast. The
CAN_U_REACH frame can be sent via multicast only and the I_CAN_REACH
frame can be sent via unicast only. All other DCAP frames can only be
sent via TCP sessions.

When the multicast and unicast support is implemented, the client
does not have to configure the server's IP address. When the client
attempts to establish a session to the host, instead of establishing
a TCP session with the pre-configured server, the client can
multicast the CAN_U_REACH frame to the 224.0.1.49 group address. When
the server receives this multicast frame, it will locate the

destination as specified in the frame. If the destination is
reachable by this server, it will send back an I_CAN_REACH frame to
the sender via unicast. The client can initiate a TCP connection to
the server and establish a DCAP session. If the I_CAN_REACH frame is
received from multiple servers, the first one who returns the
I_CAN_REACH frame will be used.

When the host initiates a session to the client, the client does not
have to pre-register its MAC address at the server. When the server
attempts to reach an unknown client, it will multicast the
CAN_U_REACH frame to the 224.0.10.50 group address. The client whose
MAC address matches the destination address in the CAN_U_REACH frame
will reply with the I_CAN_REACH frame via unicast. Once the server
receives the I_CAN_REACH frame, it can establish a DCAP session with
that client.

For NetBIOS traffic, NAME_QUERY and ADD_NAME_QUERY can be
encapsulated in the DATA_FRAME and sent out via multicast.
NAME_RECOGNIZED and ADD_NAME_RESPONSE can be encapsulated in the
DATA_FRAME but sent out via unicast. No other NetBIOS frames can be
encapsulated in the DATA_FRAME to be sent out via either multicast or
unicast.

When a client tries to locate a name or check for duplicate name on
the network, it can multicast a NAME_QUERY or ADD_NAME_QUERY frame
encapsulated in the DATA_FRAME. When a server receives these frames,
NetBIOS NAME_QUERY or ADD_NAME_QUERY frames will be forwarded to LAN.
If the NAME_RECOGNIZED or ADD_NAME_RESPONSE frame is received from
LAN, they will be encapsulated in the DATA_FRAME and sent to the
client via unicast.

When a server receives a NetBIOS NAME_QUERY or ADD_NAME_QUERY from
LAN, the server will encapsulate it in the DATA_FRAME and send it to
all clients via multicast. When a client receives the frame and
determines that the name specified in the DATA_FRAME matches its own
name, a NAME_RECOGNIZED or ADD_NAME_RESPONSE frame will be
encapsulated in the DATA_FRAME and sent back to the server via
unicast.

3. DCAP Format

3.1. General Frame Format

   The General format of the DCAP frame is as follows:
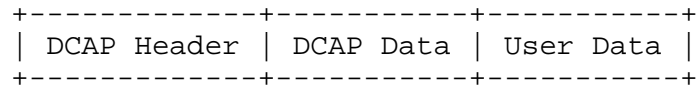
```
              +-------------+-----------+-----------+
              | DCAP Header | DCAP Data | User Data |
              +-------------+-----------+-----------+
                 Figure 3-1. DCAP Frame Format
```

   The DCAP protocol is contained in the DCAP header, which is common to
   all frames passed between the DCAP client and the server. This header
   is 4 bytes long. The next section will explain the details.

   The next part is the DCAP Data. The structure and the size are based
   on the type of messages carried in the DCAP frame. The DCAP data is
   used to process the frame, but it is optional.

   The third part of the frame is the user data, which is sent by the
   local system to the remote system. The size of this block is variable
   and is included in the frame only when there is data to be sent to
   the remote system.

3.2. Header Format

   The DCAP header is used to identify the message type and the length
   of the frame. This is a general purpose header used for each frame
   that is passed between the DCAP server and the client. More
   information is needed for frames like CAN_U_REACH and I_CAN_REACH,
   therefore, it is passed to the peer as DCAP data. The structure of
   the DCAP data depends on the type of frames, and will be discussed in
   detail in later sections.

   The DCAP Header is given below:
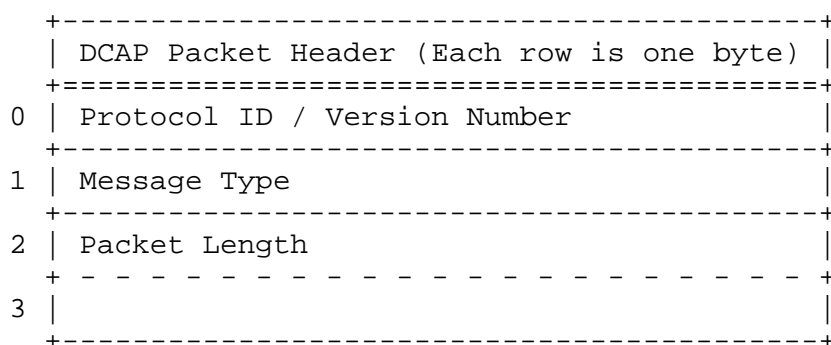
```
           +------------------------------------------+
           | DCAP Packet Header (Each row is one byte) |
           +==========================================+
        0 | Protocol ID / Version Number              |
           +------------------------------------------+
        1 | Message Type                              |
           +------------------------------------------+
        2 | Packet Length                             |
           + - - - - - - - - - - - - - - - - - - - - - +
        3 |                                           |
           +------------------------------------------+
               Figure 3-2. DCAP Header Format
```

   o The Protocol ID uses the first 4 bits of this field and is set to
     "1000".

   o The Version Number uses the next 4 bits in this field and is set
     to "0001".

   o The message type is the DCAP message type.

   o The Total Packet length is the length of the packet including the
     DCAP header, DCAP data and User Data. The minimum size of the
     packet is 4, which is the length of the header.

3.3. DCAP Messages

   Most of the DCAP frames are based on the existing DLSw frames and
   corresponding frames have similar names. The information in the
   corresponding DCAP and DLSw frames may differ; but the
   functionalities are the same. Thus the DLSw State Machine is used to
   handle these DCAP frames. Some new DCAP frames were created to handle
   special DCAP functions. For example, the new DCAP frames,
   I_CANNOT_REACH and START_DL_FAILED provide negative acknowledgment.
   The DLSw frames not needed for DCAP, are dropped.

   The following table lists and describes all available DCAP messages:

   DCAP Frame Name   Code   Function
   ---------------   ----   --------
   CAN_U_REACH   0x01   Find if the station given is reachable
   I_CAN_REACH   0x02   Positive response to CAN_U_REACH
   I_CANNOT_REACH   0x03   Negative response to CAN_U_REACH
   START_DL   0x04   Setup session for given addresses
   DL_STARTED   0x05   Session Started
   START_DL_FAILED   0x06   Session Start failed
   XID_FRAME   0x07   XID Frame
   CONTACT_STN   0x08   Contact destination to establish SABME
   STN_CONTACTED   0x09   Station contacted - SABME mode set
   DATA_FRAME   0x0A   Connectionless Data Frame for a link
   INFO_FRAME   0x0B   Connection oriented I-Frame
   HALT_DL   0x0C   Halt Data Link session
   HALT_DL_NOACK   0x0D   Halt Data Link session without ack
   DL_HALTED   0x0E   Session Halted
   FCM_FRAME   0x0F   Data Link Session Flow Control Message
   DGRM_FRAME   0x11   Connectionless Datagram Frame for a circuit

```
CAP_XCHANGE            0x12   Capabilities Exchange Message
CLOSE_PEER_REQUEST     0x13   Disconnect Peer Connection Request
CLOSE_PEER_RESPONSE    0x14   Disconnect Peer Connection Response
PEER_TEST_REQ          0x1D   Peer keepalive test request
PEER_TEST_RSP          0x1E   Peer keepalive response
```

Table 3-1. DCAP Frames

3.4. DCAP Data formats

The DCAP data is used to carry information required for each DCAP
frame. This information is used by the Server or the Client and it
does not contain any user data. The DCAP data frame types are listed
in the following sections. Please note that the sender should set the
reserved fields to zero and the receiver should ignore these fields.

3.4.1. CAN_U_REACH, I_CAN_REACH, and I_CANNOT_REACH Frames

These frame types are used to locate resources in a network. A
CAN_U_REACH frame is sent to the server to determine if the resource
is reachable. When a server receives a CAN_U_REACH frame, it should
send out an LLC explorer frame to locate the destination specified in
the CAN_U_REACH frame. If the destination is reachable, the server
responds to the client with an I_CAN_REACH frame. If the server does
not receive a positive acknowledgment within a recommended threshold
value of 5 seconds, the server should send an LLC explorer to locate
the destination again. If the server does not receive any response
after sending out 5 explorers (recommended retry value), the
destination is considered not reachable and an I_CANNOT_REACH frame
is sent back to the client. The client should decide if retry
CAN_U_REACH is necessary after the I_CANNOT_REACH frame is received
from the server.

When a server is in the process of searching a destination and
receives another I_CAN_REACH with the same destination, the server
should not send out another LLC explorer for that destination.

The server should not send the CAN_U_REACH frame to the clients in a
TCP session. When a server receives an LLC explorer whose destination
is a known client, the server should respond to it directly.

```
          +---------------+----------------------+
          | Field Name    | Information          |
          +---------------+----------------------+
          | Message Type  | 0x01, 0x02, or 0x03  |
          +---------------+----------------------+
          | Packet Length | 0x0C                 |
          +---------------+----------------------+
```
     Figure 3-3. CAN_U_REACH, I_CAN_REACH, and I_CANNOT_REACH Header

```
          +------------------------------------+
          | Field Name (Each row is one byte)  |
          +====================================+
        0 | Target MAC Address                 |
          + - - - - - - - - - - - - - - - - - -+
        1 |                                    |
          + - - - - - - - - - - - - - - - - - -+
        2 |                                    |
          + - - - - - - - - - - - - - - - - - -+
        3 |                                    |
          + - - - - - - - - - - - - - - - - - -+
        4 |                                    |
          + - - - - - - - - - - - - - - - - - -+
        5 |                                    |
          +------------------------------------+
        6 | Source SAP                         |
          +------------------------------------+
        7 | Reserved                           |
          +------------------------------------+
```
      Figure 3-4. CAN_U_REACH, I_CAN_REACH, and I_CANNOT_REACH Data

   The MAC Address field carries the MAC address of the target
   workstation that is being searched. This is a six-byte MAC Address
   field. The same MAC Address is returned in the I_CAN_REACH and the
   I_CANNOT_REACH frames.

   Byte 6 is the source SAP. The destination SAP is set to zero when an
   explorer frame is sent to the network.

3.4.2. START_DL, DL_STARTED, and START_DL_FAILED Frames

   These frame types are used by DCAP to establish a link station
   (circuit). The START_DL frame is sent directly to the server that
   responds to the CAN_U_REACH frame. When the server receives this
   frame, it establishes a link station using the source and destination
   addresses and saps provided in the START_DL frame. If the circuit
   establishment is successful, a DL_STARTED frame is sent back as a
   response. If the attempt fails within a recommended value, 5 seconds,
   the server should retry again. If the server fails to establish a

circuit for a recommended retry value, 5 times, a START_DL_FAILED
frame should be sent back to the client. If the client receives a
START_DL_FAILED frame from the server, it is up to the client to
decide if a START_DL frame needs to be sent to the server again.

The server can also send START_DL frames to clients to establish
circuits.

```
+---------------+-----------------------+
| Field Name    | Information           |
+---------------+-----------------------+
| Message Type  | 0x04, 0x05, or 0x06   |
+---------------+-----------------------+
| Packet Length | 0x18                  |
+---------------+-----------------------+
```
   Figure 3-5. START_DL, DL_STARTED, and START_DL_FAILED Header

```
      +-----------------------------------+
      | Field Name (Each row is one byte) |
      +===================================+
    0 | Host MAC Address                  |
      + - - - - - - - - - - - - - - - - - +
    1 |                                   |
      + - - - - - - - - - - - - - - - - - +
    2 |                                   |
      + - - - - - - - - - - - - - - - - - +
    3 |                                   |
      + - - - - - - - - - - - - - - - - - +
    4 |                                   |
      + - - - - - - - - - - - - - - - - - +
    5 |                                   |
      +-----------------------------------+
    6 | Host SAP                          |
      +-----------------------------------+
    7 | Client SAP                        |
      +-----------------------------------+
    8 | Origin Session ID                 |
      +-----------------------------------+
    9 |                                   |
      + - - - - - - - - - - - - - - - - - +
   10 |                                   |
      + - - - - - - - - - - - - - - - - - +
   11 |                                   |
      +-----------------------------------+
   12 | Target Session ID                 |
      + - - - - - - - - - - - - - - - - - +
   13 |                                   |
      + - - - - - - - - - - - - - - - - - +
   14 |                                   |
      + - - - - - - - - - - - - - - - - - +
   15 |                                   |
      +-----------------------------------+
   16 | Largest Frame Size                |
      +-----------------------------------+
   17 | Initial Window size               |
      +-----------------------------------+
   18 | Reserved                          |
      + - - - - - - - - - - - - - - - - - +
   19 |                                   |
      +-----------------------------------+
```

   Figure 3-6. START_DL, DL_STARTED, and START_DL_FAILED Data

   The Host MAC address is the address of the target station if the
   session is initiated from the client, or it is the address of the
   originating station if the session is initiated from the server.

The next two fields are the Host and Client SAPs. Each is one byte
long. The Host SAP is the SAP used by the station with the Host MAC
address. The Client SAP is the SAP used by the client.

The Origin Session ID, is the ID of the originating station that
initiates the circuit. The originating station uses this ID to
identify the newly created circuit. Before the START_DL frame is sent
to the target station, the originating station sets up a control
block for the circuit. This link station information is set because
DCAP does not use a three-way handshake for link station
establishment. In the DL_STARTED and the START_DL_FAILED frames, the
Origin Session ID is returned as received in the START_DL frame. The
Target Session ID is set by the target station and returned in the
DL_STARTED frame.

The Target Session ID is not valid for the START_DL and the
START_DL_FAILED frame, and should be treated as Reserved fields. In
the DL_STARTED frame, it is the session ID that is used to set up
this circuit by the target station.

The Largest Frame Size field is used to indicate the maximum frame
size that can be used by the client. It is valid only when it is set
by the server. The Largest Frame Size field must be set to zero when
a frame is sent by the client. Both START_DL and DL_STARTED use the
Largest Frame Size field and only its rightmost 6 bits are used. The
format is defined in the IEEE 802.1D Standard, Annex C, Largest Frame
Bits (LF). Bit 3 to bit 5 are base bits. Bit 0 to bit 2 are extended
bits. The Largest Frame Size field is not used in the START_DL_FAILED
frame and must be set to zero.

```
          bit   7    6    5    4    3    2    1    0
                r    r    b    b    b    e    e    e
                   Figure 3-7. Largest Frame Size
```

Please note that if the client is a PU 2.1 node, the client should
use the maximum I-frame size negotiated in the XID3 exchange.

The Initial window size in the START_DL frame specifies the receive
window size on the originating side, and the target DCAP station
returns its receive window size in the DL_STARTED frame. The field is
reserved in the START_DL_FAILED frame. The usage of the window size
is the same as the one used in DLSw. Please refer to RFC 1795 for
details.

The last two bits are reserved for future use. They must be set to
zero by the sender and ignored by the receiver.

3.4.3. HALT_DL, HALT_DL_NOACK, and DL_HALTED Frames

   These frame types are used by DCAP to disconnect a link station. A
   HALT_DL frame is sent directly to the remote workstation to indicate
   that the sender wishes to disconnect a session. When the receiver
   receives this frame, it tears down the session that is associated
   with the Original Session ID and the Target Session ID provided in
   the HALT_DL frame. The receiver should respond with the DL_HALTED
   frame. The DL_HALTED frame should use the same Session ID values as
   the received HALT_DL frame without swapping them. The HALT_DL_NOACK
   frame is used when the response is not required. The TCP session
   between the client and server should remain up after the
   HALT_DL/DL_HALTED/ HALT_DL_NOACK exchange.

```
            +---------------+-----------------------+
            | Field Name    | Information           |
            +---------------+-----------------------+
            | Message Type  | 0x0C, 0x0D, or 0x0E   |
            +---------------+-----------------------+
            | Packet Length | 0x10                  |
            +---------------+-----------------------+
```
          Figure 3-8. HALT_DL, HALT_DL_NOACK, and DL_HALTED Header

```
          +------------------------------------+
          | Field Name (Each row is one byte)  |
          +====================================+
       0  | Sender Session ID                  |
          + - - - - - - - - - - - - - - - - - -+
       1  |                                    |
          + - - - - - - - - - - - - - - - - - -+
       2  |                                    |
          + - - - - - - - - - - - - - - - - - -+
       3  |                                    |
          +------------------------------------+
       4  | Receiver Session ID                |
          + - - - - - - - - - - - - - - - - - -+
       5  |                                    |
          + - - - - - - - - - - - - - - - - - -+
       6  |                                    |
          + - - - - - - - - - - - - - - - - - -+
       7  |                                    |
          +------------------------------------+
       8  | Reserved                           |
          + - - - - - - - - - - - - - - - - - -+
       9  |                                    |
          + - - - - - - - - - - - - - - - - - -+
      10  |                                    |
          + - - - - - - - - - - - - - - - - - -+
      11  |                                    |
          +------------------------------------+
```
         Figure 3-9. START_DL, DL_STARTED, and START_DL_FAILED Data

3.4.4. XID_FRAME, CONTACT_STN, STN_CONTACTED, INFO_FRAME, FCM_FRAME,
and DGRM_FRAME

   These frame types are used to carry the end-to-end data or establish
   a circuit. The Destination Session ID is the Session ID created in
   the START_DL frame or the DL_STARTED frame by the receiver. The usage
   of the flow control flag is the same as the one used in DLSw. Please
   refer to RFC 1795 for details.

```
          +---------------+---------------------------+
          | Field Name    | Information               |
          +---------------+---------------------------+
          | Message Type  | Based on Message type     |
          +---------------+---------------------------+
          | Packet Length | 0x0C + length of user data|
          +---------------+---------------------------+
```
              Figure 3-10. Generic DCAP Header

```
        +-----------------------------------+
        | Field Name (Each row is one byte) |
        +===================================+
     0  | Destination Session ID            |
        + - - - - - - - - - - - - - - - - - +
     1  |                                   |
        + - - - - - - - - - - - - - - - - - +
     2  |                                   |
        + - - - - - - - - - - - - - - - - - +
     3  |                                   |
        +-----------------------------------+
     4  | Flow Control Flags                |
        +-----------------------------------+
     5  | Reserved                          |
        + - - - - - - - - - - - - - - - - - +
     6  |                                   |
        + - - - - - - - - - - - - - - - - - +
     7  |                                   |
        +-----------------------------------+
        Figure 3-11. Generic DCAP Data Format
```

3.4.5. DATA_FRAME

   This frame type is used to send connectionless SNA and NetBIOS
   Datagram (UI) frames that do not have a link station associated with
   the source and destination MAC/SAP pair. The difference between
   DGRM_FRAME and DATA_FRAME is that DGRM_FRAME is used to send UI
   frames received for stations that have a link station opened, whereas
   DATA_FRAME is used for frames with no link station established.

```
        +---------------+----------------------------+
        | Field Name    | Information                |
        +---------------+----------------------------+
        | Message Type  | 0x0A                       |
        +---------------+----------------------------+
        | Packet Length | 0x10 + Length of user data |
        +---------------+----------------------------+
                Figure 3-12. DATA_FRAME Header
```

```
        +-----------------------------------+
        | Field Name (Each row is one byte) |
        +===================================+
     0 | Host MAC Address                  |
        + - - - - - - - - - - - - - - - - - +
     1 |                                   |
        + - - - - - - - - - - - - - - - - - +
     2 |                                   |
        + - - - - - - - - - - - - - - - - - +
     3 |                                   |
        + - - - - - - - - - - - - - - - - - +
     4 |                                   |
        + - - - - - - - - - - - - - - - - - +
     5 |                                   |
        +-----------------------------------+
     6 | Host SAP                          |
        +-----------------------------------+
     7 | Client SAP                        |
        +-----------------------------------+
     8 | Broadcast Type                    |
        +-----------------------------------+
     9 | Reserved                          |
        + - - - - - - - - - - - - - - - - - +
    10 |                                   |
        + - - - - - - - - - - - - - - - - - +
    11 |                                   |
        +-----------------------------------+
```
               Figure 3-13. DATA_FRAME Data Format

   The definition of the first 8 bytes is the same as the START_DL
   frame. The Broadcast Type field indicates the type of broadcast
   frames in use; Single Route Broadcast, All Route Broadcast, or
   Directed. The target side will use the same broadcast type. In the
   case of Directed frame, if the RIF information is known, the target
   peer can send a directed frame. If not, a Single Route Broadcast
   frame is sent.

3.4.6. CAP_XCHANGE Frame

   In DCAP, the capability exchange frame is used to exchange the
   capability information between a client and a server. CAP_XCHANGE
   frames are exchanged between a client and a server as soon as the TCP
   session is established. The capability exchange must be completed
   before the other frame types can be sent. Once the capability
   exchange is done, CAP_XCHANGE frame should not be used again.

CAP_XCHANGE frame contains the clients MAC address, if a client has
one. If it does not, then the MAC address field must be set to zero.
When the DCAP server receives the CAP_XCHANGE frame, it should cache
the MAC address if it is non zero. The DCAP server also verifies that
the MAC address is unique. The server should return a CAP_XCHANGE
response frame with the MAC address supplied by the client if the MAC
address is accepted. If a client does not have its own MAC address,
the server should assign a MAC address to the client and put that
address in the CAP_XCHANGE command frame.

A client should record the new MAC address assigned by the server and
return a response with the assigned MAC address. If the client cannot
accept the assigned MAC address, another CAP_XCHANGE command with the
MAC address field set to zero should be sent to the server. The
server should allocate a new MAC address for this client.

During the capability exchange, both the client and the server can
send command frames. The process stops when either side sends a
CAP_XCHANGE response frame. When the response frame is sent, the MAC
address in the CAP_XCHANGE frame should be the same as the one in the
previous received command. The sender of the CAP_XCHANGE response
agrees to use the MAC address defined in the previous command.

The number of CAP_XCHANGE frames that need to be exchanged is
determined by the client and the server independently. When the
number of exchange frames has exceeded the pre-defined number set by
either the server or the client, the session should be brought down.

The flag is used to show the capability of the sender. The following
list shows the valid flags:

0x01 NetBIOS support. If a client sets this bit on, the server will
     pass all NetBIOS explorers to this client. If this bit is not
     set, only SNA traffic will be sent to this client.

0x02 TCP Listen Mode support. If a client supports TCP listen mode,
     the server will keep the client's MAC and IP addresses even
     after the TCP session is down. The cached information will be
     used for server to connect out. If a client does not support
     TCP listen mode, the cache will be deleted as soon as the TCP
     session is down.

0x04 Command/Response. If this bit is set, it is a command,
     otherwise, it is a response.

The values 0x01 and 0x02 are used only by the client. When a server
sends the command/response to a client, the server does not return
these values.

Starting with the Reserved field, implementers can optionally
implement the Capability Exchange Control Vector. Each Capability
Exchange Control Vector consists of three fields: Length (1 byte),
Type (1 byte), and Data (Length - 2 bytes). Two types of Control
Vectors are defined: SAP_LIST and VENDOR_CODE (described below). To
ensure compatibility, implementers should ignore the unknown Control
Vectors instead of treating them as errors.

0x01 SAP_LIST. Length: 2+n bytes, where n ranges from 1 to 16.
     This control vector lists the SAPs that the client can support.
     The maximum number of SAPs a client can define is 16. Therefore,
     the length of this Control Vector ranges from 3 to 18. If the
     SAP_LIST is not specified in the capability exchange, the server
     assumes that the client can support all the SAP values. For
     example, if a client can only support SAP 4 and 8, then the
     following Control Vectors should be sent: "0x04, 0x01, 0x04,
     0x08". The first byte indicates the length of 4. The second byte
     indicates the control vector type of SAP_LIST. The last two
     bytes indicate the supported SAP values; 0x04 and 0x08. This
     Control Vector is used only by the client. If the server accepts
     this Control Vector, it must return the same Control Vector to
     the client.

0x02 VENDOR_CODE. Length: 3 bytes.
     Each vendor is assigned a vendor code that identifies the
     vendor. This Control Vector does not require a response.

After the receiver responds to a Control Vector, if the capability
exchange is not done, the sender does not have to send the same
Control Vector again.

```
        +---------------+-----------------------+
        | Field Name    | Information           |
        +---------------+-----------------------+
        | Message Type  | 0x12                  |
        +---------------+-----------------------+
        | Packet Length | 0x1C                  |
        +---------------+-----------------------+
            Figure 3-14. CAP_XCHANGE Header
```

```
          +----------------------------------+
          | Field Name (Each row is one byte) |
          +==================================+
        0 | MAC Address                      |
          + - - - - - - - - - - - - - - - - +
        1 |                                  |
          + - - - - - - - - - - - - - - - - +
        2 |                                  |
          + - - - - - - - - - - - - - - - - +
        3 |                                  |
          + - - - - - - - - - - - - - - - - +
        4 |                                  |
          + - - - - - - - - - - - - - - - - +
        5 |                                  |
          +----------------------------------+
        6 | Flag                             |
          +----------------------------------+
        7 | Reserved                         |
          +----------------------------------+
            Figure 3-15. CAP_XCHANGE Data Format
```

3.4.7. CLOSE_PEER_REQ Frames

   This frame is used for peer connection management and contains a
   reason code field. The following list describes the valid reason
   codes:

   0x01 System shutdown. This indicates shutdown in progress.

   0x02 Suspend. This code is used when there is no traffic between the
        server and the client, and the server or the client wishes to
        suspend the TCP session. When the TCP session is suspended, all
        circuits should remain intact. The TCP session should be re-
        established when new user data needs to be sent. When the TCP
        session is re-established, there is no need to send the
        CAP_XCHANGE frame again.

   0x03 No MAC address available. This code is sent by the server when
        there is no MAC address is available from the MAC address pool.

```
          +---------------+----------------------+
          | Field Name    | Information          |
          +---------------+----------------------+
          | Message Type  | 0x13                 |
          +---------------+----------------------+
          | Packet Length | 0x08                 |
          +---------------+----------------------+
             Figure 3-16. CLOSE_PEER_REQ Header
```

```
         +-----------------------------------+
         | Field Name (Each row is one byte) |
         +===================================+
       0 | Reason Code                       |
         +-----------------------------------+
       1 | Reserved                          |
         + - - - - - - - - - - - - - - - - - +
       2 |                                   |
         + - - - - - - - - - - - - - - - - - +
       3 |                                   |
         +-----------------------------------+
            Figure 3-17. CLOSE_PEER_REQ Data Format
```

3.4.8. CLOSE_PEER_RSP, PEER_TEST_REQ, and PEER_TEST_RSP Frames

   These three frames are used for peer connection management. There is
   no data associated with them.

   o CLOSE_PEER_RSP
     CLOSE_PEER_RSP is the response for CLOSE_PEER_REQ.

   o PEER_TEST_REQ and PEER_TEST_RSP
     PEER_TEST_REQ and PEER_TEST_RSP are used for peer level keepalive.
     Implementing PEER_TEST_REQ is optional, but PEER_TEST_RSP must be
     implemented to respond to the PEER_TEST_REQ frame. When a
     PEER_TEST_REQ frame is sent to the remote station, the sender
     expects to receive the PEER_TEST_RSP frame in a predefined time
     interval (the recommended value is 60 seconds). If the
     PEER_TEST_RSP frame is not received in the predefined time
     interval, the sender can send the PEER_TEST_REQ frame again. If a
     predefined number of PEER_TEST_REQ frames is sent to the remote
     station, but no PEER_TEST_RSP frame is received (the recommended
     number is 3), the sender should close the TCP session with this
     remote station and terminate all associated circuits.

```
         +---------------+----------------------+
         | Field Name    | Information          |
         +---------------+----------------------+
         | Message Type  | 0x14, 0x1D, or 0x1E  |
         +---------------+----------------------+
         | Packet Length | 0x04                 |
         +---------------+----------------------+
```
    Figure 3-18. CLOSE_PEER_RSP, PEER_TEST_REQ, and PEER_TEST_RSP DCAP

4. Protocol Flow Diagram

   The following diagram shows a normal session start up/tear down
   sequence between a client and a server.

```
                        +-----------+                       +-------+
        +-----------+  Token   | DLSw/DCAP |                 | DCAP  |
        | Mainframe +- Ring ---+   Router  +-- ip backbone--+ Client|
        +-----------+          +-----------+                 +-------+

                                           TCP Session Up
                                           <-------------
                                           CAP_EXCHANGE (cmd)
                                           <-------------
                                           CAP_EXCHANGE (cmd)
                                           ------------->
                                           CAP_EXCHANGE (rsp)
                                           ------------->
                 TEST(P)                   CAN_U_REACH
              <--------                     <-------------
                 TEST(F)                   I_CAN_REACH
              -------->                     ------------->
                                           START_DL
                                           <-------------
                                           DL_STARTED
                                           ------------->
                 XID(P)                    XID_FRAME
              <--------                     <-------------
                 XID(F)                    XID_FRAME
              -------->                     ------------->
                 XID(P)                    XID_FRAME
              <--------                     <-------------
                 SABME                     CONTACT_STN
              -------->                     ------------->
                  UA                       STN_CONTACTED
              <--------                     <-------------
                I FRAME                    INFO_FRAME
              <--------                     <-------------
                I FRAME                    INFO_FRAME
              -------->                     ------------->
                 DISC                      HALT_DL
              <--------                     <-------------
                  UA                       DL_HALTED
              -------->                     -------------->
                                           CLOSE_PEER_REQ
                                           <-------------
                                           CLOSE_PEER_RSP
                                           ------------->
                                           TCP session down
                                           <-------------
```

5. Acknowledgments

    The authors wish to express thanks to Rodger Erickson of Wall Data,
    Inc. for his helpful comments and suggestions.

6. References

    [1] AIW DLSw Related Interest Group, RFC 1795,
        "DLSw: Switch-to-Switch Protocol", April 1995

    [2] IBM Token Ring Network Architecture Reference
        SC30-3374-02, September 1989.

    [3] IBM LAN Technical Reference IEEE 802.2 and NETBIOS Application
        Program Interfaces SC30-3587-00, December 1993.

    [4] ISO 8802-2/IEEE Std 802.1D International Standard.

Authors' Addresses

    Steve T. Chiang
    InterWorks Business Unit
    Cisco Systems, Inc.
    170 Tasman Drive
    San Jose, CA 95134
    Phone: (408) 526-5189
    EMail: schiang@cisco.com

    Joseph S. Lee
    InterWorks Business Unit
    Cisco Systems, Inc.
    170 Tasman Drive
    San Jose, CA 95134
    Phone: (408) 526-5232
    EMail: jolee@cisco.com

    Hideaki Yasuda
    System Product Center
    Network Products Department
    Network Software Products Section B
    Mitsubishi Electric Corp.
    Information Systems Engineering Center
    325, Kamimachiya Kamakura Kanagawa 247, Japan
    Phone: +81-467-47-2120
    EMail: yasuda@eme068.cow.melco.co.jp