6226

16 March 1984

CONTROL DATA

DISTRIBUTED COMMUNICATION NETWORK

EXTERNAL

REFERENCE

SPECIFICATION

/0078y

INSTRUCTIONS

Because all our documentation is being reproduced on reduction, it is essential that this form is either TYPED OR PRINTED LEGIBLY IN BLACK INK.

Fill out only those fields marked by bullets (o).

DOCUMENT TYPE: One of the following:
- AR - Archictecture Requirments (CYBER 80)
- ESL - Baseline Documentation Change(s)
- DAP - Design Action Paper
- DDD - Design Direction Document
- DR - Design Requirements
- EPP - Evalutaion Project Plan
- ERS - External Reference Specifications
- FTP - Feature Test Plan
- GDS - General Design Spec (CYBER 80)
- GID - General Internal Specifications
- IMB - Installation Handbook
- IPP - Implementation Project Plan
- PDA - Product Development Assurance Plan
- PJP - Project Plan
- STD - Standards Document
- STP - System Test Plan
- TIR - Transmittal, Integration & Release Criteria
- OTH - Other, specify type

TITLE: Be concise. Be sure to include the name of the product involved.

ABSTRACT: A brief description of the contents. If the item is extremely short, it may be written in designated space with nothing else attached.

PUBLICATION #: If a document with a pub. # is being changed, write the pub. # here.

SIP & PRM Numbers: If any are associated with this document, list them here.

APPROVALS: (Minimum DCS Requirements)
Project leader for all.
Unit Manager for all.
Section Manager: Project Plans only.
DCS will assign referee.

PROJECT: The project name that the author belongs to, e.g., CLUE.

DCS Mailing List Pasted Here -
(On top of instructions)

---

*For ARHOPS Use Only:

CONTROL DATA CORPORATION

DOCUMENT CONTROL FORM

o DOCUMENT TYPE: ERS

o LOG ID: ARH67

o TITLE: CONTROL DATA DISTRIBUTED COMMUNICATIONS NETWORK

o ABSTRACT: DESCRIBES COMPONENTS & FUNCTIONS OF THE CDCNET (aka DCN)

o PRODUCT AFFECTED: DCN RELEASE 1

o AUTHOR: L.L. LUCAS / B.S. SEKHON / M.C. STEELE

o MAIL STATION: ARH207

o EXTENSION: 6007/6047

o PROJECT:

o SECTION: DESIGN ASSURANCE.

o RSM or PSR #:

o SIP Number(s):

o PUB #:

o PRM #:

o Redesign* o Reimplementatic

Name	Approval Initials	Date
o Internal Reviewer*		
o Project Leader		
o Unit Manager		
o Section Manager	[signature]	3/16/84
Design Team Referee	[signature]	3/16/84

DESIGN TEAM: DEFAULT CYCLE: 20 WORKING DAYS

o Special Distribution: D.E. Stahl

Referee's Distribution Codes: AC-O

NUMBER	DATE	TYPE	SUBMITTED BY	DEFAULT APPROVAL DATE	CURRENT STATUS: REVIEW	HOLD	APPROVED	WITHDRAWN
1	3-21-84	Orig.	Lucas/Sekhon/Steele	4-23-84	R			
2.	3-24-84	APP.	DCS	4-23-84			A	

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

1.0 INTRODUCTION

The Control Data Distributed Communications Network (CDCNET or DCN for short)(*) is an implementation of the Control Data Network Architecture (CDNA). CDNA is a layered architecture based upon ISO's Open Systems Interconnection (OSI) reference model. An overview of both the OSI model and CDNA appears in the CDNA General Design Specification (ARH4243); a condensed version of that overview will be supplied here in a future update. It will only be noted here that the standardization efforts on the OSI model and on the supporting protocols are not yet complete; CDNA, and therefore the DCN implementation, currently lead ISO standardization efforts in some areas. CDNA will track standardization efforts as they progress and CDNA implementations, such as DCN, will be upgraded according to marketing and customer requirements.

This document describes the Distributed Communication Network (DCN) hardware and software as they will appear in the first release. Release 1 supports interconnection of C170 mainframes running C170 NOS and Network Products; asynchronous terminal support via the DCN is also provided.

DCN hardware is known collectively as the Device Interface (DI). DCN software that runs on the DI is known as the Distributed Communications Network Software (DCNS). DCNS is concerned with moving data across the network and managing the network.

Release 1 also includes a set of C170-resident software that is concerned with the definition and analysis of the DCN, as well as providing mass storage and host console access for DCNS.

This document is primarily intended to collect in a single place all information related to the external design of the DCN; this measure will facilitate a review of the overall DCN design and should also serve to identify design holes in external interface definitions. This document is also intended to provide input to the Publications and Graphics Division for use in preparing customer documentation.

(*) This product family has been recently renamed so there is still documentation in existence that refers to the DCN by its former name, ACN.

1.1 INITIAL GLOSSARY

An understanding of most large computer-related efforts involve learning a new vocabulary consisting largely of acronyms, and the Distributed Communications Network (DCN) is no exception. In general, new terms and acronyms will be defined in context in individual sections; however, an initial glossary is provided to give the reader at least a running start.

Network Solution	A term used to describe a single instance of a communications medium; e.g. an Ethernet cable is a "network solution;" so is an X.25 Public Data Network.
Catenet	A term used to describe a group of multiple interconnected "network solutions" and systems, which collectively form a "concatenated network."
Network	A context-dependent term that may mean either "network solution" or "catenet."
CDNA	Control Data Network Architecture - the definition of the network architecture to which future Control Data communications products are to adhere. The term "network" in this context is really a catenet.
DCN	Distributed Communications Network - The name of the first product implementation of CDNA; DCN includes both hardware and software. The term "network" in this context is really a catenet.
DI	Device Interface - the name of the hardware used in the DCN product.
DCNS	Distributed Communications Network Software - the name of the software used in the DCN product. DCNS runs in the DI. The term "network" in this context is really a catenet.

1.2 DOCUMENT ORGANIZATION

The remainder of Section 1 briefly describes the organization/content of this ERS and provides a list of references for further reading.

Section 2.0 of this ERS provides an overview of the hardware used to implement the DCN. The hardware is known generically as the Device Interface (DI). The basic DI can be configured into a number of variants serving specific purposes within the DCN, e.g., Mainframe Device Interface (MDI), Terminal Device Interface (TDI), Network Device Interface (NDI), and others to be defined for subsequent releases.

Section 3.0 describes the major components of the DCN Software, aka DCNS. DCNS operates in the DI hardware and is composed of the following major elements:

- System Management Software - responsible for creation and maintenance of the system environment for the other functions. Creation of the system environment is provided by the Network Load & Startup software, which is responsible for dumping/loading/reloading the DI hardware as needed and for putting the network into an operational state. Other system management software provides the environment during normal operation of the network.
- Layer Software - responsible for transmission of data for the end-users of the DCN.
- Network Management Software - responsible for providing network management functions.

The DCN software provides for the following three major external interfaces:

- C170 Interface
- X.25 Interface
- Terminal Interface

An overview of each of the major interface components is given in last three subsections of Section 3; each is described in more detail in Sections 4.0, 5.0, and 6.0, respectively.

1.2 DOCUMENT ORGANIZATION (continued)

The remaining sections of this ERS, summarized below, describe various aspects of DCN; additional introductory remarks will be added as drafts of the remaining sections become available.

- Section 7.0 describes the process of network validation;
- Section 8.0 defines Titles and Addresses;
- Section 9.0 describes the process of defining/configuring a DCN;
- Section 10.0 describes the operation of a DCN, primarily from the Network Operator's perspective;
- Section 11.0 describes failure management; in what manner can the network fail and what actions are taken to lessen/eliminate the impact of failure;
- Section 12.0 describes the accounting facilities provided for the DCN;
- Section 13.0 describes analysis and tuning of the DCN. This section also provides a description of the facilities provided to handle dumps.
- Section 14.0 describes DCN performance expected under various loads.
- Section 15.0 defines finite state machines.

ERS appendices include:

- a summary list of all commands
- a summary list of all log messages
- file conventions and formats
- utilities
- scenarios; e.g., typical installation process, typical log-in process, etc.
- network initialization flow

1.3 REFERENCES

1.3.1 Product References

Additional information on the DCN product family as a whole may be found in the following documents:

Network Architectural Objectives	ARH5011
ACN Design Requirements	ARH5123
CDNA General Design Specifications	ARH4243
Device Interface GDS	ARH4948
C170/CDNA Interface Specification	S4234
ACN Maintenance Software ERS	ARH5176

1.3.2 Hardware References

Additional hardware information may be obtained from the following documents:

DI Equipment Specification	53984774
CIM Equipment Specification	67328072
MCI Equipment Specification	67328070
RS449 Equipment Specification	67328074
RS232 Equipment Specification	67328073
ESCI Equipment Specification	67328071
PMM Equipment Specification	67328069
SMM Equipment Specification	21935466
MC68000 16-bit Microprocessor User Manual	

1.3.3 Software References

DCNS can be logically divided into the major component groups listed below, each of which is discussed in its own subsection. A bibliography of existing documentation for each component group is provided below under the component group heading.

SYSTEM MANAGEMENT SOFTWARE

DI Load & Startup Software

System Ancestor ERS IDS	ARH5377 (*)
Initial Loader ERS	ARH5377
Main/Initialization Bootstrap ERS IDS	ARH5377 (*)
Software Load Process ERS	ARH5377
Configuration Procurer ERS	ARH5377
On-line Loader ERS	(*)
Initialization M-E ERS	ARH5377

Operational Software

Executive ERS IDS	ARH4976 ARH5704
Common Subroutines Handbook	(*)

(*) Not submitted to DCS; no number available.

1.3.3 Software References (continued)

LAYER SOFTWARE

Lower Layers

Generic 3A ERS	(*)
Ethernet/Firmware/SSR ERS	(*)
HDLC SSR ERS	(*)
CIM Controlware ERS	(*)
Device Manager (DVM) ERS	(*)

Middle Layers

Internet (Xerox) ERS	(*)
Transport Generic Xerox ERS	(*)

Higher Layers

ITS	(*)

(*) Not submitted to DCS; no number available.

1.3.3 Software References (continued)

NETWORK MANAGEMENT FUNCTIONS

Routing M-E ERS	(*)
Directory M-E ERS	(*)
IDS	(*)
Dependent Log M-E (LSA) ERS	(*)
Independent Log M-E ERS	(*)
Error/Echo M-E ERS	ARH6065
File Access M-E ERS	ARH5960
IDS	(*)
Dependent Command M-E ERS	ARH5451
Independent Command M-E (OSA) ERS	(*)
Status Command Processor DAP	ARH6014
ERS	(*)
Statistics Command Processor/M-E DAP	(*)
ERS	(*)
Initialization M-E ERS	ARH5377

(*) Not submitted to DCS; no number available.

1.3.3 Software References (continued)

C170 INTERFACE

MDI-resident software

C170 Support Software DAP	(*)
C170 Gateway ERS	(*)
BIP ERS	ARH5378
MFI SVM ERS	ARH5376
K-Display Support ERS	ARH5968

C170-resident software

C170/CDNA Console Access Helper Application	S4406
DCN Operator Facility GID	S4500
DCN File Utility GID	S4566
DCN File Server Application ERS	S4507
DCN File Server Application GID	(*)
DCN Log Server Application DAP	(*)
DCN Log Server Application ERS	(*)

(*) Not submitted to DCS; no number available.

1.3.3 Software References (continued)

X.25 INTERFACE

X.25 Support Layer ERS	ARH5678
X.25 Packet Level ERS	(*)
X.25 Gateway ERS	(*)

TERMINAL INTERFACE

DCN Terminal Support DAP	ARH5627
LCM/TDSM ERS	(*)
Asynch TIP ERS	(*)

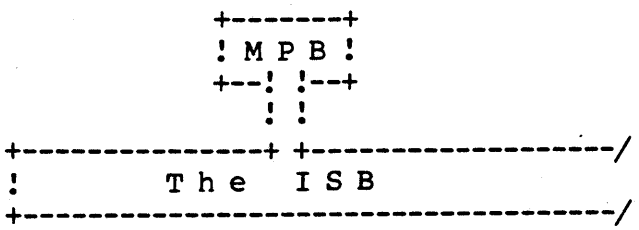
(*) Not submitted to DCS; no number available.

2.0 DI HARDWARE OVERVIEW

The Device Interface (DI) is a highly modular, microprocessor-based device. DI architecture is based on a multi-master bus and shared resources.

The basic DI is diagrammed below and consists of:

- One Master Processor Board (MPB)
- An Internal System Bus (ISB) that can connect up to eight cards, one of which is the MPB
- Room for up to eight Line Interface Modules (LIMs).
- Power supplies, cabinetry and cooling facilities



2.0 DI HARDWARE OVERVIEW (continued)

Because of the modularity of both the hardware and software, a DI can be configured to support a wide variety of functionality. For the first release, the following DI variants have been defined and named according to their function:

MDI — Mainframe Device Interface, which connects the DCN to a C170 mainframe. The source or destination of the data transferred to/from the C170 is dependent upon other modules configured within the MDI.

NDI — Network Device Interface, which acts as a packet switcher to transfer data between network solutions. Depending upon the modules configured, a particular NDI may interface to any or all of the following:

- Public Data Network (PDN) - X.25 Type
- Local Area Network (LAN) - Ethernet
- Communications Line(s) - Point-to-Point Links

TDI — Terminal Device Interface, which is used to interface terminal devices to the DCN. Future releases may allow a TDI to interface to high-speed line(s) and/or a public data network to function as a "remote concentrator".

2.0 DI HARDWARE OVERVIEW (continued)

The DI variants are created by configuring the appropriate combination of cards from the list below. These cards are known as "peripheral cards;" peripheral cards that contain microprocessors are known as "intelligent peripherals."

- Mainframe Channel Interface (MCI)
- Ethernet Serial Channel Interface (ESCI)
- Communications Interface Module (CIM)
- Private Memory Module (PMM)
- System Main Memory (SMM)

The MPB provides the main processing power for the DI. The MPB is shielded from most of the real-time external interrupts to allow parallel processing and high performance; for example, the CIM card handles all interrupts from communication lines, while the ESCI card handles all interrupts from Ethernet. The MPB is therefore free to process the next (or previous) message while the peripheral cards are busy transmitting the previous (or receiving the next) message.

Normally, all cards share access to the system bus and to other system resources (e.g. SMM). However, the MPB can selectively prevent one or more cards from accessing the bus, thereby allowing isolation of a bad card from the rest of the DI without physically removing the card from the DI.

Each card in the DI has a certain amount of Read Only Memory (ROM); the ROM is used to keep card identification (e.g. version number) and self-test (on-board diagnostics) code. In addition, ROM on some cards contains boot code to facilitate loading of the DI across that specific card. All ROMs include a 16-bit checksum to facilitate detection of memory failures.

All memories and data paths in the DI have a certain amount of built-in error detection capability. The system main memory (SMM) contains logic for single bit error correction/double bit error detection (SECDED). All Random Access Memory (RAM) on individual cards contains a parity bit for each byte; parity is checked on all read operations. The data bus portions of the system bus, 68000 extension bus, and the LIM bus contain one parity bit for every eight bits of data.

2.1 INTERNAL SYSTEM BUS

The Internal System Bus (ISB) is used for all communications between different cards; the ISB consists of the following two parts:

Internal Transfer Bus (ITB) - The ITB is used for data transfer between individual cards as well as for the bus contention management. It consists of a 24-bit address bus and a 16-bit data bus. It supports a 16-megabyte address range and a minimum transfer rate of 2.5 megawords-per-second across the bus; each word contains 16 bits.

The ITB is a multi-master bus in the sense that it may be owned by any one of the eight cards at a given point in time. The contention logic ensures that all cards have an equal opportunity to gain access to the ITB.

Internal Control Bus (ICB) -The Internal Control Bus (ICB) is owned by the MPB and is used for control activities such as status and interrupts. The ICB contains an 8-bit data bus and a 4-bit address bus.

The address bus allows the selection of up to sixteen cards, even though only eight cards are supported at present.

The data bus allows the MPB to send a command byte to a selected card as well as to receive (read) a Status byte from a selected card. Each data bus access can be qualified in four different ways to facilitate support of up to four command and four status bytes. One of the command bytes is used to enable/disable access to the ITB from a specific card.

The ICB is also used to reset a card from the MPB.

2.2 INTERRUPTS

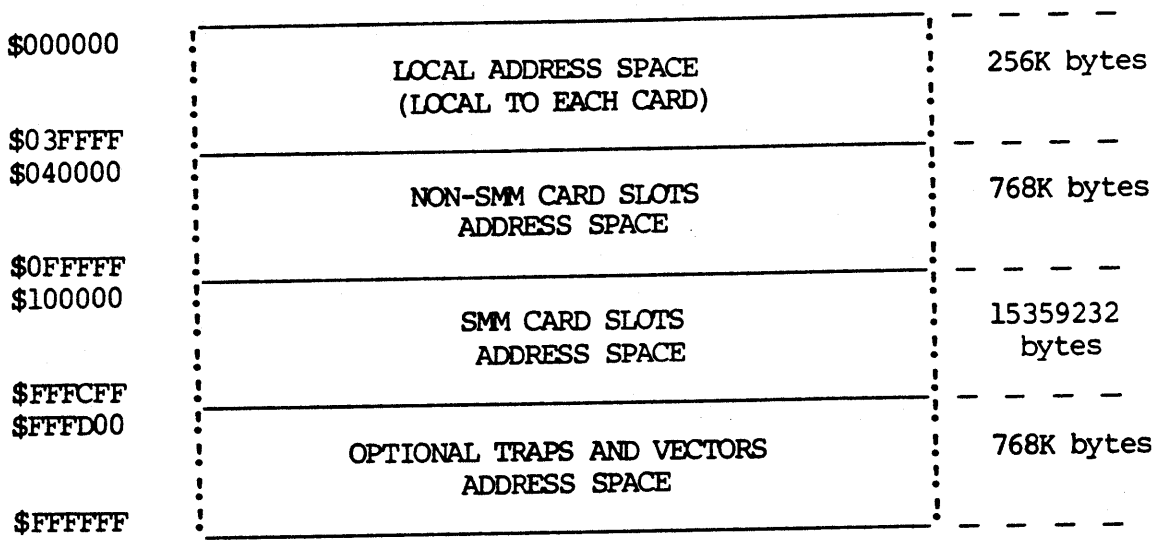
The MPB can send two types of interrupt to an intelligent card (e.g. CIM, ESCI). One type of interrupt is non-maskable and is used for re-starting the card; the second type of interrupt is maskable. Both types are invoked by sending a command byte to the card via the ICB bus.

All cards (intelligent or otherwise) can send an interrupt to the MPB card. These interrupts are scanned by the MPB in the following way:

> The MPB maintains a hardware counter which increments from 0 to 7 (for an 8-card backpanel) or from 0 to 15 (for a 16-card backpanel).
>
> Every time this counter is incremented, the card whose slot number equals the current value of the counter is checked for an outstanding interrupt.
>
> If one is present, the hardware stops incrementing the counter and initiates interrupt processing. At the end of the interrupt processing, the hardware continues incrementing the counter.

2.3 DI ADDRESS STRUCTURE

The DI address structure provides a 16M-byte address space. The lower 256K of this address space is local to each card in the sense that it is used for the on-card memory as well as for addressing the local devices. This address space is not accessible via the ISB.



Address space from $040000 to $0FFFFF makes up twelve 64K-byte segments. This address space is called the non-SMM card slots address space and is used to allow direct access from the MPB to peripheral devices (e.g. DMA chip) on other cards. Each 64K-byte segment will be associated with a specific card slot during installation. Note that, at present, only MCI card allows this type of access.

The address space from $100000 to $FFFD00 is reserved for the SMM cards. This address space is accessible via the ISB to all cards connected to the ISB.

The 768K bytes at the top of the address space are reserved for the interrupt and trap vectors for the cards and devices that require these vectors to reside at high addresses.

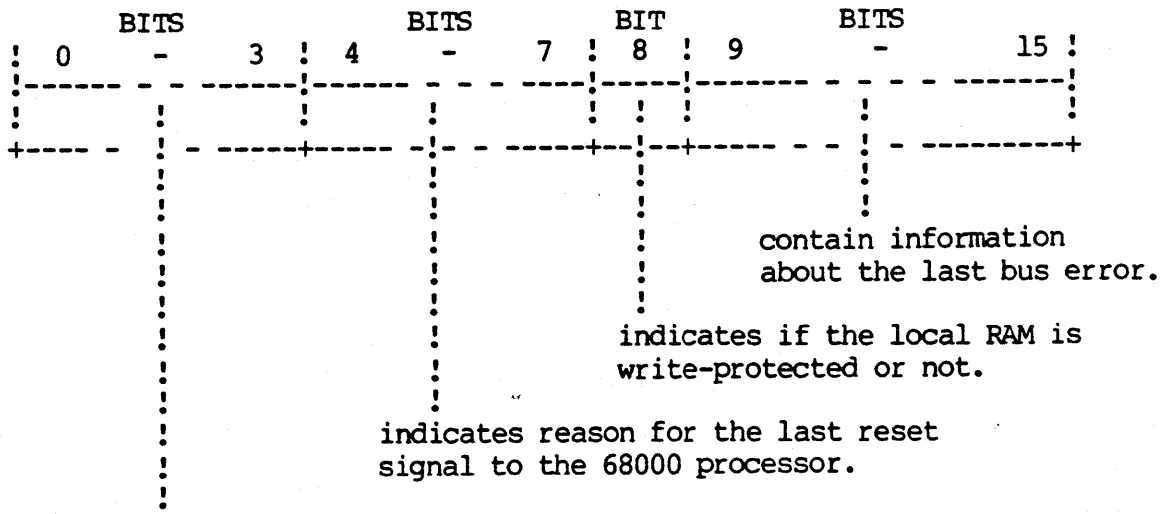
2.4 MAIN PROCESSOR BOARD

The MPB provides the primary processing power for the DI software. It includes one Motorola 10-MHZ 68000 microprocessor, 16K bytes of RAM and 16K bytes of ROM. 4K bytes of the RAM are battery-backed. The MPB also contains a 68000 extension bus, which allows the MPB to access up to 128K bytes of RAM on a Private Memory Module (PMM) card. The MPB's on-board RAM, as well as PMM memory, is provided to support fast memory access. Read and write operations to these memories occur with no 68000 "wait states". In addition, the MPB's on-board RAM can be dynamically write-protected via I/O instruction; however, the write protection is valid only if the 68000 microprocessor is in the user state.

The MPB contains a 32-MHZ oscillator, which is used to provide a 16-MHZ clock for the ISB. The MPB also contains a real-time clock and calendar, which are battery-backed. The MPB also contains a Serial Communications Controller (SCC) device, which provides two RS232 ports; the baud rate for these ports is programmable between 300 and 9600 baud.

The MPB includes a deadman timer which needs to be reset within nine seconds by software writing to a specific address ($00610A). If a deadman time out occurs, halt and reset signals are sent to the 68000 processor, which results in a DI reset.

The MPB contains a software-readable 16-bit status register, illustrated below.



2.5 MAIN PROCESSOR BOARD (continued)

The MPB supports an "interrupt vector table" in memory locations $00000 to $0003FF, which points to interrupt handler routines. The interrupt vector table resides in the MPB ROM, following a power-on or manual reset; the interrupt handler routines to which it points are rudimentary and contain only the code needed during DI loading. During normal network operation, the interrupt handler routines need additional sophistication. The MPB therefore contains logic to support alteration of the interrupt vector table. An I/O instruction is provided to interchange ROM with RAM; specifically, RAM locations ($008000 to $00BFFF) are interchanged with ROM locations ($00000 to $003FFF). This allows dynamic changes to the interrupt vector table by re-writing the appropriate RAM location.

The 68000 microprocessor supports seven interrupt levels, which are used on the MPB board in the following way:

Level 7 — The level 7 interrupt is generated as a result of the following two conditions:

AC low (power failure) indication

Temperature shut down (overheated)

Software is expected to read the MPB status register to determine which of above two conditions is responsible for the interrupt.

Level 6 — The level 6 interrupt is generated by the SCC device

Level 5 — The level 5 interrupt is generated by the 68000 extension bus.

Level 4 — The level 4 interrupt results from the scanned ISB interrupts and therefore can be from any one of the other cards.

Level 3 — The level 3 interrupt is generated by a device called CIO. This device is responsible for monitoring the various external switches on the MPB board.

Level 2 — The level 2 interrupt is generated by the Real Time Clock (RTC).

Level 1 — The level 1 interrupt is, at present, not used.

2.4 MAIN PROCESSOR BOARD (continued)

The MPB 68000 microprocessor hardware provides a set of commands to the software to function the MPB hardware. These commands are executed by reading from/writing to address $0061XX, where the value of XX depends upon the specific command to be executed. The following is a list of these commands, which are described in detail in the MPB Engineering Specification:

- Read MPB status register
- Set and clear RAM write protect
- Set bus lock
- Read bus error address register
- Set and clear diagnostics mode
- Reset deadman timer
- Set and clear swap of ROM and RAM
- Set external indicators
- Read ITB loop back data

2.5 COMMUNICATIONS INTERFACE MODULE (CIM)

The CIM is the controller for up to eight (8) Line Interface Modules (LIM). Each CIM contains a 10-MHZ MC68000 16-bit microprocessor with a minimum of 16K bytes of RAM with byte parity and zero wait state operation. A minimum of 8K bytes of ROM with checksum is included for on-board diagnostics and initialization code.

The CIM has two interfaces:

- the LIM Bus interface to communicate with the LIMs
- the ISB interface to communicate with the rest of the DI

The LIM Bus is an 8-bit data bus that can connect the CIM to a maximum of eight LIMs. LIMs are numbered according to their physical position in the LIM backpanel; the LIM closest to the outside is LIM 7.

A "daisy chain" method is used to determine the specific source of an interrupt on the LIM Bus. The lowest numbered LIM has the highest priority. The CIM contains gating to bypass each LIM and substantially reduce the amount of time required for the disable signal to ripple thorugh successive LIMs. The bypass scheme on the CIM also allows for an empty slot in the middle of the daisy chain, which allows LIMs to be removed without the need to reconfigure the system's interrupts.

The LIM Bus is protected with a single parity bit. Occurrence of a parity error causes a bus error .

I/O control logic on the CIM generates properly timed control signals required by the CIO and the peripheral chips on the LIM. It also generates the necessary handshake signals that are used by the processor control circuitry.

The CIM drives the LIM Bus using information provided by the MPB.

The ICB is used for receiving commands from the MPB and to allow the MPB to read hardware and software status from the CIM. The CIM communicates with the SMM over the ITB. The MPB communicates with the CIM over the ICB.

Parity bits (upper and lower) are generated for all write operations regardless of the destination of the data, and without regard to byte or word operations. Parity bits may or may not be used depending on the destination of the data. Parity is checked, on a byte-basis, during read operations from the ITB, the LIM Bus, and the CIM RAM.

2.6 ETHERNET SERIAL CHANNEL INTERFACE (ESCI)

The ESCI transports data between SMM and one Ethernet 10 Mbit Serial Channel. The ESCI card contains:

- Intel 82586 Ethernet Controller chip (the Controller);
- Intel 82501 Serial Interface chip
- Motorola MC68000 microprocessor (the Processor)
- 16K bytes of RAM
- 16K bytes of ROM

The on-board ROMs contain initialization bootstrap code, diagnostics, and checksum.

The Controller has priority access to both local memory and SMM. Extra on-card bus buffering is used to permit the Processor to access local memory while the Controller accesses SMM.

The ESCI is connected to the Ethernet cable transceiver and to the DI's Internal System Bus (ISB). Communication with the DI is primarily through data structures in SMM.

Three kinds of Ethernet data loopback are available for use by diagnostics:

- local loopback on the Ethernet controller
- intermediate loopback on the Intel 82501 Ethernet Serial Interface
- external loopback on the Ethernet cable

Because Motorola and Intel differ in their handling of memory stacks, the Processor and Controller store data differently on byte and 24-bit long-word operations; 16-bit operations are handled in the same manner.

On byte operations, the Controller stores/loads the first byte in/from bits 0-7 when starting on an even address; the Processor stores/loads the first byte in/from bits 8-15 under the same conditions. Transfers between the Controller and the ITB mask this difference by swapping the two bytes across this interface; ITB bits 0-7 = controller bits 8-15 and bits 8-15 = controller bits 0-7 on all direct transfers between the Controller and ITB devices. Byte-swapping does not occur on transfers to local memory or between the Processor and the Controller.

On long-word operations, the Processor stores/loads the two most significant bytes in/from the first location and the two least significant bytes in/from the following location; the Controller stores/loads the two least significant bytes in/from the first location and the two most significant bytes in/from the next location. If only receive-data and transmit-data is transferred directly between the ITB and the Controller, the byte-swap performed by this interface resolves any addressing conflicts. Addressing conflicts between the Processsor and the Controller must be resolved by on-board software.

2.7 MAINFRAME CHANNEL INTERFACE (MCI)

The MCI, which operates under MPB control, will transport data between a C170 12-bit channel and SMM. A chained DMA capability and on-board ROM containing bootstrap code, diagnostics and checksum is provided. Two data packing and unpacking modes will be supported with channel and bus parity, bit and byte. Only one MCI per channel will be supported. Pass-on/pass-back will not be supported. Up to three MCI's per MDI will be supported.

2.8 PRIVATE MEMORY MODULE (PMM)

The PMM provides additional zero-wait-state RAM for use by the MPB via the 68000 extension bus. Use of the PMM reduces the number of instruction accesses across the ISB and increases throughput of the MPB. The PMM can be configured in any of the DI variants.

The PMM contains either a 64K-byte memory or a 128K-byte memory; both have the same artwork and differ in the number of memory chips assembled. The smaller PMM is not field-upgradable.

Maximum transfer rate for a read cycle is 140 nanoseconds. Since the memory is static, there is no constraint on minimum transfer rate.

Maximum transfer rate for a write cycle is 180 nanoseconds.

If a data parity error occurs on a write operation, the PMM sends a bus error signal; however, the incorrect data is written into the memory.

2.9 SYSTEM MAIN MEMORY (SMM)

The SMM has a minimum of 512K bytes of RAM with internal refresh, Single Error Correction/Double Error Detection (SEC/DED), and error reporting to the MPB. A diagnostic mode on the SEC/DED chip and programmable start address are also features.

ITB interface parity errors, SEC/DED multiple errors, and (if enabled) SEC/DED single bit errors are recorded on the SMM error log. The error log is a 16-bit register that contains information about the type and location of error(s). When an error is recorded in the error log, an ICB interrupt is generated and the log is locked. The error log logs only one error at a time; new errors can be recorded only after the log is unlocked by a Master Clear, Reset, or by reading the log. The error log is read by an ITB diagnostic mode read when the ICB Mode Control is off. Contents of the error logs are unknown on power-on.

An Am2960 EDC chip is used for error detection and correction. The EDC contains the logic necessary to generate check bits on a 16-bit data word according to a modified Hamming code. The EDC will correct an single-bit error and will detect all double and some triple-bit errors on data-read operations.

If the SMM detects a single bit error on a read cycle, it will correct the word, generate a new check code and write the corrected data word/check code to the address specified by the read cycle.

Maximum transfer rate during a read cycle is 312.5 nanoseconds (625 ns for read with corrected error). Minimum read access time is 312.5 ns (437.5 for read with corrected error).

Maximum transfer rate for a write cycle is 312.5 nanoseconds (625 ns for byte write.).

2.10 LINE INTERFACE MODULES (LIM)

Multiple versions of this board will be developed. LIMs have their own space within the DI and up to eight (8) can be configured within a single DI. Each LIM interfaces to the LIM Bus and can drive external lines.

A LIM can be connected to only one CIM at a time. It can have from two to four ports, each of which connects to a communications line or a unit record device.

All LIMs employ VLSI line controllers with the following programmable features:

- Baud Rate/Channel/Receive or Transmit
- External/Internal Clock
- Synchronous (Bit or Byte) and Async
- Auto Echo and Loop Back
- Interrupt (Vectors, Enable)
- Modem Control Response
- CRC-16 or CCITT CRC Generation
- Mark or Space Idle
- Odd or even line parity

The first release will include the LIM types described below.

2.10 LINE INTERFACE MODULES (continued)

2-Channel RS449 LIM

This is a two-channel device which provides, as standard, two DTE connections that comply with all the requirements and options for the RS449 primary channel. Both RS422 and RS423 electrical standards are supported via a strappable option.

FEATURES SUPPORTED	COMMENTS
RS449 PRIMARY CHANNEL DTE + RS422	STANDARD
RS449 PRIMARY CHANNEL DTE RS423	STANDARD WITH STRAPS
RS449 PRIMARY CHANNEL DCE + RS422	REQUIRES MODEM ELIMINATOR
RS449 PRIMARY CHANNEL DCE + RS423	AS FOR (3) WITH STRAPS
RS449 SECONDARY OR BACKWARD CHANNEL, DTE, RS423	CABLE ADAPTOR 9 PIN TO 37 PIN
RS449 PRIMARY CHANNEL WITH (SDCD) FLOW CONTROL. DTE, RS422	CABLE ADAPTOR 37 PIN AND 9 PIN TO 37 PIN.PSEUDO STANDARD ONLY
6 WITH RS423 OPTION	AS PER 6 WITH RS423 STRAPS
RS366A AUTO DIAL OPTION	REQUIRES SOFTWARE AND ADAPTOR (REF. 3.1.1.7)
X.24/X.21 PARTY LINE	REQUIRES ADAPTOR, STRAPS AND SOFTWARE

Each of the features listed require one of the two available channels on this LIM.

- In asynchronous mode this LIM will support both data rates of 50 to 38.4Kbps per channel, and different receive and transmit rates (e.g. send 1200 baud, receive 75 baud).

- In synchronous mode this LIM will support up to 64Kbps with a local clock; with an external clock, up to 250Kbps can be supported by the hardware. As a synchronous DCE, this LIM will supply a transmit clock.

4-Channel RS449 LIM

This LIM will provide both RS232/V.28 and RS422/V.11 electrical interfaces. The physical interface will be compatible with RS232/V.24 (via an adaptor cable if necessary), with X.21, and with X.24. Allowance will be made for both DTE/DCE operation and both local or external clock sources.

2.10 LINE INTERFACE MODULES (continued)

4-Channel RS232 LIM

RS232 will support per channel all those DTE signals defined in "RS232 application notes option D" VIZ:

CCIT CIRCUIT	RS232 CIRCUIT	NAME
101	AA	Ground
102	AB	Logic RTN
103	BA	Transmit Data
104	BB	Receive Data
105	CA	Requst to Send
106	CB	Clear to Send
107	CC	Data Set Ready
108.2	CD	Data Terminal Ready
125	CE	Ring Indicator
109	CF	Carrier Detect
113	DA	Transmit Clock (DTE)
114	DB	Transmit Clock (DCE)
115	DD	Receive Clock (DCE)

The RS232 implementation will allow different transmit and receive rates as used by V.23 style modems.

Signalling rates supported will range from 50 bps to 38.4Kbps asynchronous and 56Kbps synchronous. With external clocking, or using non standard baud rates, speeds up to 250Kbps can be supported. The higher baud rates will limit the cabling (as wave shaping will be used).

4-Channel X.24/X.21 LIM

This will support the following as a DTE.

X.24 CIRCUIT	NAME
G	Ground
T	Transmit Data
R	Receive Data
C	Control
I	Indication
S	Signalling Clock

The electrical level will be RS422/V.11. All of the above will be supported as either Data Communication Equipment (DCE) or Data Terminal Equipment (DTE).

2.10 LINE INTERFACE MODULES (continued)

V.35 LIM

This 2-Channel LIM is based on the RS449 LIM with appropriate changes to the electrical and physical interfaces to comply with CCITT V.35 + APPENDIX II, and ISO 2593-1973(E) for all CCITT-specified circuits.

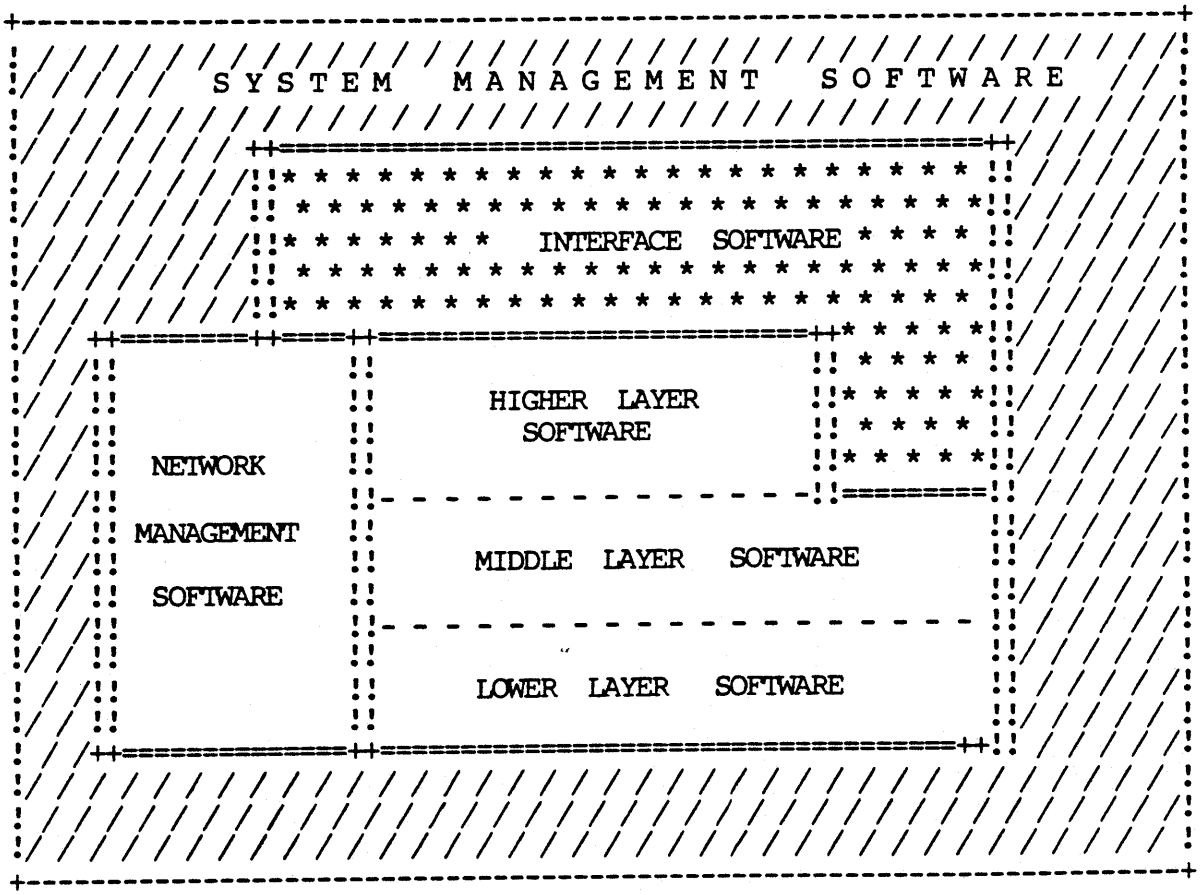
Signalling rates up to 108KHz will be supported with external (DCE) supplied clock.

3.0 DISTRIBUTED COMMUNICATIONS NETWORK SOFTWARE (DCNS)

DCN software (DCNS) resident in the DIs provides four major functions:

- System management software - implementation of software to create and maintain a system environment for the other functions.
- CDNA layer software - implementation of the Control Data Network Architecture (CDNA) layer functions.
- CDNA network management software - implementation of the Control Data Network Architecture (CDNA) network management functions.
- Interface software - implementation of interface software to connect foreign networks, systems, and devices to the DCN.

The diagram belows illustrates the relationship between the major DCN software functions.



3.0 DISTRIBUTED COMMUNICATIONS NETWORK SOFTWARE (DCNS) (continued)

System management software operates initially to create an operational environment in a DI. Software resident in the ROMs operates in conjunction with the Initialization M-E to load a DI; the following components then put the DI into an operational state:

- Initial Loader
- System Ancestor
- Configuration Procurer

Section 3.1 describes the network load & startup process.

Once the network is operational, system management software maintains the environment for the layer and network management functions; operational system management components are:

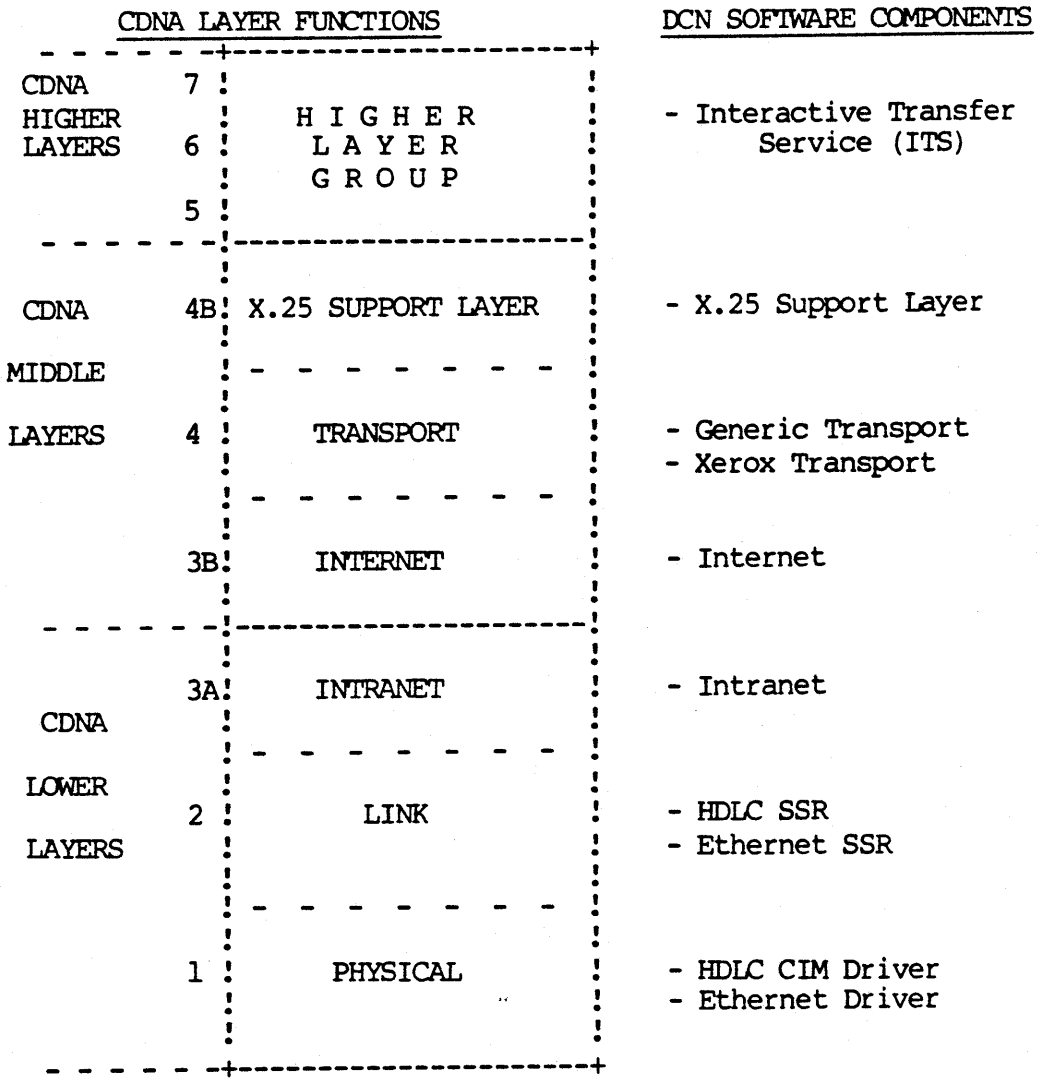
- Executive
- Common Subroutines
- On-line Loader
- Diagnostic Management Services
- Failure Management

Every DI contains all of the operational system management software.

Section 3.2 provides an overview of the operational system management software components.

3.0 DISTRIBUTED COMMUNICATIONS NETWORK SOFTWARE (DCNS) (continued)

CDNA layer software provides communication between systems, terminals, applications, and end users connected to the DCN. DCN implements one or more software components for each layer to provide the functions required by CDNA for that layer, as illustrated below.



Every DI contains software components for layers 1 through 4B. HDLC and/or Ethernet components are present for layers 1 and 2. These layers are necessary in order to support network management functions.

DCN layer components are discussed in Section 3.3.

3.0 DISTRIBUTED COMMUNICATIONS NETWORK SOFTWARE (DCNS) (continued)

CDNA network management is provided by software components called Management-entities (M-Es). Management-entities are the following:

- Routing M-E — handles routing of data within the catenet.
- Directory M-E — maintains a directory of Titles and Addresses.
- File Access M-E — provides access to secondary storage files.
- Command M-E — processes network commands.
- Log M-E — provides logging of messages.
- Echo M-E — allows data to be "echoed" back to the sender.
- Error M-E — provides for communication of error conditions.
- Initialization M-E — operates as part of the DI load & startup process.

Every DI contains all or some portion of the above M-Es. Whether or not a given DI contains the full-blown version of a particular M-E depends on whether that DI is configured to support both an active ("independent") and a passive ("dependent") role in network management or just a passive ("dependent") role. Generally, DIs that have access to secondary storage are the best candidates to be configured in support of active network management roles.

Section 3.4 provides an overview of each Management-entity and further elaborates upon the distinction between active and passive roles.

3.0 DISTRIBUTED COMMUNICATIONS NETWORK SOFTWARE (DCNS) (continued)

Any system that contains at least the CDNA layer and network management functions defined above is said to be a "CDNA system." In Release 1, each DI is therefore a CDNA system; the C170 mainframe is not a CDNA system because it does not contain any of the CDNA layers or management functions. In contrast, the C180 mainframe, which will be supported in a subsequent release, does contain CDNA layers and management functions and is therefore a CDNA system.

CDNA implementations, such as DCN, are expected to facilitate interconnections with non-CDNA systems; DCN provides this capability via the DCN interface software.

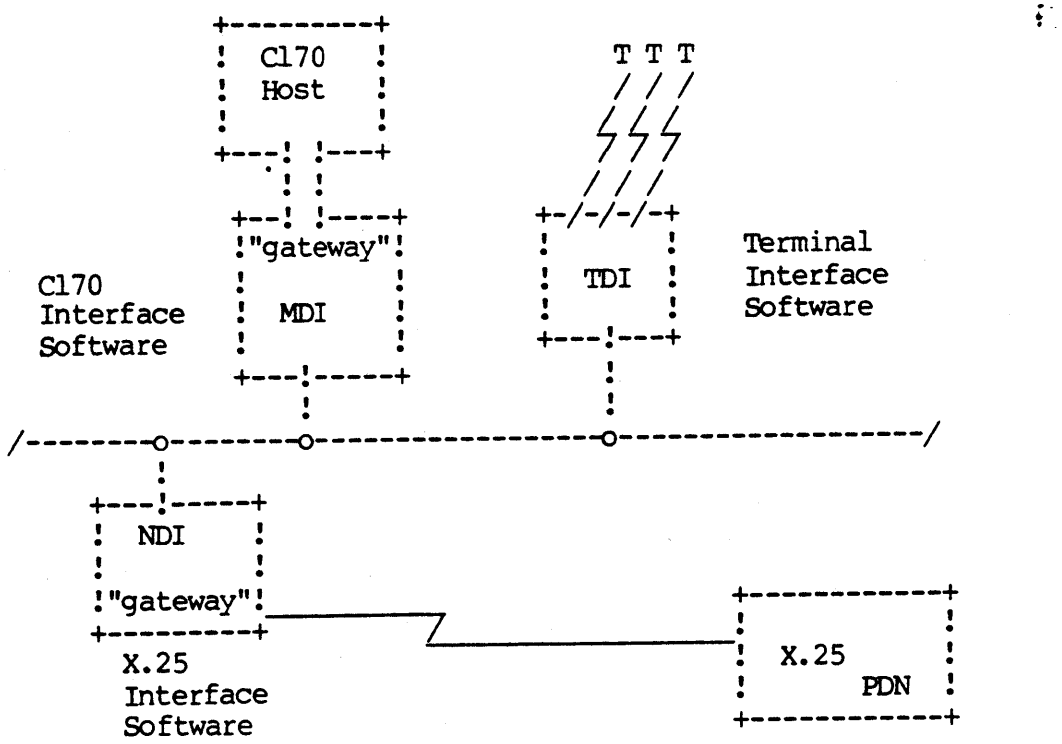
DCN interface software supports three major external interfaces, as follows:

C170 interface	allows DCN end users to access services of a C170 host connected to the DCN. C170 interface software is also used to access C170 host resources for use by the DCN network management functions. Section 3.5 provides an overview of this interface.
X.25 interface	provides a mechanism whereby two or more physically disjoint DCN networks can be logically connected via an X.25 PDN or X.25 trunk. This same interface provides, to foreign systems and networks, a transparent access to the DCN. Section 3.6 provides an overview of this interface.
Terminal user interface	allows DCN end users to access the services of the DCN via terminal devices connected via asynchronous communication lines. Subsequent releases will also support synchronous communication lines. Section 3.7 provides an overview of this interface.

The C170 and X.25 interface software both include a "gateway function" to permit communication between unlike architectures. DCN implements the "gateway function" by software components called "transforms" that provide a mapping between the services offered by DCN and those offered by the other architectures. Gateway functions and the associated transforms are further discussed in Sections 3.5 and 3.6.

3.0 DISTRIBUTED COMMUNICATIONS NETWORK SOFTWARE (DCNS) (continued)

The diagram below illustrates the three major external interfaces, including the gateway functions.



The specific functional role of a DI determines which DCN interface software components are resident in that DI.

If the role of a DI is to interface a C170 mainframe to the DCN, the DI will contain C170 interface software and the DI is called a Mainframe Device Interface (MDI).

If the role of the DI is to interface terminals to the DCN, the DI will contain terminal interface software. Such a DI is called a Terminal Device Interface (TDI).

If the role of a DI is to interface an X.25 network to the DCN, the DI will contain the X.25 interface software and the DI is called a Network Device Interface (NDI). DIs that interconnect HDLC and Ethernet networks do not require any specific non-CDNA interface software; these DIs are also called NDIs.

Because of the modularity of the DI hardware and software design, DIs can be configured to support more than one type of function. For example, a DI can be configured with both the terminal and mainframe support hardware/software. The role of such a DI becomes Mainframe and Terminal Interface (MTI). Other varieties of hybrid DIs are possible.

3.1 SYSTEM MANAGEMENT OF DI LOAD & STARTUP

This section describes the DCNS components involved in bringing the DI hardware to an operational state. Components are introduced and described in the order in which they operate during network loading and startup.

In Release 1, a C170 mainframe is the initial source of load files and is the only destination for dump files; consequently the load process begins with interaction between a C170 mainframe and a C170 MDI that is connected via a channel interface.

A program called INITMDI in the C170 mainframe is activated as the result of a request from the MDI; INITMDI is responsible for executing the Initialization protocol with the MDI across the channel. Further information on the process of loading across the channel will be supplied in a future update to this document; the remainder of this section assumes that at least one MDI is loaded and ready to load other DIs in the catenet.

In the descriptions that follow, the DI to be loaded is referred to as the "remote DI," while the DI performing the load (and optionally receiving the dump) is referred to as the "local DI." A local DI can load a remote DI only through two types of network solution, namely the Ethernet and an HDLC line. Since the remote DI initially contains only the code that is burned into its ROMs, loading over an X.25 packet network is impractical.

Each remote DI must receive its load information from a neighbor that is already loaded and operational. A remote DI sends a "Help Request" to all of its neighbors on one of the network solutions to which it is connected; each neighbor either ignores the "Help Request" or acknowledges by sending a "Help Offer." If more than one "Help Offer" is received, the remote DI accepts help from a single neighbor by sending a "Help Accept." A "Help Offer" contains a priority field that will, in subsequent releases, be used to determine which neighbor should be selected.

If the DI to be loaded does not receive a "Help Offer" from at least one neighbor, it sends another "Help Request" to all of its neighbors on another network solution to which it is connected. This process is repeated until a "Help Offer" is received.

3.1 SYSTEM MANAGEMENT OF DI LOAD & STARTUP (continued)

Each DI contains software in read-only memory (ROM) to provide the functions needed for initial self-test and loading of the DI; these functions are invoked after a power failure or a reset due to any other failure, such as hardware and software failures.

The software contained in ROM consists of two major components:

The ROM On-Board Diagnostics, which:

- Check the ROM memory for integrity;
- Determine which cards are physically present in the DI and builds the hardware configuration table, also known as the DI Card Map; and
- Test the individual hardware cards.

The ROM-based Initialization Software, which:

- Locates possible load sources (e.g connected networks) and requests help in loading the DI from the selected source, and
- Provides an optional dump of the DI, with the assistance of the selected source.

3.1.1 ROM On-Board Diagnostics

The ROM on-board diagnostics are self-test diagnostics that check the DI hardware for correct operation. They are run every time the card is reset. On-board diagnostics exist on the following cards:

- o MPB
- o MCI
- o ESCI
- o CIM
- o SMM

On-board diagnostics on the MPB board are responsible for testing the MPB and PMM boards. In addition, they are responsible for coordination of on-board diagnostics on the other cards. The MPB on-board diagnostics are responsible for construction of the DI Card Map, which contains a list and status of cards physically present in the DI; it also determines the total SMM address space based on the SMM cards present and operational in the DI.

On-board diagnostics on the other cards are initiated by the MPB on-board diagnostic based on information from well-known ROM locations on the board under test. They test the hardware present on the card and provide its status to the MPB on-board diagnostics.

3.1.2 ROM-based Initialization Software

The ROM-based initialization software consists of the following two parts:

MPB ROM Bootstrap

The MPB ROM Bootstrap in the remote DI is responsible for coordination of the bootstrap load process and execution of the CDNA Initialization protocol with either a peer Initialization M-E in the local DI or INITMDI in the C170 host. The MPB ROM Bootstrap includes the following major components:

Main Bootstrap Controller
Initialization Bootstrap

Card-specific Bootstrap

Card-specific Bootstraps exist in ROM on the following cards:

CIM
ESCI
MCI

A Card-specific Bootstrap is called by the Main Bootstrap Controller as a subroutine and, in turn, calls the Initialization Bootstrap software to help in loading the DI. The Card-specific Bootstrap provides the layers 1, 2, and 3A functions for the network solution connected to the card. In the case of the CIM Card-specific Bootstrap, up to two connected HDLC lines are tried, one at a time, according to switch settings.

The reader might note that the MPB ROM Bootstrap is the DCNS implementation of the dependent Initialization M-E in the remote DI, as described in the CDNA GDS. The Initialization M-E in the fully loaded local DI functions as the independent Initialization M-E.

3.1.2.1 Main Bootstrap Controller

The Main Bootstrap Controller is invoked by MPB on-board diagnostics to read the "boot enable" switch on the MPB card to determine the first card to be used for the bootstrap load. If the selected card is present and operatiional (as determined from the DI Card map), the Card-specific Bootstrap from the card's ROM is moved to SMM and an attempt is made to load the DI across this card by using the Card-specific Bootstrap as a subroutine.

If the attempt is unsuccessful, other cards that are potential bootstrap sources are selected and the load attempt is retried. Cards are selected in numerical card slot order.

If all attempts fail, the Main Bootstrap Controller displays an error code via the light indicators on the MPB Card and halts the processor. This results in a Dead man time out, which re-starts the whole process by starting the on-board diagnostics.

3.1.2.2 Initialization Bootstrap

The Initialization Bootstrap is called as a subroutine from the Card-specific Bootstrap; it broadcasts a Help Request on the network solution connected to the card across which the load is being attempted. It waits for a certain amount of time for possible Help Offers; selects one based on the priority included in the Help Offer; and sends a Help Accept to the selected system.

The Initialization Bootstrap uses the Auto Dump table to determine if the DI memory is to be dumped before loading the DI. The Auto Dump table resides at a well known address and is protected via a checksum. The table is built by the operational software present in DI prior to this load. It contains a list of portions of DI memory to be dumped. If the Auto Dump table does not exist or is invalid, a default Auto Dump consisting of all of SMM and the MPB RAM is sent; RAM on other card types is not included in the default Auto Dump.

The Initialization Bootstrap calls subroutines in the Card-specific Bootstrap to dump the DI memory and then load the DI. It executes the CDNA Initialization protocol to do so.

3.1.3 Initialization M-E

Each MDI and NDI that is loaded and operational, contains a copy of the independent Initialization Management-entity (M-E), which is used to dump and load other DIs.

The independent Initialization M-E in the local DI uses the File Access M-E to read the load file from a C170 mainframe and executes the Initialization protocol with the MPB ROM Bootstrap (dependent Initialization M-E) in a remote DI in order to load the remote DI. The Initialization M-E uses lower layers (layers 1-3A) to send and receive data to and from the remote DI.

Lower layers associated with Ethernet do not have to do anything special to effect loading of a remote DI. Lower layers associated with the HDLC lines have to prime the line for the Initialization protocol by using "RIM" and "SIM."

The following is a step-by-step description of the process used to load a remote DI, from the local DI's perspective.

1. The network solution connecting the local and remote DIs is configured in the local DI via a configuration command. As a result, the lower layer software is initialized for data transfer on the network solution.

2. The Initialization M-E in the local DI informs its lower layers of its willingness to receive data from any connected network solution. Information in the protocol header of the incoming data is used to determine if the data is addressed to the Initialization M-E.

3. The Initialization Bootstrap in the remote DI sends a Help Request, which is received by the Initialization M-E in the local DI.

4. The Initialization M-E uses the File Access M-E to read a file that contains information about DIs that need a non-default version of the load file as well as a list of DIs that should not be loaded. This file is further described in the section on Network Definition.

   If no restriction is found on loading that remote DI, the Initialization M-E sends a Help Offer. The Help Offer contains a priority field which, in Release 1, can contain only a single value; future use of this field remains to be determined.

5. The Initialization Bootstrap in the remote DI decides if it wants to accept the Help Offer. If so, it sends a Help Accept to the Initialization M-E in the local DI.

3.1.3 Initialization M-E (continued)

6. After receipt of the Help Accept, the Initialization M-E in the local DI uses information in the original Help Request to determine the need to dump the memory of the remote DI; the criterion used in this determination is the reason for loading the remote box.

7. If the remote DI memory needs to be dumped, the Initialization M-E uses the File Access M-E to create a new dump file on an accessible C170 mainframe. An Initiate Auto Dump is sent to the remote DI, which responds by sending dump data.

   The Initialization M-E in the local DI and the Initialization Bootstrap in the remote DI follow the Initialization protocol to dump the memory of the remote DI. The Initialization protocol allows for partial and full dump. Throughout this process, the Initialization M-E uses the File Access M-E to write dump data to the dump file.

8. Once the dump is complete or if the dump is not needed, the Initialization M-E uses the File Access M-E to find the appropriate load file on an accessible C170 mainframe. It uses the File Access M-E to read the load file and the lower layers to transmit the information to the remote DI. The Initialization M-E and the Initialization Bootstrap use the Initialization protocol to manage transmission of load data.

3.1.4 Startup

This section provides an overview of the three software components that execute immediately after the DCN software has been loaded into DI memory. These three components bring the loaded software to a fully operational state. The three processes are the Initial Loader, System Ancestor, and Configuration Procurer and execute successively in this order.

3.1.4.1 Initial Loader

The Initial Loader is the first software loaded into the DI by the Initialization Bootstrap. The Initial Loader arrives as a prelinked MC68000 absolute record ready for execution. The Initial Loader is given control by the Initialization Bootstrap after all software has been placed into DI memory.

The first function of the Initial Loader is to absolutize the loaded software modules. This function includes:

- allocating memory for the modules,
- moving code and data into the allocated memory, and
- linking external references.

The second function involves the initialization of the Executive, which includes:

- initializing the system vector and configuration tables,
- creating buffer and memory pools, and
- setting up hardware timers.

The Initial Loader then starts the System Ancestor as a task. This process starts both the Executive and the System Ancestor. At this point, all software is ready for startup; the Executive is initialized and running; and the System Ancestor has been started as a task.

3.1.4.2 System Ancestor

The System Ancestor's function, during initialization, is to start the tasks associated with a subset of the loaded software. This subset consists of the software common to all DI variants plus the software required to drive the network interface that was used as the load source.

The System Ancestor makes successive calls to the Initial Loader to obtain information about the next task to be started. This information includes the task's transfer address, stack size, and priority. This information is given to the Executive for each task as it is started. Each time the System Ancestor calls the Initial Loader, it determines if the task returned is the System Ancestor itself. If so, this informs the System Ancestor that its function is complete with respect to DI initialization. The System Ancestor also plays a role in the management of failures; this role is discussed in the next section.

The Initial Loader works off a list of loaded software, called the Module Name List. The System Ancestor task's position within the Module Name List differentiates tasks started directly by the System Ancestor from tasks started through some other process such as configuration commands. Thus, some tasks are started as the result of the initialization process; others are started as the result of the configuration process; and yet others are started as the result of some other stimulus.

When the System Ancestor completes its initialization function, the minimum set of system tasks have been started.

3.1.4.3 Configuration Procurer

The Configuration Procurer uses the services of the File Access M-E to read a file containing the configuration commands specific to the DI being initialized. The Configuration Procurer issues an Open File request that specifies the type of configuration file (via the "file-type Title" parameter) and the System ID of the system being configured (via the "file name" parameter).

The Configuration Procurer submits the commands to the Command M-E for processing by the responsible command processors. In this way, the set of software and tasks that supports the DI's unique configuration are started.

The Configuration Procurer receives command responses for each command submitted and issues log messages for those commands that were not processed to normal completion. Commands in error do not prevent execution of subsequent commands. Once a DI is operational, configuration command errors can be corrected manually via the operator interface.

The execution of the Configuration Procurer can be bypassed by creating a configuration file containing a "Bypass Configuration" command; all DI configuration information can then be entered manually by an operator.

3.2 SYSTEM MANAGEMENT

The components described in this section provide support for the layer and network management functions during normal network operation.

DCNS provides system management functions, as follows:

Executive -	provides the functions needed in a multiprocessing/multitasking software environment.
Common Subroutines -	provide commonly used functions and CYBIL interfaces to Executive subroutines.
On-line Loader -	used to load and remove software modules in an operational on-line environment.
Diagnostics Management -	provides the framework for execution of on-line diagnostics
Failure Management -	provides detection, logging, and recovery of hardware and software failures

3.2.1 Executive

The DCNS Executive provides, to all processes, the kernel set of primitive functions required in a multiprocessing/multitasking software environment. These functions include such services as:

- CPU scheduling
- exception processing and interrupt vector services
- intertask communication
- memory and buffer management
- timer services

Statistics, journaling, debug, and queueing services are also provided, although they are not at present used.

Code for the Executive can reside within the MPB RAM, SMM, or PMM if available. The Executive executes on the Main Processor Board (MPB) of the DI.

The MPB's MC68000 microprocessor is always in one of three processing states:

normal	associated with instruction execution; the memory references are to fetch instructions and operands, and to store results.
exception	associated with interrupts, trap instructions, tracing and other exceptional conditions.
halted	an indication of catastrophic hardware; only an external reset can restart a halted processor.

3.2.1 Executive (continued)

The processor operates in one of two states of "privilege:"

supervisor — the higher state of privilege. All instructions can be executed in the supervisor state; all exception processing is done in the supervisor state.

user — the lower state of privilege. Some instructions which have important system effects are privileged and cannot be executed in user state. Once the processor is in the user state, only exception processing can change the privilege state.

The Executive is responsible for the allocation of MPB MC68000 processor and system memory between two classes of software processes:

Exception processors - Exception processors are high-priority predefined code sequences that are given CPU control by the hardware when an exception is sensed by the CPU. A hardware interrupt is an example of an exception.

Tasks - Tasks provide network layer and management functions during that period when the CPU is not busy servicing exceptions.

The Executive provides interfaces for defining both types of processes. It is the Executive's responsibility to schedule the CPU between the tasks.

3.2.1.1 CPU Scheduling

Tasks are scheduled to the CPU based on their task priority. Tasks with the same priority are scheduled to the CPU in a round-robin fashion. The CPU scheduling algorithm is executed when the currently running task completes execution or when a higher priority task is waiting for the CPU. The Executive will recognize and schedule the higher priority task under the following conditions:

- the currently running task has exceeded its time slice;
- the currently running task makes a request for Executive services;
- an exception processor completes processing. In this case, the Executive schedules the higher priority task instead of rescheduling the task that was interrupted.

The Executive may temporarily reduce a task's priority as a result of executing the CPU scheduling algorithm. This is only done when it is determined that a task is using an inordinate amount of CPU time. In this case, a task's priority is reinstated at that time when the task has completely processed all of its queued intertask messages and has entered a waiting state.

Exception processors are given CPU control whenever the hardware senses an exception event. Exception events are prioritized by the hardware. This provides resolution of simultaneous exception events and also allows a lower priority exception processor to be interrupted by a higher priority exception processor. Details of MC68000 exception processing are described in the MC68000 16-bit Microprocessor User Manual.

CPU scheduling is currently undergoing design; some of the questions remaining to be answered are:

When the priority is lowered, is the task put at the head or tail of the queue containing tasks at the lowered priority?

What is an "inordinate" amount of time?

What happens if the task gives up the CPU before the ITM queue is empty?

3.2.1.2 Exception Processing

Exceptions can be generated either externally or internally. Externally generated exceptions are:

- reset requests
- interrupts from peripheral devices
- bus errors

Internally generated exceptions may arise as the result of execution of an instruction. The TRAP, TRAPV, CHK, and DIV instructions can generate an exception as part of their execution. Other instructions can generate exceptions:

- as the result of being an illegal instruction
- by attempting word fetches from odd addresses
- by attempting to execute a privileged instruction while the MC68000 processor is in the non-privileged state.

The Executive maintains an exception vector table that maps the exception vector number to the entry point of the corresponding exception processor. The vector table contains 255 entries and is initialized as part of the Executive initialization process.

Exceptions not known to the Executive at initialization time are mapped to the spurious interrupt processor; such exceptions are normally interrupts from peripheral devices that are not yet initiallized. When that device is undergoing initialization, an Executive service is provided to change the device's exception vector from the spurious interrupt to the appropriate peripheral device interrupt processor.

When peripheral devices interrupt the CPU, they present an exception vector number that is used as an index into the exception vector table so the appropriate interrupt exception processor can be executed. The Executive provides services for making entries into the exception vector table for purposes of servicing hardware interrupts and software-defined TRAP levels.

All exception processors execute in the MC68000 supervisor state and utilize the system stack.

3.2.1.3 Inter-Task Communication

The main function of tasks is to process intertask messages sent to them by exception processors or by other tasks. In support of this function, the Executive provides services for sending and receiving intertask messages. Each task has two associated intertask message queues that are maintained for the task by the Executive. Tasks execute in the MC68000 user state and utilize their own associated stack area.

The Executive maintains a list of Task Control Blocks. Each entry in the list contains control information for one of the currently defined tasks.

Associated with every task is a task stack and task priority.

The task stack is used to hold procedure automatic variables and to facilitate the linking/unlinking of nested procedure calls. The task stack size is set once at task startup time and remains unchanged for the life of the task. There is no mechanism to prevent a task from overflowing its stack space; there will, however, be methods for detection, isolation, recovery, and prevention of stack overflow.

The task priority is used by the Executive to determine the next task to be scheduled for the CPU. Tasks with higher priority are given first consideration. Task priority is specified at task startup time and can also be altered dynamically via an Executive service.

3.2.1.4 Buffer Management and Thresholds

Buffer Management is the Executive service that supports the dynamic allocation and deallocation of buffers used to hold message data. The design of the Executive buffer management service considers both system performance and the unique processing requirements of a layered architecture.

Every attempt has been made to minimize or eliminate movement of data within system memory. Such functions as adding or deleting message headers or trailers do not affect the residence of message data. Multiple copies of messages are maintained at the logical level by utilizing one physical copy and an associated usage count.

To support this level of buffer management sophistication, a descriptor buffer design was chosen. Associated with every data buffer is a descriptor buffer that keeps track of buffer chaining, usage counts, data offsets, data counts, and pointers to associated data buffers. Both descriptor and data buffer sizes are configurable.

The Executive provides services to obtain and release buffers. The services of the Executive buffer management are augmented by a set of common subroutines that perform the high frequency functions on buffers, such as addition/deletion of message headers/trailers, message fragmentation/assembly, logical/physical message copy, etc.

The Executive can be configured to support any number of buffer thresholds. Both the number of thresholds and the total number of buffers available at each threshold level are configurable.

The Executive uses the configured parameters while processing "buffer get" requests. The user process must specify a buffer threshold along with every "buffer get" request. The Executive will honor the request only if the number of currently available buffers is greater than the sum of the number of requested buffers and the configured threshold value.

Threshold level zero is predefined by the Executive; requests at threshold level zero are always honored. If enough buffers are not available to satisfy a threshold level zero request, buffer space is allocated from global memory until enough buffers exist to honor the request. All common subroutines that potentially need to obtain an Executive buffer, require a buffer threshold parameter.

Buffer threshold use is currently undergoing design; some of the questions to be answered are:

How is buffer threshold used?

An overview of memory management should be supplied.

3.2.1.5 Timer Services

The Executive provides a set of timer services that allow a process to request that a specified subroutine be called. In addition to specifying the subroutine to be called, the user can also optionally specify a parameter to be included with the subroutine call when the subroutine is activated.

The user can request that the specified subroutine be called:

- at a specified time with a time resolution of one minute
- periodically
- after an interval of elapsed time.

The service also exists to cancel a previously issued timer request.

All timers are accurate to one millisecond.

3.2.1.6 Non-buffer Memory Management

The Executive provides a service whereby a variable-length memory exent may be obtained by processes requiring non-message memory areas. Variable-length memory extents can be obtained directly through the use of an Executive-supported service or via the CYBIL ALLOCATE statement. The Executive internally utilizes global memory for allocating and deallocating memory space to and from the data buffer pool.

This feature may or may not be retained; investigation is currently underway.

3.2.2 Common Subroutines

The Common Subroutines are a collection of frequently used functions, which can be grouped into the following categories:

- CYBIL interface to Executive functions
  - Queue management
  - Task management
  - Timer services
  - Miscellaneous Executive functions, such as obtaining/releasing memory, start/stop of tools, etc.
  - Common Type definitions
- Non-Executive functions
  - Tree (table) management
  - Buffer management
  - Operations on integers

3.2.3 On-line Loader

The On-line Loader provides the following services to other DI tasks:

- Translate entry point names and module names into tasks/addresses
- Interlock use of modules to prevent deloading of modules while still in use
- Provide a status interface for communication between loaded modules and operator status enquiries.

On-line Loader functions are:

- Load — provides loading of modules by entry point name and module name. The load process consists of reading the object module file, allocating memory for the module sections, moving code and data into the sections, and linking external references.
- Deload — provides deloading of unused modules. The deload process consists of dereferencing other modules, deallocating memory for the module sections, and removing entry point names.
- Translation — provides the entry point name and module name translation to already loaded modules or modules to be loaded. It also provides translations to support the status interface, checksum all loaded modules, validate the program counter, and return transfer addresses.

3.2.4 Diagnostics Management Services

Diagnostics Management Services (DMS) co-ordinate on-line diagnostics and provide support for failure management software. DMS consists of the following components:

- DIAG_INIT_PROCESSOR — reviews the final status of on-board diagnostics.
- GEN_DIAG_COMMAND — builds a "send" command for the Attention Sequence of Auto-Initiate processors.
- MONITOR_ON_BOARDS — Starts and monitors an on-board diagnostic for a specific device while operating in an on-line environment.

On-line diagnostics are SMM-resident programs that test individual boards within a DI while the remainder of the DI remains operational. Running the diagnostic may temporarily cause the system to stop normal processing in cases where there is only one board of the type being diagnosed; however, the board's controlware is not disturbed by the diagnostic so that it can be returned intact to the system when the diagnostic completes.

On-line diagnostics are loaded on an as-needed basis to test both the DI hardware elements and their interfaces with external devices such as C170 mainframe channel, Ethernet Transceiver, modems, and communication lines. The following on-line diagnostics are planned for Release 1:

- CIMO - tests a CIM and its associated LIMs, via standard controlware resident in the CIM. CIMO uses loop-backs provided in the LIMs and attached modems to isolate problems in the communication lines.
- ECHO - tests the DI-to-DI link, across either Ethernet or communications line/modems. Testing consists mainly of echoing data from one DI to the other.
- ESCO - tests the ESCI board and the Ethernet Transceiver via the standard ESCI-resident controlware. ESCO uses loopbacks provided by the ESCI module and Ethernet Transceiver to isolate problems in the Ethernet.
- MCIO - tests the MCI board. Testing is done from the MPB, using the MPB-resident controlware. Testing simulates customer operation.
- PMMO - tests the PMM board; testing is done from the MPB.
- SMMO - tests the SMM board. Testing is done from the MPB.

3.2.5 Failure Management

Failure management software is concerned with the detection, logging, and recovery of DCN hardware and software failures.

Failure detection includes the following aspects:

1) How each failure is detected.

2) What information is available at the time of failure; for example, what software component was executing; what memory address was being accessed.

3) How severe the failure is; dependent upon the type of failure, recent occurrences of the same failure, and which software component experienced the failure.

Failure logging includes the following aspects:

1) which failures are reported

2) when failures are reported

3) which software components reports a failure and to whom

4) what information about a failure is saved and how

Failure recovery may include any of the following:

1) reloading the DI

2) restarting the DI without reloading

3) downgrading part of the hardware resources

4) removing/replacing a failing hardware component while the DI remains on-line

5) restarting a specific software component, with or without a reload

6) retrying the failing function

7) executing appropriate on-line diagnostics

Failure recovery must be preceded by failure isolation and saving of relevant information. Isolation of hardware failures is limited to identification of the failed card; isolation of software failures is limited to identification of the failed software component, where software component is defined to be an individually loadable piece of software.

3.2.5 Failure Management (continued)

Recovery procedures must take into account the probability of that failure occurring; for example, if an error is likely to occur very rarely, a reload or restart of the DI may be an acceptable recovery procedure. Recovery procedures must also be sensitive to the environment in which the failure occurred; for example, recovery procedures for a CIM board failure in an NDI may differ, depending on whether or not a back-up CIM board exists.

If the recovery procedure includes a hardware-reset of the total DI, the reason for the reset must be saved for use by the on-board diagnostics that run as part of the reset process; for example, if part of an SMM card is unusable, the on-board diagnostics must not include that part in its definition of the DI configuration.

If a failure is unrecoverable, other DIs connected to the failed DI must be able to detect the situation and initiate appropriate recovery action, such as operator notification.

DCN failure management is discussed in more detail in a subsequent section of this document.

3.3 LAYER SOFTWARE

The CDNA/DCN layer software provides the functions needed to provide communication between systems, terminals, applications and end users connected to the DCN. Since CDNA is based on the ISO's OSI model, its layer structure is very similar to the one proposed in the OSI model, but includes some enhancements and alterations.

This section introduces the layer groups and defines some terms that are necessary to understand the layers. Subsequent subsections deal with each layer group in greater detail.

The lower layer group is concerned with transmission of data across specific network solutions. This group consists of three layers, numbered 1, 2, and 3A:

The Physical layer (1) includes software to activate, maintain and deactivate interconnection with various media.

The Link layer (2) includes software to support synchronization, error control and flow control functions for data transfer between two or more DIs connected to the same network solution. Layers 1 and 2 are identical to layers 1 and 2 of the OSI model.

The Intranet layer (3A) provides a single network-solution-independent interface for users of the lower layers. It is responsible for routing and relays within a specific network solution. The layer 3A is a result of the sub-layering of the Network layer of the OSI model.

The middle layer group provides data-independent, end-to-end services and protocols. This group consists of the following:

The Internet layer (3B) provides a relay function between different network solutions to allow users connected to different network solutions to exchange information. The layer 3B is a result of the sub-layering of OSI layer 3 into Internet (3B) and Intranet (3A) layers.

The Transport layer (4) functions are identical to the OSI layer 4. The CDNA/DCN Transport functions are provided by two "sub-layers," called Xerox Transport and Generic Transport.

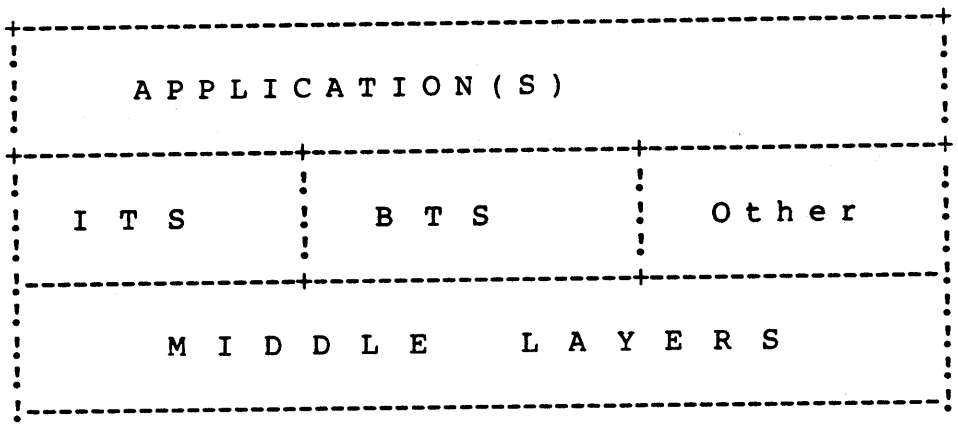
The X.25 Support layer (4B) provides a service equivalent to that provided by an X.25 network.

3.3 LAYER SOFTWARE (continued)

The higher layer group provides data-dependent, end-to-end services and protocols in direct support of applications. This group provides the functionality of the layers identified as Application, Presentation and Session layers in the OSI model, although it is not defined as three distinct layers.

CDNA/DCN defines a distinct higher layer group for each different type of application. In Release 1, the higher layer group provided is the Interactive Transfer Service (ITS); ITS supports the OSI Application, Presentation and Session layer functions for data transfer between two interactive applications and between an application and a terminal.

A separate higher layer group, Batch Transfer Service (BTS), will provide batch data transfer functions in a subsequent release.



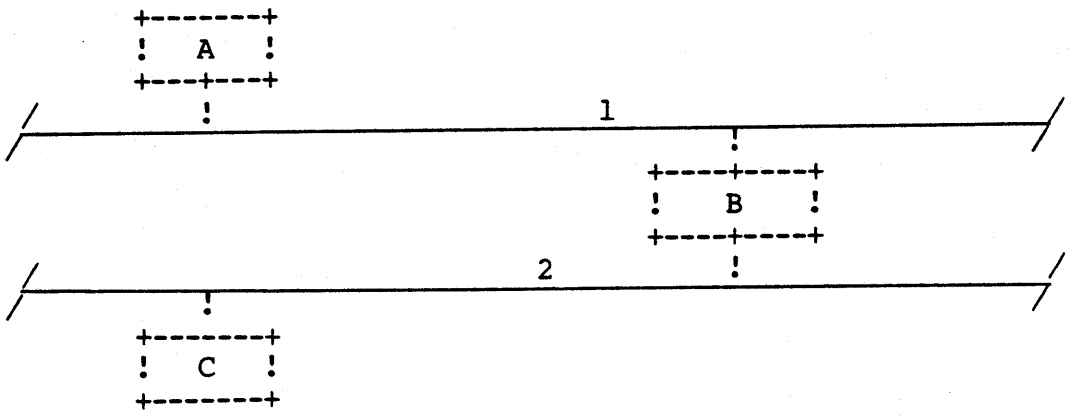
3.3 LAYER SOFTWARE (continued)

LAYERLAND GLOSSARY

DATA UNIT — A "data unit" is data that is transmitted, as a unit, between peer entities in different systems; the term is usually used synonymously with "protocol data unit (PDU)."

RELAY — The term "relay" is used to define the process in which a CDNA system receives a data unit from one locally connected network solution and transmits it on another locally connected network solution.

In the configuration pictured below, there are three CDNA systems, labelled A, B, and C, and two network solutions, numbered 1 and 2. System A is connected to network 1; system C is connected to network 2; and system B is connected to both networks 1 and 2.



Suppose system A wants to send a data unit to system C. System A will send it to system B on network solution 1. System B will receive it on network solution 1 and will determine that it is destined for system C. System B will then transmit the data unit on network solution 2. System C will receive it on network solution 2. The function performed by system B, in this example, is called a relay.

HOP — The term "hop" is associated with the term "relay." In the above example, one says that system C is one hop away from system A, etc. A hop count is maintained in the Internet header of each data unit and is incremented by the Internet layer each time a hop is made. Should the hop count ever exceed sixteen, the data unit is discarded. This use of the hop count is intended to prevent a data unit from wandering aimlessly through the DCN forever.

LAYERLAND GLOSSARY (continued)

LOCALLY CONNECTED NETWORK SOLUTION A network solution is said to be "locally connected" to a system if that system can directly transfer data across that network solution to another system on the same network solution. In the above example, systems A and C each have one locally connected network solution; system B has two locally connected network solutions.

Note that this term has been adopted for this document because of the potential confusion between "Distributed Communications Network (DCN)" and "directly connected network (DCN)." The latter term is used in the CDNA GDS which was written before the term "Distributed Communications Network" came into being.

REMOTELY CONNECTED NETWORK SOLUTION A network solution is said to be "remotely connected" to a system if that system is not locally connected to that network solution, but there exists some path whereby that system may transmit data to that network solution in one or more hops.

3.3 LAYER SOFTWARE (continued)

LAYERLAND GLOSSARY (continued)

INTERFACE METHODS Software modules within the same system can communicate with each other using any of the methods described below; software modules in different systems can communicate with each other only by use of a connection.

Call/return interface - within the same system, one module may "call" the other via the call/return interface; only the two modules are involved in this form of interface.

Inter Task Message interface - within the same system, a module may communicate with another module using the inter-task message interface provided by the Executive.

Connection - if modules reside in different systems, they can communicate with each other only by establishing a "connection." Depending on the expected duration and usage of a connection, one of the following connection modes is used:

Liaison-mode has discrete establishment, data transfer, and termination phases and is used when significant amounts of data are expected to be sent back and forth over some period of time.

Datagram mode is transaction-oriented; a datagram is a single data unit exchanged between two entities. There are no discrete establishment, data transfer, and termination phases; address information is included within each datagram.

Broadcast/multicast mode is an expansion of datagram mode in that it permits transmission of a datagram to more than one entity.

3.3 LAYER SOFTWARE (continued)

LAYERLAND GLOSSARY (continued)

NETWORK ADDRESS All externally addressable DCN software is identified via a three-part address that provides a unique identification within the catenet; a full network address consists of the following parts:

Network ID Each network solution with a catenet is assigned a unique 32-bit Network ID. DCN systems will use Network IDs assigned by Xerox to ensure their being unique worldwide.

System ID At the time of its manufacture, each DI is assigned a unique 48-bit identification number from a pool of numbers allocated to Control Data by Xerox. This number is written into battery-backed RAM and is used throughout the catenet as the System ID for that DI.

The System ID is used as the Ethernet address for any system that is locally connected to one or more Ethernet network solutions.

HDLC station addresses are handled locally by any two DIs that are locally connected by an HDLC line; the station address is not used as the System ID.

SAP ID A SAP can be viewed as a "port" through which two components communicate with each other. The SAP ID is a 16-bit number that identifies a SAP in such a way that software in the local system can access another component by its SAP ID. Dedicated ("well known") SAP IDs are assigned to some frequently used components; other SAP IDs are dynamically assigned as needed.

3.3.1 Lower Layers

Lower layer software is concerned with moving data from one system to another system within a specific network solution. In addition, the lower layers are responsible for routing and relaying data units within a single network solution. Note that the network solutions to be supported in Release 1 do not need software support to relay or route within a network solution and consequently do not perform any routing. Lower layer software will be involved in routing in subsequent releases, in particular for the C180 channelnet.

The following network solutions will be supported in DCN release 1:

- Ethernet
- Point-to-point HDLC line (up to 56Kbps)
- X.25 virtual circuit used to interconnect two DIs

In subsequent releases, when a C180 mainframe is supported as a CDNA system, the channel interfaces between a C180 mainframe and one or more DIs will make up another example of a network solution called the "channelnet."

Independent software components exist for layer 1 of each network solution. Layers 2 and 3A software includes code that is common to all network solutions as well as code specific to each network solution. Layer 1, 2 and 3A software for a given network solution work together to move data between systems connected to that network solution.

3.3.1.1 Layer 1

Layer 1 software for Release 1 consists of the following:

- CIM controlware
- ESCI controlware

CIM controlware resides and executes on the CIM card to support physical interconnection with different types of communication lines. It consists of three parts:

interface with layer 2 software

state programs to provide the interface to communication lines. Different state programs exist for different types of communications lines. At Release 1, state programs will exist to support synchronous HDLC lines and asynchronous terminal lines.

the base system, which runs in the background to provide resource (e.g. CPU) sharing between state programs.

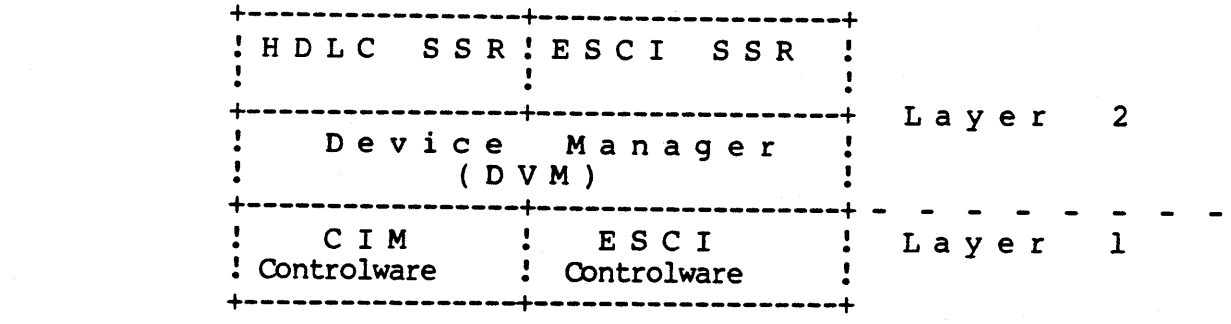
ESCI controlware resides and executes on the ESCI card to support physical interconnection with the Ethernet transceiver. Data to and from the Ethernet is physically moved by the Ethernet controller Intel 82256 chip, which is part of the ESCI card. The ESCI controlware programs this chip to move data between the SMM and Ethernet transceiver. A set of commands is programmed for the controller chip to inform it about the physical location of data buffers for outgoing data and empty buffers for incoming data.

Both the CIM and ESCI controlware interface with layer 2 software via the Device Manager (DVM). DVM resides and executes on the MPB card.

3.3.1.2 Layer 2

Layer 2 software includes the following:

- Device Manager (DVM)
- HDLC Stream Service Routine (SSR)
- ESCI Stream Service Routine (SSR)



DVM provides a general-purpose interface between layer 1 software on intelligent cards (e.g. CIM and ESCI) and the Stream Service Routines (SSRs), which execute on the MPB card. This interface is provided via a table called Device Control Block (DCB)

A separate DCB exists for each intelligent card in the DI. The DCB contains:

a command ring, which contains command packets sent to the layer 1 software by the layer 2 software.

a status ring, which contains status packets sent to the layer 2 software by the layer 1 software

two chains of empty buffers, which contain empty buffers to receive the incoming data units. Layer 1 software uses one chain at a time. When all the buffers in one chain are used up, layer 1 software starts using the second chain; layer 1 software notifies DVM when this occurs, so that DVM can obtain a new buffer chain.

Layer 1 software communicates with the SSR software by constructing a status packet, adding it to the status ring in the DCB, and interrupting the MPB card. The interrupt is processed by the DVM, which checks for any outstanding status packets. If one is found, it is sent to the SSR via an intertask message.

3.3.1.2 Layer 2 (continued)

SSR software calls a DVM subroutine in order to send a command packet to layer 1 software. DVM adds the packet to the command ring in the Device Control Block and interrupts the intelligent card which contains the corresponding layer 1 software. Upon receipt of the interrupt, layer 1 software checks the DCB for any outstanding command and processes the command(s) sent by the SSR.

SSRs are responsible for statistics collection and error logging for themselves as well as associated layer 1 software.

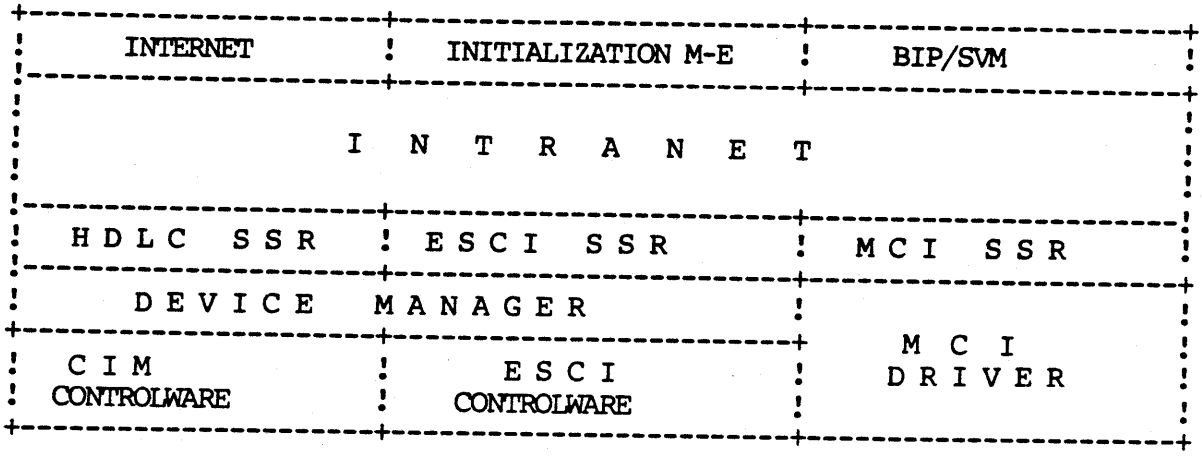
The HDLC SSR provides layer 2 functions for the point-to-point synchronous HDLC line. HDLC SSR implements the balanced mode HDLC protocol; options to support the LAPB protocol are included, as are those options needed to load a remote DI across the HDLC line. These options can be selected dynamically. HDLC SSR exists as a separate task for each configured line. It is used to support the layer 2 functions for both a general-purpose X.25 interconnection to a PDN and a point-to-point HDLC line used as a network solution.

The ESCI SSR is not required to support any complex protocol. As a result, the ESCI SSR is a relatively simple piece of software.

3.3.1.3 Layer 3A

Layer 3A software consist of a single component called Intranet. Intranet provides an interface between the other lower layers (1 and 2) and a subset of the users of these layers. This subset includes users who view layers 1 and 2 as providing an interface to a network. Intranet users, illustrated below, are:

- Internet layer (3B)
- Initialization M-E
- BIP/SVM



Intranet provides the following services to its users:

- open and close a 3A SAP
- send data requests
- receive data and broadcast indications
- notification, to users with an open 3A SAP, of changes in the status of underlying network solutions.

Intranet maintains output queues for the underlying network solutions. For each queue, Intranet maintains two thresholds, called congestion and uncongestion thresholds; the congestion threshold is higher than the uncongestion threshold. If the number of outgoing messages in a queue exceeds the congestion threshold, the associated network solution is considered "congested." It remains congested until the number of messages in the queue becomes less than the uncongestion threshold. Two distinct thresholds are maintained to avoid the situation where the status of a network solution is continuously changing between congested and uncongested.

3.3.1.4 C170 Channel Support

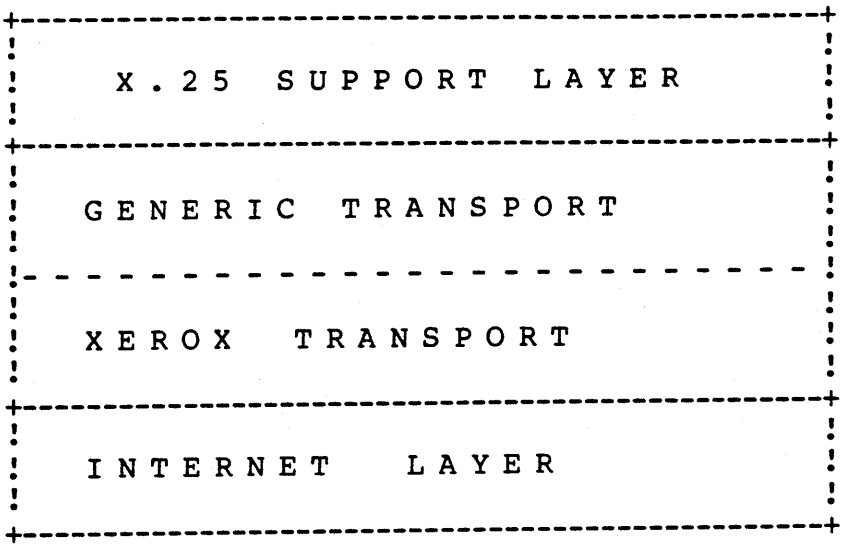
Release 1 supports a channel interface with a C170 mainframe. Although the C170 mainframe channel interface software cannot be associated with any network solution and is not actually a CDNA lower layer, the DCNS implementation is patterned after the lower layer groups discussed in this section.

The "layer 1" software for the C170 mainframe channel interface is called the MCI driver. It resides on the SMM card and executes on the MPB card. It uses the hardware on the MCI card to move data to and from the channel interface. The MCI driver does not use the DVM to interface with the MCI "layer 2" software, because both "layers" reside on the same card; they can communicate using a direct interface.

The "layer 2" software for the C170 mainframe channel interface is not required to support any complex protocol; as a result, the MCI SSR is a relatively simple piece of software.

3.3.2 Middle Layers

The OSI model defines a single middle layer, called the Transport layer (4); CDNA/DCN defines the Transport layer to consist of two "sub-layers," Generic Transport and Xerox Transport. CDNA/DCN defines two additional middle layers: the Internet layer (3B) and the X.25 Support layer. The relationship between these middle layer components is illustrated below.



3.3.2.1 Internet Layer

The Internet layer is responsible for relaying of messages between different network solutions within the catenet. In DCN release 1, the Internet layer functions are provided via the Xerox Internet protocol.

The Internet layer provides the following services to users within its system:

- Datagram Request
- Datagram Indication
- Broadcast Indication

In addition, the Internet layer manages an interface with the Intranet (3A) layer to transmit and receive data units from locally connected network solutions.

The Internet layer routes data units based on the destination network address, which consists of a Network ID, System ID and 3B SAP ID.

The Internet layer receives data units from two sources:

from within the CDNA system in which it resides

from outside the system across a locally connected network solution.

In either case, if the destination Network ID identifies a locally connected network solution, the destination System ID is used to determine if the data is destined for this system.

If it is, the destination 3B SAP ID is used to determine the appropriate software component within this system.

If it is not, the data is transmitted across the specified locally connected network solution to the destination system specified by the destination System ID.

If the destination Network ID identifies a remotely connected network solution, the Internet layer uses the Internet Routing tables to determine if one or more paths to the final destination exists. The Internet layer uses the relative cost of the paths to share the load over the possible paths in exact proportion to their cost.

3.3.2.1 Internet Layer (continued)

If a path exists, the Internet Routing tables contain the Network ID/System ID of next "hop," i.e., the next system enroute to the final destination; the Network ID identifies the locally connected network solution to be used by the Internet layer to transmit the data unit.

Each time Internet layer relays a data unit, it increments the cumulative hop count in the Internet header of the data unit. If the resultant hop count exceeds sixteen, the Internet layer discards the data unit and notifies the originator. If the Internet layer cannot re-route or deliver a received data unit for any other reason, it discards the data unit and notifies the originator.

The Internet layer relies on the Routing M-E to maintain the Internet Routing tables as well as to provide services to open and close Internet SAPs. The section on the Routing M-E discusses these services in more detail.

3.3.2.2 Transport Layer

The Transport layer is implemented as two software components, or "sub-layers:"

Generic Transport -	provides services identical to NBS and ISO class four Transport. It uses the Xerox Transport to help provide these services. Where there is no direct mapping between NBS/ISO services and Xerox Transport services, it provides the functions needed to support this mapping.
Xerox Transport -	implements the Xerox sequence packet protocol.

All Release 1 DI-resident users of the Transport layer use the Generic Transport services. (Subsequent DCNS releases may include support of standards other than Xerox Transport; use of Generic Transport will eliminate the need for Transport layer users to be concerned with the specification Transport layer standard.) Implementation of Xerox Transport does not, however, preclude future applications from using its services directly.

Generic Transport provides the services listed below to its users. Additional information on these services appear in the CDNA GDS.

- SAP management services
- Connection establishment and disconnect services
- Data transfer services
- Expedited data transfer services
- Status services

Generic Transport does not limit the maximum data size of data being transferred via the Data Transfer services; however, Xerox Transport has an upper limit on the maximum data unit size. As a result, Generic Transport includes functions to fragment and re-assemble data units as necessary; fragmentation and re-assembly is transparent to Generic Transport users.

3.3.2.2 Transport Layer (continued)

Services provided by Xerox Transport are very similar to those provided by the Generic Transport. There is a one-to-one match between the primitives (interfaces) provided by the two Transports to support the services; however, Xerox Transport primitives have the following additional features:

- Each data unit can be assigned a data stream type. Xerox Transport simply passes it on from one user to the other (peer) user.
- Each data unit can be qualified to be or not to be the last data unit in a sequence. This feature is used by Generic Transport to facilitate its fragmentation and re-assembly of data units.

3.3.2.3 X.25 Support Layer

The X.25 Support layer provides a service equivalent to that provided by an X.25 network. It provides this service by using the services of the Transport layer and a CDNA protocol that encompasses X.25-specific characteristics.

In Release 1, services of the X.25 Support layer are used by the following functions:

- o C170 Transform for Transparent A-A connections between DCN and C170 Network Products
- o X.25 Transform between DCN and X.25 network

In subsequent releases, the X.25 Support layer will provide X.25 network access to C180 applications that do not have direct access to an X.25 network.

The X.25 Support layer provides the following services to its users:

SAP management services allow a user to open and close an X.25 Support layer SAP. An open SAP is required to receive incoming call (connect) requests. Connection establishment and disconnect services support all primitives (e.g. request, indication, response, confirm) normally associated with these services.

Data transfer services support the data request and data indication primitives. Both primitives allow specification of one or more of three possible flags:

- The Q flag corresponds to the qualifier bit in the X.25 packet header.
- The D flag corresponds to the delivery confirmation bit in the X.25 packet header.
- The M flag corresponds to the "more data" bit in the X.25 packet header.

Expedited data transfer services allow a user to send one byte of data which is guaranteed to be delivered before any subsequent data requests.

Reset services allow a user to re-initialize the connection. All queued data at both ends of the connection is discarded by the two peer X.25 Support layer entities.

Change block size services allow a user of the X.25 Support layer, to change the maximum data unit size for the incoming data units.

3.3.3 Higher Layers

The higher layers provides data-dependent, end-to-end services and protocols to allow applications to communicate with other applications or with terminals.

DCNS higher layer services are implemented as a dual-endpoint liaison connection service. The end users at the endpoints of the connection are viewed as "associates." The connection is called an "association." Conceptually, an association may be viewed as a logical connection between two peer associates.

DCNS implements the OSI higher layers as "layer groups." Consolidating the layer 5-7 functions into groups is necessary since industry standards for individual layers are not complete and agreement has not been reached as to how the higher layer functions are to be distributed. Although the provided services and protocols of each DCNS layer group are not directly mappable to OSI layers 5-7, they do provide functions needed to allow communication between user/application processes.

DCNS will initially provide two higher layer groups:

Interactive Transfer Service(ITS) - provided in Release 1

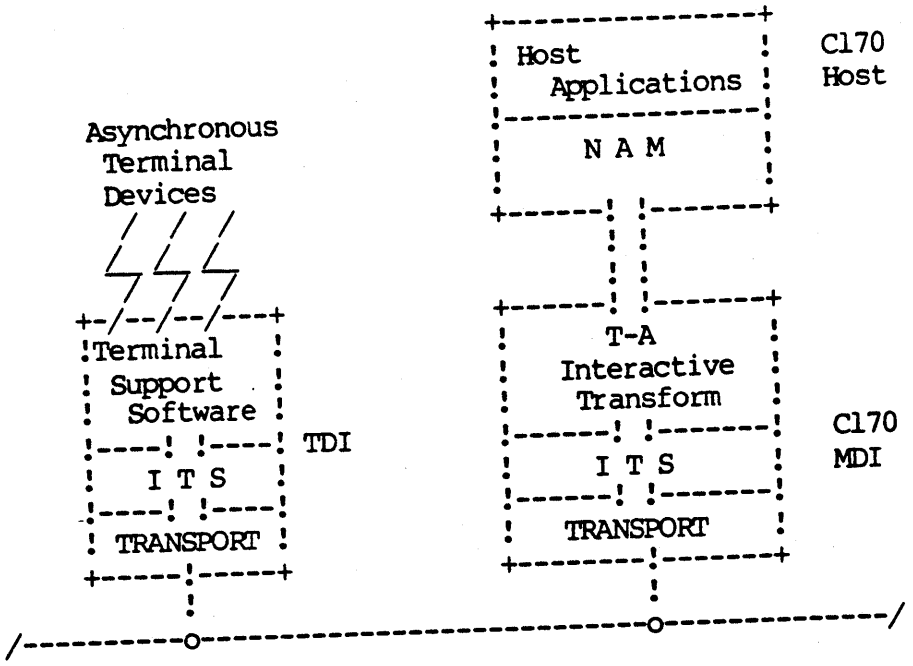
Batch Transfer Service (BTS) - provided in subsequent release(s)

ITS is the only higher layer group available in the initial release of DCN. ITS provides the set of services required for communication between two DCN end-users engaged in the transfer of interactive data. The two end-users may both be applications (A-A) or one may be an application and the other a terminal user (T-A).

Full implementation of ITS will be phased, with early releases of DCN focusing on ITS support of Terminal-to-Application (T-A) communication. The initial ITS implementation specifically addresses the requirements of asynchronous terminal communication with C170 applications.

The diagram below illustrates DCNS components involved in support of asynchronous terminal access to a C170 host.

3.3.3 Higher Layers (continued)



The C170 Interactive Virtual Terminal (IVT) representation of data must be transformed to the ITS representation for transmission across the DCN. This transformation is performed by the "T-A Interactive Transform." The T-A Interactive Transform maps the IVT protocol on to the one supported by ITS. The T-A Interactive Transform is not a layer and is mentioned here simply to provide a more complete picture of the Release 1 higher layer group.

One additional component must be mentioned and that is the Terminal Support Software, which is also not a layer. The Terminal Support Software is the DCN implementation of the architecturally defined Terminal Access Process. In the initial DCN release, ITS associations (connections) exist between Terminal Support Software in a TDI and the C170 Interactive Transform in an MDI, with the Terminal Support Software being the initiator of the association.

The Terminal Support Software offers the terminal user the flexibility to establish and maintain multiple, simultaneously active associations in Release 1. Future releases will support such features as screen management and windowing.

The Terminal Support Software uses a set of ITS services that are different from the set used by the C170 Interactive Transform; ITS services for T-A communication are said to be "asymmetric" for this reason.

3.3.3 Higher Layers (continued)

The Terminal Support Software uses the ITS asymmetric service for the following functions:

- Establish an ITS association
- Transmit Edited terminal input text
- Transmit Transparent terminal input text
- Transmit notification of terminal break
- Transmit one character of terminal interrupt data
- Transmit notification and values of changed terminal attributes
- Transmit a response to a request for the current values of terminal attributes
- Terminate an ITS association

The C170 Interactive Transform uses the ITS asymmetric service for the following functions:

- Respond to a request to establish an ITS association
- Transmit Formatted terminal output text
- Transmit Transparent terminal output text
- Transmit request to discard queued terminal output data
- Transmit request for current terminal attributes values
- Transmit request to change terminal attributes values
- Terminate an ITS association

3.4 NETWORK MANAGEMENT

Network management consists of those functions within the DI that are required to manage the Distributed Communications Network. Network management functions are provided by Management-entities (M-Es). M-Es are defined in the CDNA GDS as part of the Control Data Network Architecture (CDNA). M-E software is always present in the DIs.

Management-entities provide the following functions:

- Creation and management of Internet Routing Tables
- Title definition and translation
- Command processing
- Logging
- File access
- Error processing
- Echo processing

Each function in this list is provided by a separate Management-entity. Several M-Es consist of two parts, "dependent" and "independent;" the dependent M-E relies upon the independent M-E to provide services. This division is necessitated by the fact that it is either impossible (or impractical) to support the entire set of required functions in each DI. For example, a sub-function of file access is concerned with direct or indirect access to some sort of mass storage; not all DIs have access to mass storage and therefore will not be able to support this sub-function. Similarly, a sub-function of command processing provides an interface to a command source such as a terminal user or an application; since not all DIs support terminal users or applications, there is no need to support this sub-function in each DI.

The Command M-E, Log M-E and File Access M-E have explicit and stand-alone dependent and independent parts, where each part is implemented in a separate piece of software. The software implementing the dependent part of these M-Es will be present in every DI. The software implementing the independent part will be present in some of the DIs. As a result, some DIs will have both dependent and independent parts of a given Management-entity.

3.4.1 Routing M-E

The Routing M-E is responsible for creation, update and management of the tables needed to route information between CDNA systems. The Routing M-E also provides the services necessary to open and close 3B SAPs.

Routing involves finding a path from one system to another, using this path to transfer data between the two systems, and locating the software that is to receive the data within the destination system. The routing process starts when layer 3B software is requested to send a data unit to another system. This request includes the network address of the destination system (Network ID and System ID) as well as the 3B SAP ID of the destination software. The routing process consists of the following sub processes:

Inter-network routing - responsible for ensuring that the data unit reaches the network solution whose Network ID is the same as that of the destination system. This is done by delivering the data unit to any system that is locally connected to the desired network solution.

Intra-network routing - takes over once the data unit reaches a system connected to the destination network solution. It is concerned with delivering the data unit to appropriate system on the destination network solution. The System ID in the destination address is used to locate the desired system.

Intra-system routing - responsible for delivery of the data unit to the appropriate software component once the data unit reaches the addressed system. The 3B SAP ID is used to accomplish this routing. Each of these three types of routing is described next in more detail.

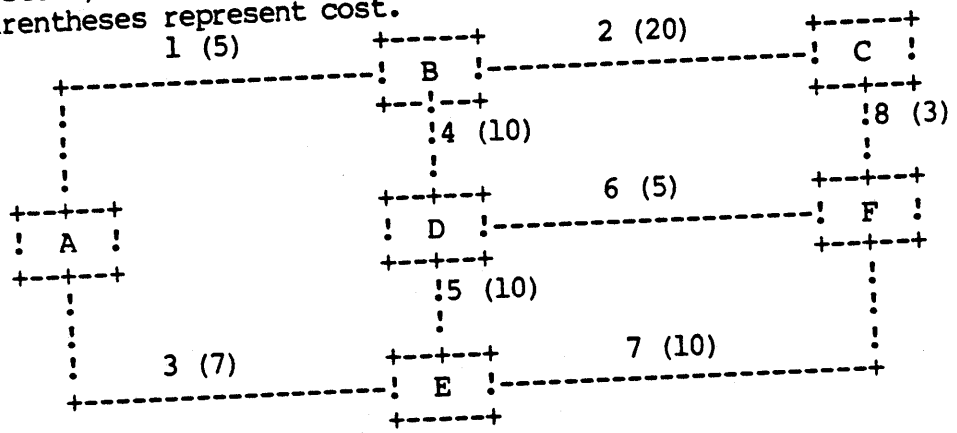
3.4.1.1 Inter-network Routing

Inter-network routing is the most complex of the three sub-processes. It requires each CDNA system to know the next hop in the path to every network solution in the catenet. Fortunately, in any catenet configuration, the number of network solutions will be considerably smaller than the number of systems.

Each CDNA system contains Internet routing tables. These tables are created when the system first comes up and are dynamically updated. The process to build and update these tables is described in the "generation of routing tables" section.

Internet routing tables are also known as the Least Cost Routing Data Store or LCR-DS for short; the LCR-DS contains an entry for each locally connected network solution. These entries exist as rows in the LCR-DS; each row contains information about one or more paths to the network solution associated with the row. This information is included in what is known as a routing entry. There is one routing entry for each path.

Let us consider the following configuration as an example to illustrate some of the concepts. In this configuration, lettered boxes are used to represent systems; numbered lines are used to represent network solutions; numbers in parentheses represent cost.



Let us assume that we are looking at routing tables in system A and we are interested in all paths from system A to the network solution numbered 8. The following is a partial list of all possible paths from system A to network solution 8.

Path-1 A-1-B-2-C-8
Path-2 A-1-B-4-D-6-F-8
Path-3 A-1-B-4-D-5-E-7-F-8

Path-5 A-3-E-7-F-8
Path-6 A-3-E-5-D-6-F-8
Path-7 A-3-E-5-D-4-B-2-C-8

Some of these paths will show up in the LCR-DS row associated with network solution 8.

3.4.1.1 Inter-network Routing (continued)

The first criterion for the paths in the LCR-DS row is that they should not be overlapping; two paths are said to be overlapping if they start out on the same network solutions. In the above list, paths 1 through 3 start out on network solution 1; therefore they overlap with each other and so only one of these will end up in the LCR-DS row for network solution 8. Similarly, only one path out of paths 5 through 7 will find its way to the LCR-DS row.

The second criterion for a path to be included in the LCR-DS is that, of the overlapping paths, the one with the "least cost" will be selected. For Release 1, the cost of a network solution is inversely proportional to its accessible (static) bandwidth. The cost of a path will be the sum of costs of all network solutions that make up the path. A cost of "1" for a network solution with bandwidth of 100 megabits per second is used to normalize the cost of various network solutions. This normalization yields the following cost for some typical bandwidths:

Bandwidth	Cost
10 Mbs	A(16)
1.54 Mbs	41(16)
56 Kbs	6FA(16)
9.6 Kbs	28B0(16)

In the sample configuration, the numbers in parenthesis show the cost of each network. Using these numbers, one gets the following costs for the listed paths:

Path	Cost
1	25
2	20
3	35
5	17
6	22
7	47

Since only one of paths 1 through 3 is to be selected, path number 2 will qualify. Similarly, of paths 5 through 7, path number 5 will be selected. Paths 2 and 5 will be included in the LCR-DS row according to their increasing cost. Therefore, path number 5 will be the first path and path number 2 will be the second path in the LCR-DS row.

However, only limited information about the two selected paths is included in the LCR-DS. The following table shows the information included in the LCR-DS for these paths:

	COST	FIRST NETWORK SOLUTION	FIRST RELAY SYSTEM
(Path 5)	17	3	E
(Path 2)	20	1	B

3.4.1.1 Inter-network Routing (continued)

It is very important to understand that the Internet routing table (LCR-DS) does not contain complete information about each path. Basically it contains only the following information:

- o Existence/non-existence of a path
- o Network ID and System ID of the first network solution and system in the path; the next hop
- o Cost of the total path

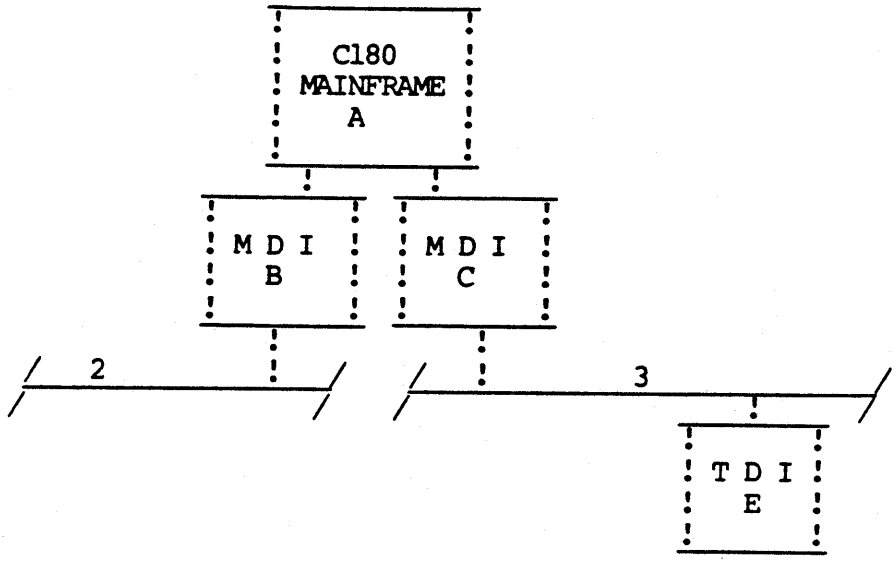
This allows dynamic and adaptive selection of the path. In the above example, when a data unit leaves system A, it starts out on the best possible path, based on current information. It will travel on network solution 3 to system E. In system E, the best path will be selected once again to take it to its final destination. Normally, this path will be the network solution 7. However, the routing process will have built-in intelligence to react to dynamic changes in the configuration, such as failure or congestion of network solution 7.

There is one more restriction on paths considered for inclusion in the LCR-DS. Any path that involves more than sixteen hops is automatically rejected. The Internet layer (3B) software keeps the cumulative hop count with the data unit. If the hop count exceeds 16, the data unit is discarded; this prevents endless relaying of a data unit in the catenet.

3.4.1.2 Intra-network Routing

Intra-network routing is concerned with routing of a data unit within a network solution. In Release 1, intra-network routing by the software is not needed, so this concept will be illustrated by using the channelnet network solution that will be supported in a subsequent release.

Intra-network routing is initiated when a data unit reaches a system that is locally connected to the destination network solution. Let us use the following configuration as an example to illustrate the intra-network routing.



Suppose the C180 mainframe is sending a data unit to TDI-E. The address of this TDI will be stated to be Network ID=3/System ID=E. When this data unit reachs MDI-C, the MDI-C 3B software will do the following:

a) It will check the destination Network ID of the data unit and find it to be 3.

b) It will check its LCR-DS to see if network 3 is accessible.

c) It will find that this network is accessible and is locally connected.

d) It will compare the destination System ID (E) with its own System ID (C) and find that they differ.

e) It will transmit across network solution 3 and address it for system E.

f) TDI-E will repeat steps a through d and accept the data unit.

3.4.1.2 Intra-network Routing (continued)

The above configuration shows two MDIs connected to the C180 mainframe via channel connections; both channel connections are assumed to be configured as part of a single network solution. All or some of the channel connections to the same C180 mainframe may be configured as a single network solution.

In the above configuration, systems A, B, and C are all connected to the same network solution, namely the channelnet numbered 1. Now let us assume that system B wants to send a data unit to system C. The destination address of this data unit will be specified as Network ID=1/System ID=C. The following steps describe the intra-network routing processing:

a) The Internet (3B) software in system B will use its LCR-DS to determine that network solution 1 is accessible and locally connected. It will also determine that the data is not destined for itself.

b) The Internet (3B) software will transmit the data unit to the system A.

c) This data unit will be received by the channelnet Intranet (3A) software in the C180 mainframe (i.e., System A).

d) The channelnet Intranet (3A) software in system A will determine that the destination network is the channelnet itself, but the destination system is other than System A.

e) The channelnet Intranet software will relay the data unit to system C.

3.4.1.3 Intra-system Routing

Intrasystem routing is concerned with delivering the received data unit to the appropriate software component within the system. The Internet (3B) software maintains a table of 3B SAP IDs of all software components that either have opened 3B SAPs or have a dedicated SAP. This table also contains the current status of the SAP (i.e. open or closed) as well as the address of a procedure to be called to deliver the incoming data units.

After verifying that the Network ID and System ID of the received data unit are the same as that of the system where the data unit is received, the Internet (3B) software uses the 3B SAP in the address part of the received data unit to find a matching entry in its 3B SAP table. If the entry is found and the SAP is open, the received data unit is delivered to the associated software component. Otherwise, the data unit is discarded and an error PDU is sent to the source of the data unit.

3.4.1.4 Generation of Routing Tables

The following tables are created and updated by the Routing M-E and are used to accomplish the three types of routing described in the previous section:

- 3B SAP table
- Local Directly Connected Network Solution Data Store (L-DCN-DS)
- Remote Directly Connected Network Solution Data Store (R-DCN-DS)
- Local Network/Community Title - Address Table
- Remote Network/Community Title - Address Table
- Least Cost Routing Data Store (LCR-DS)

The 3B SAP table contains information about dedicated 3B SAPs as well as non-dedicated SAPs that are open. Creation and update of this table is straightforward and does not need any further description. This table is used for the intrasystem routing.

The L-DCN-DS contains information about all locally connected network solutions.

The R-DCN-DS contains information about all remotely connected network solutions.

The Local Network/Community Title/Address Table contains Titles/Addresses of locally connected network solutions and of the "communities" to which the local system belongs. Communities are defined via configuration commands.

The Remote Network/Community Title/Address Table contains Titles/Addresses of remotely connected network solutions and of "communities" to which the remote multi-homed systems belong. This information is received in Routing Information data units from remote multi-homed systems.

The LCR-DS is a simple table and is easy to use to do the inter-network and intra-network routing. However, the process used to create and update this table is rather complex. The following discussion provides a simple overview of this process; the reader is referred to the CDNA GDS for rigorous details.

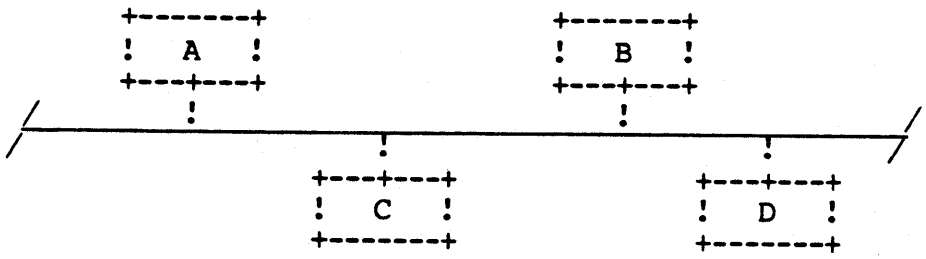
3.4.1.4 Generation of Routing Tables (continued)

The LCR-DS contains information about least cost paths to accessible network solutions. Even though this table does not contain information about every possible path, the process used to create and update this table has to know these paths completely in order to select the least cost non-overlapping paths.

Information about different paths can be either pre-defined or computed dynamically. In CDNA this information is not pre-defined; instead it is dynamically computed and updated periodically.

Any path definition has to be based on configuration information of individual CDNA systems that make up the path. However, no single system can compute these paths based on its own configuration information (trivial paths being exceptions). Therefore, any system or process interested in computing such paths must obtain up-to-date configuration information from other systems if tables are to be maintained dynamically. Fortunately, configuration information is needed from a relatively small number of systems.

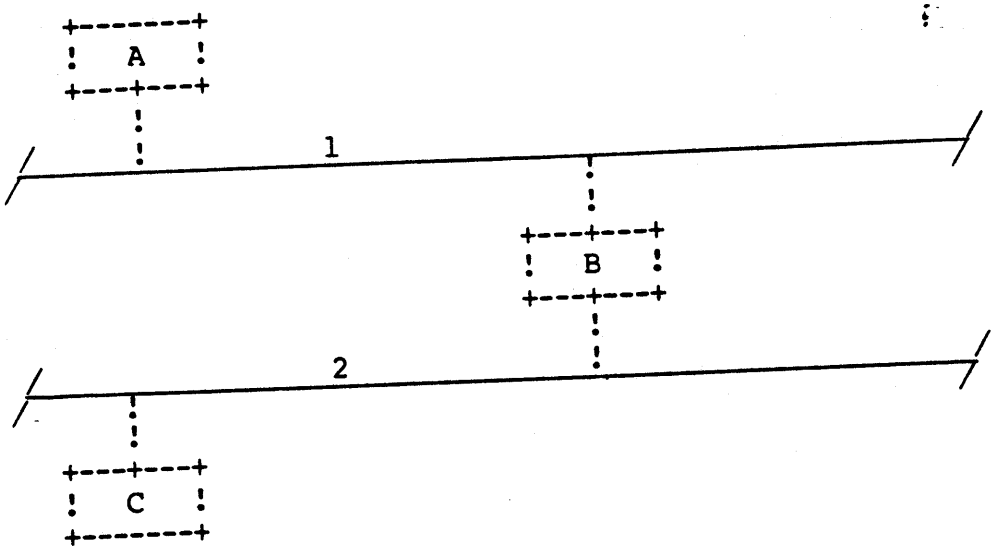
For example, consider the following configuration, which includes four DIs connected to a single network solution, namely Ethernet.



Let us assume we are trying to build the LCR-DS in system A. This configuration contains only one network solution that is locally connected to system A. Systems B, C, and D cannot provide any new information to system A to help it build its LCR-DS. In this case, the LCR-DS will contain a single row that is associated with the locally connected network solution.

3.4.1.4 Generation of Routing Tables (continued)

Next, let us look at a more complex configuration.



Once again, let us try to build the LCR-DS in system A. Both network solutions numbered 1 and 2 are accessible from system A; however, system A by itself cannot find out that network solution 2 is remotely connected to it. System B must tell system A that it (System B) is locally connected to both network solutions. Only then system A can deduce the fact that it can reach network solution 2 via system B. Similarly, system C must receive the same information from system B, in order to learn that network solution 1 is accessible to it.

The points being made by the above examples may be summed up as follows:

- Systems that are locally connected to a single network solution do not need to tell other systems about their configuration.

- Systems that are locally connected to more than one network solution need to tell all other systems in the catenet about network solutions locally connected to them.

3.4.1.4 Generation of Routing Tables (continued)

If a system is locally connected to more than one network solution, the Routing M-E periodically broadcasts Routing Information PDUs (RI-PDUs) on all its locally connected network solutions. The RI-PDU contains information about locally connected networks as well as local network and community titles. Normally, an RI-PDU is sent once every thirty seconds, but will be sent sooner if a change takes place in the information included in the RI-PDU.

Each RI-PDU contains a sequence number and a change/no change flag. Each time an RI-PDU is transmitted, its sequence number is incremented by one and all changed information is flagged.

The Routing M-E in each system, receives RI-PDUs from other systems. The sequence number of the received RI-PDU is checked. If the sequence number is the same as one received previously from the same system, it is discarded. Otherwise, the following is done:

- A copy is made and saved for local processing.
- The RI-PDU is broadcasted on all locally connected network solutions, except the one on which it was received.
- The local copy is examined for change flag(s). If there are no changes, the processing ends. Otherwise, the changed information is used to update the corresponding information in the R-DCN-DS and remote network/community title-address table.
- If there were any changes in the R-DCN-DS entries, the LCR-DS is re-computed and updated.

3.4.1.4 Generation of Routing Tables (continued)

DCN Release 1 systems connected to more than one network solution are called NDIs. Each NDI sends information about its locally connected network solutions via Routing Information PDUs. As the status of connected network solutions in the L-DCN-DS or R-DCN-DS changes, the LCR-DS is re-computed to incorporate effects of these changes. The step-by-step process used to create the LCR-DS is as follows:

a) Create the Local Directly Connected Network Solution Data Store (L-DCN-DS), which contains a list of, and information about, all locally connected network solutions.

b) Maintain the Received Directly Connected Network Solution Data Store (R-DCN-DS), which contains an entry for each system from which Routing Information PDUs are received. Each entry will contain information about network solutions locally connected to the system associated with that entry.

c) Since all locally connected networks in the L-DCN-DS are obviously reachable, create a row for each locally connected network solution in the LCR-DS (i.e., Internet routing table.)

d) For each locally connected network solution in the LCR-DS, search the R-DCN-DS entries to see if its Network ID matches one of the locally connected network solutions in the R-DCN-DS entry. If a match is found, then all other locally connected network solutions in that entry will also be reachable from the system, where the LCR-DS is being created. Therefore, add these other locally connected network solutions (i.e. create a new row) to the LCR-DS, if they are not already present. If they are already present and the newly discovered path does not overlap one already present in the LCR-DS row, add this path to the row. If the newly discovered path overlaps one already present, compare the two costs and retain the lower cost path in the row.

e) As a result of step d, the LCR-DS will contain rows for remotely connected network solutions. Repeat step d for these rows, etc.

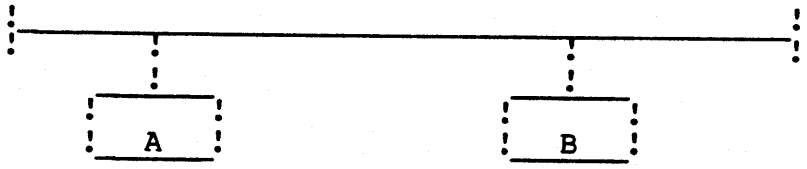
3.4.1.5 Open Issues on CDNA Routing

The following is a list of limitations and problems with CDNA routing as described in the CDNA GDS. Some of these limitations are inherent to the chosen philosophy for routing. Others are problems or holes which need to be addressed.

a) The algorithm used to compute the cost of a path does not take into account the overhead involved in relaying a data unit from one network solution to another. This is the CPU overhead in the system doing the relay. This cost should be easy to compute and include in the total cost.

b) The impact of congestion on the cost associated with a specific network solution is not adequately addressed.

c) The routing tables do not contain any information about the status of systems which are connected to one network solution only. For example in the following configuration, system A has no way of finding if system B is up or down. The best it can do is to detect the absence of system B via time out mechanisms in Transport layer software. This limitation is inherent in the approach chosen for the CDNA routing, but is not serious.



3.4.2 Directory M-E

A Directory M-E is resident in every DI. Each individual Directory M-E maintains local data stores that contain Title information that is most beneficial to its local system and the catenet as a whole. Collectively, the Directory M-Es maintain a distributed Directory of Titles and associated Addresses.

Directory M-Es use Internet datagram and broadcast services to distribute Title information to Directory M-Es in other systems. Dissemination of this information is conducted under the control of the Directory M-E protocol, which allows Directory M-Es to communicate in a manner that minimizes the amount of network traffic required to maintain a destributed Directory and provides Directory services at a high performance level.

Directory M-E services fall into two categories:

Registration services	Software components use Registration services to announce availability, location, and specific characteristics of services they offer. During initialization, each component determines its own network address and uses Registration services to announce its availability to other components at the location specified by its network address. Registration services are also provided to alter and delete entries from the Directory.
Translation services	Software components use Translation services to determine the network location of a required or requested service.

3.4.2.1 Directory Entries

The Directory consists of multiple Directory Entries, each of which contains a registered Title/Address pair. A Title/Address pair consists of the following information:

Title - any string of 1 to 256 ASCII (Parity Bit = 0) characters.

Address - a field of octets that contains one of the following:

A network address that consists of:

3B Network ID, or
3B Network ID + 3B System ID, or
3B Network ID + 3B System ID + 3B SAP ID
3B Network ID + 3B System ID + 3B SAP ID + 4 SAP ID

A non-network address that is a 32-bit machine-address field used by components within a system to exchange interface addresses. Non-network addresses have only local system significance and therefore Directory information relative to non-network addresses is not distributed throughout the network.

Each Directory Entry also contains a Directory Entry Identifier that uniquely identifies when and where that entry was created or last updated. Whenever a Directory Entry is changed in any fashion, its Directory Entry Identifier is also changed. The Directory Entry Identifier is an 18-octet value comprised of the local Network ID / System ID and date/time stamps.

A Directory Entry may optionally contain attributes that can be used in the following ways:

- specify additional Title-matching criteria that must be met before that Directory Entry will be considered to be a match in a request for translation;
- indicate a priority relative to other entries with the same Title;
- identify protocol(s) associated with that Directory Entry.

3.4.2.1 Directory Entries (continued)

The Directory user has some controls over domains in which a registered Title is meaningful, via the "Title Domain" and "Search Domain."

Title Domain - allows the Registration services user to provide some insight to the Directory M-E as to where information about a specific Title should be distributed. Title Domain specifies a subset of networks and systems within the catenet, from whence most of the requests for the registered Title will probably originate. A Title Domain specification does not, however, restrict knowledge of the registered Title to those networks and systems in the specified domain.

Search Domain - allows the Translation services user to specify a subset of networks and systems within the catenet to which the Directory M-E should limit its Title Translation search. A translation of a Title can be obtained from outside its registered Title Domain, if the Translation services user specifies a Search Domain that includes some portion of the specified Title Domain.

The following special Titles are used to specify domain parameters:

- "Community xx" — where "xx" is a 1-32 octet community name; community titles are registered as the result of configuration commands.

- "Broadcast Network rr" — where "rr" is the hex number of 3B relays (i.e., the number of hops) to the remote network; the Routing M-E registers one such Title for each locally connected network solution ("00" hops away), one such Title for each network solution that is "01" hops away, and so forth.

These special Titles are used by Directory users when they want to specify a domain in terms of "all systems belonging to community xx" or "all networks within rr hops from the local network." When the Directory M-E receives a request containing a domain identified by one of the above Titles, it searches its own Directory for that Title in order to determine the Address(es) corresponding to the specified domain.

3.4.2.2 Registration Services

Registration services include primitives to;

- register a Title, its corresponding Address, and other optional information in the Directory
- change information about a Title previously registered
- delete a Title from the Directory

Create Entry Primitive

A Directory user utilizes the Create Entry primitive to request that a new Directory Entry be created. Parameters are:

Title - specifies the ASCII Title, as defined above.

Address - specifies the Address to be associated with the Title in the Directory Entry. The user may specify the Address in any of the ways defined above, or, may specify only the 3B SAP ID or the 3B SAP ID + the 4 SAP ID. If only SAP ID(s) are specified or if no Address parameter is given at all, the Directory M-E supplies the Network ID / System ID of the local system.

Directly Accessible Service - optionally specifies the set of CDNA end-to-end protocols associated with this Directory Entry. This parameter can be used to ensure that the two potentially communicating end-users have compatible underlying services. This attribute is used as a Title-matching criterion only if it is specified on a Translation request, in which case only Titles registered with the same "directly accessible service" will be supplied in response.

Password - optional parameter that can be used as an extra measure of security. If this attribute is registered, it is always used as a Title-matching criterion; i.e., subsequent Translation requests must contain a matching password.

User Information - optional parameter that can be used to pass up to 32 octets of information in responses to subsequent Translation requests. This parameter is not used as a Title-matching criterion.

3.4.2.2 Registration Services (continued)

Create Entry Primitive (continued)

Title Domain - optional parameter that can be used to specify the domain over which the Title is to be known. A Title Domain may be specified as one of the following:

- Local System (default value)
- Local Network
- Broadcast Network "rr"
- Community "xx"
- Catenet

Distribution Frequency - specifies when and where to distribute the information associated with this Directory Entry:

- distribute, upon request, only to the requesting Directory M-E
- distribute, upon request, to the requesting Directory M-E and all Directory M-Es on the same network (default value)
- distribute, upon request, to the requesting Directory M-E and all Directory M-Es on the same network and also periodically distribute throughout the Title Domain. The frequency of distribution is configurable.

Priority - specifies the priority to be asociated with this Directory Entry. The range of priorities supported is 0 - 3, with 3 being the highest and default priority. When multiple entries exist for the same Title, the entry with the highest priority is returned in a Translation indication. This parameter is not used as a Title-matching criterion.

The Directory M-E processes a Create Entry request by creating an entry in its Registered Data Store (RDS). The RDS contains Directory Entries created by users in the local system.

The Directory M-E creates a Directory Entry Identifier for the newly created entry; the Directory Entry Identifier is returned to the requestor upon completion of processing.

If the Create Entry request specifies a Title Domain other than "Local System" and a periodic Distribution Frequency, the Directory M-E sends a copy of the newly created entry to all networks identified by the Title Domain.

3.4.2.2 Registration Services (continued)

Change Entry Primitive

A Directory user utilizes the Change Entry primitive to request that a specific Directory Entry be altered. The user must identify the target entry by supplying the Directory Entry Identifier. The user can change any of the Create Entry values except the Title parameter.

The Directory M-E processes a Change Entry primitive in the same manner as a Create Entry primitive, except that instead of creating a new RDS entry, an existing entry is updated. A new Directory Entry Identifier is generated and returned to the requestor.

Distribution, if any, of the altered entry is done according to the same rules used for the Change Entry primitive.

Delete Entry Primitive

A directory user utilizes the Delete Entry primitive to request that the a Directory Entry be deleted. In order to delete a Directory Entry, the user must specify both the Title and Directory Entry Identifer.

The Directory M-E processes a Delete Entry request by removing the entry from the RDS. If the Title Domain of the deleted entry is not "Local System" and periodic Distribution Frequency is registered, the Directory M-E notifies all networks identified by the Title Domain that the Directory Entry has been deleted.

3.4.2.3 Translation Services

Translation services include primitives for:

- requesting a Title Translation,
- notification that a Translation has been found,
- requesting that a previously requested Translation be cancelled,
- notification that a Translation request has been terminated.

Processing of Translation requests may require inter-system and/or inter-network communication between Directory M-Es. It is possible that a single Translation request will result in multiple Translation indications. For these reasons, one or more Translation indications may be returned asynchronously in response to a Translation request. Completion of Translation request processing is signalled by a Translation Request Termination primitive. Each of these primitives contains both a "user ID" and a "Directory M-E ID" that can be used to match requests and indications.

Translation Request Primitive

A Directory user utilizes the Translation Request primitive to request that the Directory M-E return Address(es) corresponding to the user-supplied Title and optional Title-matching criteria. The parameters of this primitive are as follows:

Title - specifies the 1 - 256 character Title to be matched. The full Title can be specified or the "wild-card" characters described below can be used to locate multiple Directory Entries that are related in some way:

? represents any single character

* represents any string of characters of arbitrary length

[ ] represents any single character that is:

a member of the set enclosed in square brackets (e.g., [abqz] means "a" or "b" or "q" or "z")

within a range (e.g., [0-9])

3.4.2.3 Translation Services (continued)

Translation Request Primitive (continued)

Out-of-date Identifier - specifies a Directory Entry Identifier that the user previously obtained in a Translation indication; the Address associated with that Identifier could not be used to reach the destination service. If an Out-of-date Identifier is specified, the Directory M-E does not consider that Directory Entry to be a match for the Translation request. Up to eight Out-of-date Identifiers may be specified on a single Translation request.

Directly Accessible Service - optional parameter that specifies the set of CDNA end-to-end protocols that are to be associated with any matching entries. This parameter can be used to ensure that the two potentially communicating end-users have compatible underlying services.

Password - provides a measure of Directory Entry security. If specified on a Translation request, a matched Directory Entry must have the same value. If not specified on a Translation request, a matched entry must not contain a password.

Search Duration - optional parameter that specifies the maximum amount of time that the Directory M-E is to search for the requested Translation. Minimum, maximum, and default time values are configurable.

Search Domain - specifies the domain across which a search for the Title is to be made. A Search Domain may be specified as one of the following:

- Local System (default value)
- Local Network
- Broadcast Network "rr"
- Community "xx"
- Catenet

3.4.2.3 Translation Services (continued)

Number of Translations - specifies the number of Translations to be searched for and returned to the requestor; the default value is one.

If the Search Domain is not "Local System," the value restricted to the range 1 - 8.

If the Search Domain is "Local System," the value can be in the range 1 - 255.

The number of translations obtained may be less if the Search Duration expires prior to receiving the number of requested translations.

The Directory M-E processes a Translation request by making an entry in its Translation Request Data Store (TRDS) and generating a Translation confirm to the requestor. The Translation confirm notifies the user that the request was valid and that the Directory M-E is now searching for the requested Title Translation.

The Directory M-E begins searching the Search Domain by searching data stores in its local system. Two data stores are involved in the local system search:

The first data store is the RDS which contains entries resulting from the processing of Create Entry request primitives from local system components.

The other data store is the Translation Data Store (TDS). The TDS contains entries for Title Translations learned about through received Directory protocol data units. Entries in the TDS are directly related to the Title Domain specified for a Title registered in another system. The size of the TDS is fixed at one-hundred entries; the oldest entries are cycled out when maximum size is reached.

After both local data stores are searched, the Directory M-E sends out Directory protocol data units, requesting the aid of Directory M-Es in other systems.

Whenever a matching Title is found, a Translation Indication is sent to the originator of the Translation request and the number of translations in the corresponding TRDS entry is decremented. When this number reaches zero, Translation request processing is complete. The TRDS entry is deleted and a Translation Request Termination is sent to the originator of the Translation request.

3.4.2.3 Translation Services (continued)

Translation Indication Primitive

The Translation indication primitive is used by the Directory to inform the Directory user that a Title Translation has been found for a previously issued Translation request. The parameters of this primitive are as follows:

Title - The Title contained in the matched Directory Entry.

Address - The Address associated with the matched Directory Entry.

Directory Entry Identifier - The identifier of the matched Directory Entry.

Directly accessible Service - The Directly Accessible Service attribute of the matched Directory Entry.

User Information - The User Information contained in the matched Directory Entry.

Priority - The priority associated with the matched Directory Entry.

Abort Translation Primitive

This primitive provides the Directory user with capability to cancel a previously issued Translation request.

Translation Request Termination

This primitive is used by the Directory M-E to notify the Directory user that processing relative to a specific Translation request has terminated.

3.4.3 Command M-E and Command Processors

The Command M-E is undergoing design; information will be supplied in a future update to this document.

3.4.4 Log M-E

The Log Management-entity is responsible for providing a service to all DI-resident software to generate log messages and ensure that they are written to mass storage for subsequent analysis. In Release 1, the physical writing of the log messages to mass storage is handled by a C170 mainframe program called the C170 Log Server, which is described later in this document.

The Log M-E also allows one or more network operators to receive a defined set of log messages as alarms at their consoles.

The following are examples of types of information that may be logged.

- Resource utilization and error statistics
- Hardware and software failures
- Configuration changes
- Accounting information

The type of log analysis to be done can vary, based on the needs of different sites; so rather than qualify each log message by its attributes (such as error statistics message, software failure message, etc.), CDNA does not call for any qualification of the log messages. Instead, a unique message number is associated with each potential log message. Qualification of log messages is left to analysis software, such as the Network Performance Analyzer.

CDNA defines a concept called "Log Groups." A Log Group contains a list of log message numbers and identifies the destination for these messages. A set of Log Groups is associated with each system that can generate log messages. Information in the Log Groups is used to control the generation and transmission of log messages.

The Log M-E is implemented as two parts:

- Dependent Log M-E
- Independent Log M-E

A Dependent Log M-E exists in each DCN system. An Independent Log M-E exists only in those systems that have access to mass storage. In Release 1, the Independent Log M-E functions are implemented by software components that reside in a C170 MDI and connected C170 mainframe.

Each Dependent and Independent Log M-E belongs to one or more Log Groups, each of which is identified by a Log Group Number. Each Log Group contains a list of log messages included in it. This list is implemented as a bit map of all log messages included in the group. If a bit is set in the bit map, the corresponding log message is included in that Log Group. The Dependent Log M-E supports on-line commands to allow an operator to add or delete log messages in a Log Group.

3.4.4.1 Dependent Log M-E

A Dependent Log M-E exists in each DCN system and provides a service to other software components in that system for generation and transmission of log messages.

As a part of system configuration, the Dependent Log M-E is given a list of the Log Groups to which it belongs. This list is a set of integers (n_i), each of which identifies a Log Group. The Dependent Log M-E combines each such integer with the character string, "Log M-E Group" to generate a Log Group Title of the form "Log M-E Group n_i. The Dependent Log M-E requests a Translation of each such Title to obtain the address of the Independent Log M-E to whom the log message in the associated Log Group should be sent.

The Dependent Log M-E establishes a Transport connection with one Independent Log M-E in each Log Group. The Independent Log M-E uses the connection to send the log message number bit map for the Log Group to the Dependent Log M-E. The Dependent Log M-E uses the connection to transmit log messages to the Independent Log M-E.

The Dependent Log M-E maintains a table called the Correspondent Data Store, which contains an entry for each Log Group to which the Dependent Log M-E belongs. Each entry contains the bit map of the log messages belonging to that Log Group. Each entry also contains address(es) of the Independent Log M-E(s) that belong to that Log Group, one of which is identified as the one being currently used (via Transport connections) for transmission of log messages.

When the Dependent Log M-E receives a request for the generation of a log message, it checks each Correspondent Data Store entry to identify the Log Group(s) that include that log message. The log message is then transmitted to the Independent Log M-E associated with the identified Log Groups.

3.4.4.2 Independent Log M-E

The Independent Log M-E function is implemented by two software components. One component, called the C170 Log Server, resides in the C170 mainframe. The other component, called the Independent Log M-E, resides in the MDI connected to the C170 mainframe. These two components interface with each other via a C170 Network Products A-A connection.

As a part of system configuration, the Independent Log M-E receives a list of Log Groups to which it belongs as well as the bit map for log messages in each Log Group. For each Log Group n_i in this list, it registers the following Title in the Directory:

"LOG M-E GROUP n_i"

The Independent Log M-E opens a Transport SAP and accepts connection requests from Dependent Log M-Es that belong to one or more of its Log groups. Once the connection is established, the Independent Log M-E transmits group bit maps to the Dependent Log M-E for the Log Groups belonging to that Dependent Log M-E. It receives log messages from the Dependent Log M-Es on the Transport connections and delivers the log messages to the C170 Log Server.

The Independent Log M-E also provides an alarm service to the DCN network operators. The process used to provide the alarm service needs to be defined.

3.4.5 File Access M-E

The File Access M-E provides components in CDNA systems with access to secondary storage files; it allows components to behave as if they had local system access to those files.

The File Access M-E consists of two parts:

- Dependent File Access M-E (DFA), resident in every DI in the catenet. To software components within a DI, the DFA appears to be providing access to local secondary storage .

- Independent File Access M-E (IFA), resident only in DIs that have a system-specific interface to secondary storage. In Release 1, only C170 MDIs are capable of being optionally configured with an IFA. At least one MDI within the catenet must contain an IFA.

3.4.5.1 Dependent File Access M-E (DFA)

The DFA provides two types of primitives to components in its local system:

- file-specification primitives

  Open File
  Create File
  Delete File

- file-operation primitives

  Read File
  Write File
  Position File
  Close File

During initialization or reconfiguration, an IFA registers a "file-type Title" for each type of file that it supports. File-type Titles are used to control the location(s) within the DCN of particular types of files.

When a file-specification primitive is issued, its parameters must include both a file-type Title and a file-name. For example, a component responsible for generating memory dumps as part of error processing would issue a Create File request to the DFA, supplying a file-type Title indicating "dump file" service. Since a DFA has no direct access to secondary storage, it must locate an IFA that supports the requested file-type. The DFA uses the supplied file-type Title in a Translation request and is returned a corresponding network address.

The DFA then uses that IFA network address to establish a Transport connection with the IFA. The Transport connection is used by the DFA and IFA to exchange File Access Protocol Data Units.

Once the requesting software has been notified that the requested file has been opened or created, file-operation primitives can be used to manipulate the file. The Close File request causes termination of the Transport connection.

The Delete File request causes both the establishment and termination of the Transport connection and hence is considered to be a file-specification primitive.

3.4.5.2 Independent File Access M-E (IFA)

In Release 1, IFAs will reside only in C170 MDIs. The IFA does not access secondary storage directly, but instead communicates over a C170 Network Products connection to a File Server application on the C170 host. The File Server application:

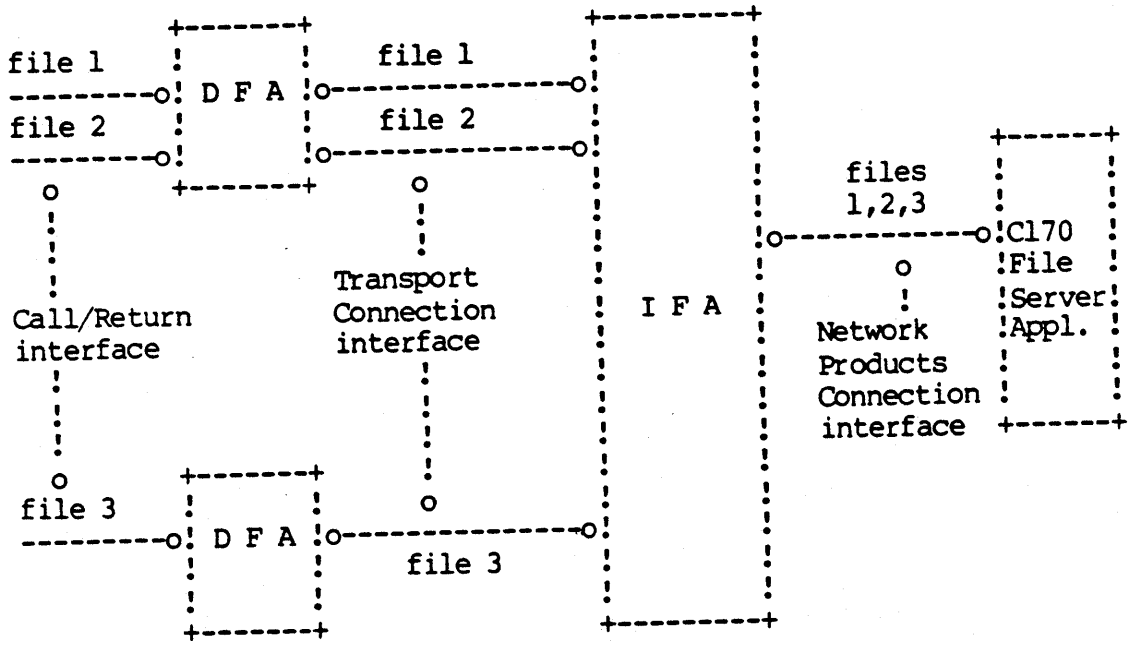
receives file access requests from the IFA

transforms them into NOS file access requests

sends the results back to the IFA

All file access requests/responses are multiplexed onto one Network Products connection. This connection is initiated by the IFA during IFA initialization and remains active during the life of the IFA.

The following block diagram illustrates the software processes and interfaces involved when an IFA is supporting two active file accesses from one DFA and one active file access from another DFA.



Each IFA in the network can be configured to support any subset of the supported file-types. The configuration can be dynamically altered via configuration commands. The IFA registers a separate Title for every file-type supported. Each Title is associated with the IFA's network address.

3.4.6 Error M-E

The Error M-E exists in all DIs to provide Internet and Internet users with the ability to create Internet Error Reports (IERs) and send them to the originator of the message that caused the error. The Error M-E receives directives via intertask messages, usually from Internet, and sends IERs using Internet. The Error M-E can write IERs to log files.

The Error M-E facility is provided via an intertask message interface. An error number, error parameter and 3B SAP ID of the entity detecting the error are specified along with the erroneous 3B PDU.

An IER is created and sent to the message source, which is determined by examining the Internet header of the message in error. The IER is a 3B PDU, consisting of an error description field prefixed to the first 42 bytes of the message in error.

Certain IERs, listed below, are inappropriate to be returned to the message source; these are written to a log file instead. The Error M-E uses the Independent Log M-E (aka Log Support Application) to issue log requests under the following conditions:

- Checksum error - since the integrity of the Internet header is in doubt, the message source address may be incorrect.
- Multicast/broadcast - since one multicast may result in several data indications, IERs are not sent back to the source.
- IERs - since the Error M-E generates IERs, it makes no sense to send IERs due to the receipt of bad IERs.

3.4.7 Echo M-E

The Echo M-E provides the ability for an Internet user to send a message to a particular system and have it returned. The user of Echo M-E services issues a normal Internet datagram request to send a 3B PDU to the Echo 3B SAP in any DI in the catenet. Either datagram or broadcast/multicast indications may be processed by the Echo M-E;;in either case, datagrams will be returned to the sender.

The 3B PDU used for the Echo M-E is a normal 3B PDU with an "echo-operation" field prefixed to the text portion of the message. The echo-operation field indicates whether the operation is a request or a reply.

A 3B PDU received by the Echo M-E is sent, via Internet, to the message source if the echo-operation field specifies "echo-request." The 3B PDU will be returned with echo-operation set to "echo-reply.""

If the echo-request message is checksummed, the echo-reply message wil also be checksummed.

The Echo M-E uses Internet to send the 3B PDUs back to the message source.

3.4.8 Initialization M-E

The Initialization M-E provides the services needed to dump and load DIs. The Initialization M-E is implemented in two parts:

- Dependent Initialization M-E (MPB ROM Bootstrap) - resides in every DI's ROM.
- Independent Initialization M-E - resides in the C170 and in each loaded/operational MDI and NDI.

The functions of the Initialization M-E were described previously as part of the DI load & startup process.

3.5 DCNS AND THE C170

DCNS and the C170 together provide the following functions:

- support of C170 application-to-application (A-A) connections
- support of terminal-to-C170-application (T-A) connections
- network management via a C170 host console.
- access to C170 secondary storage files for loading, dumping, log information storage and other network management functions.

Primary considerations in the design of the DCN interface to C170 Network Products were:

- accommodation of future features such as C170/C180 A-A communications - C170/C180 A-A communication will, in future releases, be accomplished by terminating all Network Products logical links in the MDI. The Network Products Connection and Block Protocols are never carried across the DCN. C170 application data traverses DCN as the data portion of DCN higher or middle layer protocol data units. Since the C180 supports the same protocols as DCN, C170/C180 A-A communication requires only an agreement on application level protocols.

- transparency of the interface to the C170 - The C170 MDI presents a local "255X NPU-like" interface to the C170. Physical and link protocols at the channel level are unique to DCN, but the Network Products Connection and Block Protocols have been preserved.

  The C170 MDI is configured in the Network Definition Language (NDL) as a local NPU. A C170 MDI is capable of supporting multiple logical links, but each logical link requires a separate CYBER channel and a separate MCI card in the MDI.

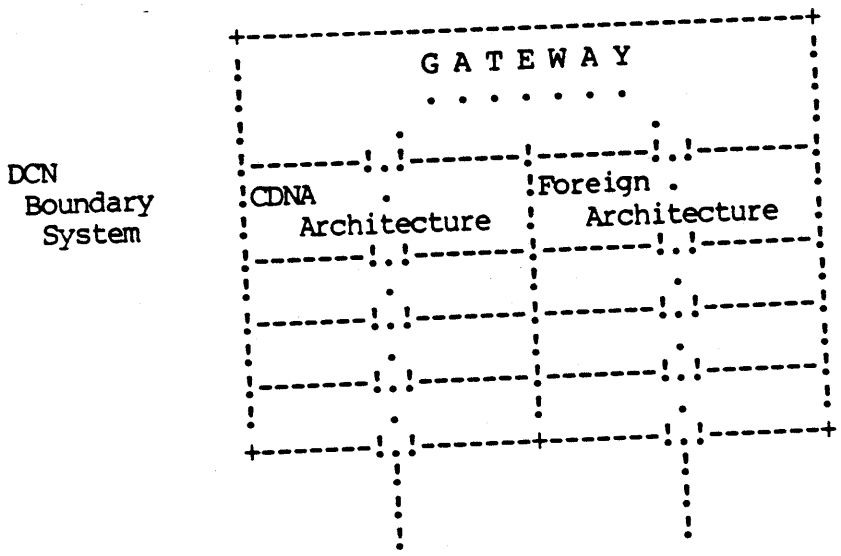
- provision for support of all existing C170 networking features - Existing C170 Network Products features are supported in the MDI by "NAM-level" software that is capable of processing the Network Products (NP) Connection and Block Protocols in support of A-A and T-A connections. The MDI contains a "gateway" function to interface DCN connections to NP connections.

3.5.1 The Gateway Function and NP Transforms

The gateway function is provided by software entities that reside in DCN "boundary systems;" boundary systems are DIs that connect a CDNA network with a non-CDNA network. Boundary systems understand both architectures and are a member of both networks.

- C170 MDIs reside at the boundary between CDNA and C170 Network Products and provide the NP gateway function, which is described in this section.

- NDIs configured with an X.25 interface reside at the boundary of CDNA and the architecture defined by CCITT Recommendation X.25; NDIs so configured provide the X.25 gateway function, which is described in the section entitled "DCNS and X.25."



3.5.1 The Gateway Function and NP Transforms (continued)

The gateway function in the C170 MDI is provided by software components called "NP Transforms." NP Transforms interface C170 NP Connection/Block protocols to DCN Connection and Data Transfer protocols; thus, no special software in the C170 is required to support these connection types across the DCN. The term "transform" is used in this document to mean a specific component that provides a gateway function; elsewhere, "transform" and "gateway" are used synonymously.

An NP Transform is defined for each of the following connection types:

- A-A Transparent (Release 1)
- T-A Interactive (Release 1)
- T-A Batch (future release)

Each NP Transform is a separate entity and each maintains an entry in the network Directory. Each NP Transform provides a unique service for accessing a specific C170 host in the DCN network. NP Transform names reflect the type of data traffic they support relative to the higher layer protocols involved in the transmission of the data.

The A-A Transparent Transform deals with the transmission of A-A data and does not require the services of any higher layer protocols; hence, the name "Transparent".

The T-A Interactive Transform deals with the transmission of interactive data and requires the services of the ITS higher layer.

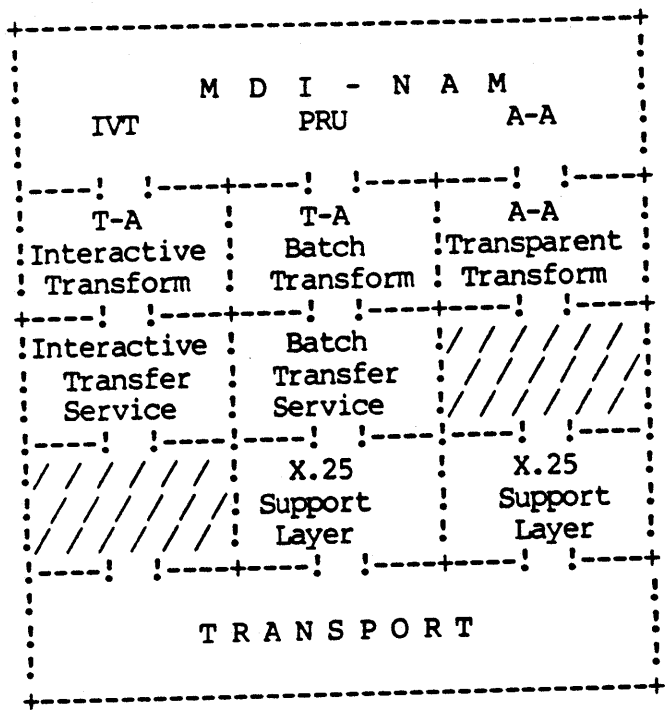
The T-A Batch Transform deals with the transmission of batch data and requires the services of the BTS higher layer. This NP Transform will be provided in subsequent releases.

NP Transforms cannot be used to select a specific C170 host application. C170 host application selection/switching is completely within the realm of the C170.

Each NP Transform maintains one DCN connection/association for each active Network Products connection and must therefore be knowledgable in the connection management protocol of Network Products and one connection management protocol of DCN.

The following diagram illustrates the relationship of the NP Transforms to other system components.

3.5.1 The Gateway Function and NP Transforms (continued)



During initialization, each NP Transform requests access to the appropriate underlying service by "opening a SAP;" the underlying service is identified by the "directly accessible service" field. In response to the open SAP request, the NP Transform receives a 3B SAP address. The NP Transform then notifies other network components of its availability by registering itself with the Directory M-E, using the following fields:

- Title - the ASCII equivalent of the "host node" of the logical link between the C170 host and MDI; NP Transforms in any given MDI all register the same Title.
- Network Address - the Network ID and System ID of the local system plus the 3B SAP ID as received in response to the open SAP request.
- "directly accessible service" - identifies the NP Transform within an MDI by its underlying service.

3.5.1 The Gateway Function and NP Transforms (continued)

The "directly accessible service" field serves two purposes:

- explicitly identifies the CDNA end-to-end protocols that are to be used when communicating with the associated NP Transform.

- implicitly identifies the unique function that the NP Transform performs. For example, a "directly accessible service" of ITS implies that the associated NP Transform supports T-A Interactive.

The table below lists directory entries for a C170 MDI that supports all three possible Transforms. The host node number in this example is assumed to be "1D."

TRANSFORM NAME	TITLE	TRANSPORT SAP ADDRESS	DIRECTLY ACCESSIBLE SERVICE
T-A Interactive	"1D"	system-assigned	ITS
T-A Batch	"1D"	system-assigned	BTS
A-A Transparent	"1D"	system-assigned	X.25 Support Layer

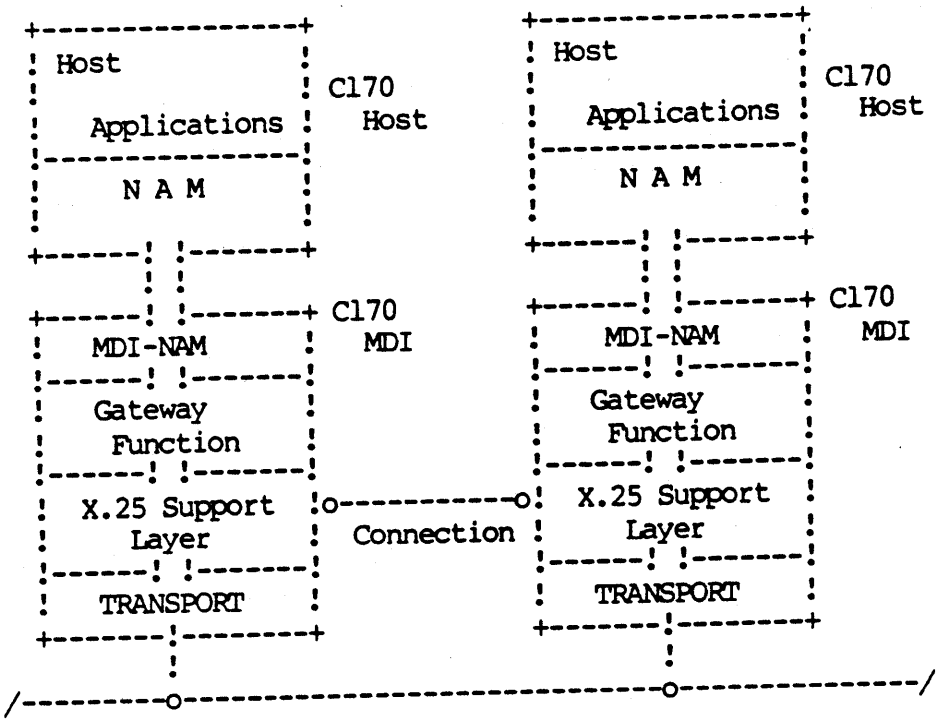
Any network components wishing to communicate with an NP Transform must first obtain the network address from the Directory. For DCN release 1, an application may initiate contact with another application and a terminal may initiate contact with an application; therefore, the software entities that query the Directory are:

- Terminal Support Software in TDIs must determine the existence and network address of T-A Transform(s) prior to establishing higher layer associations in support of terminal devices. Terminal Support Software is discussed in the section entitled "DCNS and Terminals."

- A-A Transparent Transforms must determine the existence and network address of other A-A Transparent Transforms prior to establishing connections in support of C170 A-A transfers across DCN.

3.5.2 C170 A-A Connections

C170 A-A connections are supported via an X.25 Support Layer connection between gateway functions in two C170 MDIs, as illustrated below.



Each C170 host is interfaced to the DCN via a C170 MDI. An NP connection exists between the gateway function in each MDI and its associated C170 host. Gateway functions communicate with each other using an X.25 Support Layer connection.

The C170 A-A connection can be initiated by either host. The C170 MDI of the calling host selects the called host based on information in the Networks Products A-A outcall. Application selection is processed by the Network Validation Facility (NVF) in the called host.

C170 applications refer NAM to a predefined OUTCALL block in order to identify the host to be connected. The "dhost" field in the OUTCALL block contains the ASCII destination host node number. The "dhost" field is included in the text portion of the A-A connection request service message sent from NAM to the MDI.

The A-A Transparent Transform uses the "dhost" string as a Title for querying the directory for the network address of another A-A Transparent Transform that previously registered that Title with a "directly accessible service" of X.25 Support Layer.

3.5.2 C170 A-A Connections

Because an A-A connection can be initiated in either host, the A-A Transparent Transform must be capable of processing connection indications initiated by either C170 Network Products or DCN. Likewise, the Transform must be capable of making connection requests to either network.

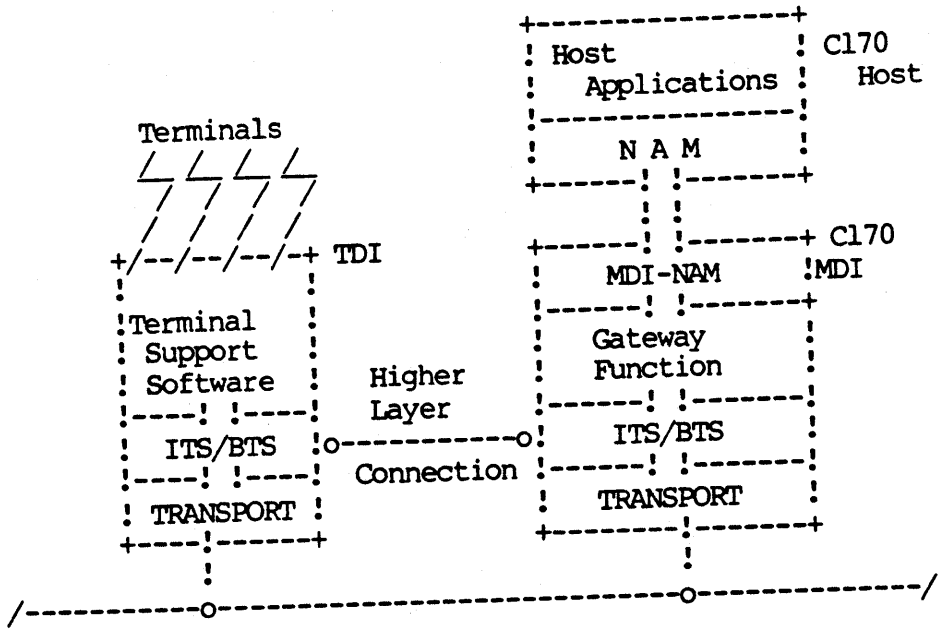
When connection processing is complete, the A-A Transparent Transform maintains two paired connections, both in the data transfer state. Data and control information arriving on either connection must be mapped into corresponding data and control information and sent on the paired connection. The A-A Transparent Transform is only involved in protocol conversion when processing data and qualified data.

The Network Products Data Block Clarifier (DBC) is removed from all data and qualified data received on the Network Products connection before the data or qualified data is sent on the DCN connection. Likewise, a DBC is constructed and prefixed to all data and qualified data received from DCN prior to sending to Network Products. The DBC removed/appended by the A-A Transparent Transform always has only the Transparent bit set.

A connection termination message received on either connection causes the Transform to send a corresponding connection termination message on the paired connection.

3.5.3 Terminal-to-C170-Application (T-A) Connections

Terminals accessing a C170 host are supported via a DCN higher layer connection between Terminal Support Software in the TDI and the gateway function in the C170 MDI, as illustrated below.



Terminals are interfaced to the DCN by Terminal Support Software in TDI components. The Terminal Support Software maintains a higher layer connection to a T-A gateway function in a C170 MDI, which interfaces the higher layer connection to an NP terminal connection. Host selection is processed by DCN, based on terminal user input. Application selection is processed by NVF in the C170 host.

Since the Network Products design requires that all T-A connections be initiated by terminals, the T-A gateway functions need only recognize connection indications initiated by DCN. The DCN terminology for these higher layer connections is "associations".

Upon receipt of a Create Association indication, the T-A gateway function establishes a corresponding NP connection. Like the A-A Transparent Transform, both T-A Transforms maintain a pairing between association and connection, with the T-A Interactive Transform maintaining a pairing between an ITS association and an IVT connection and the T-A Batch Transform maintaining a pairing between a BTS association and a PRU connection.

3.5.3 Terminal-to-C170-Application (T-A) Connections (continued)

Once the association/connection pairing is established, the T-A gateway function receives data and control information on an association/connection and sends corresponding data and control information on the paired association/connection.

Unlike the A-A Transparent Transform which transmits transparent data, the T-A Transforms must provide mappings between dissimilar data presentation protocols. As such, their protocol conversion functions are somewhat more substantial.

The T-A Interactive Transform does have a relatively simplistic mapping between IVT commands and ITS command services, because the initial ITS design is based on C170 IVT design with the mapping between the two a significant consideration.

The conversion between IVT and ITS data units is more complex, involving IVT DBC creation based on ITS header and ITS header creation based on IVT DBC.

The initial release of DCN, supports both edited terminal input and transparent terminal input. Physical representation of these two data formats are the same in both IVT and ITS data units. The initial implementation of the T-A Interactive Transform does not have to be concerned with any text processing functions that would be required in converting from one data format to another.

T-A Batch Transform functions are not as yet completely defined. T-A Batch support is not a DCN Release 1 feature.

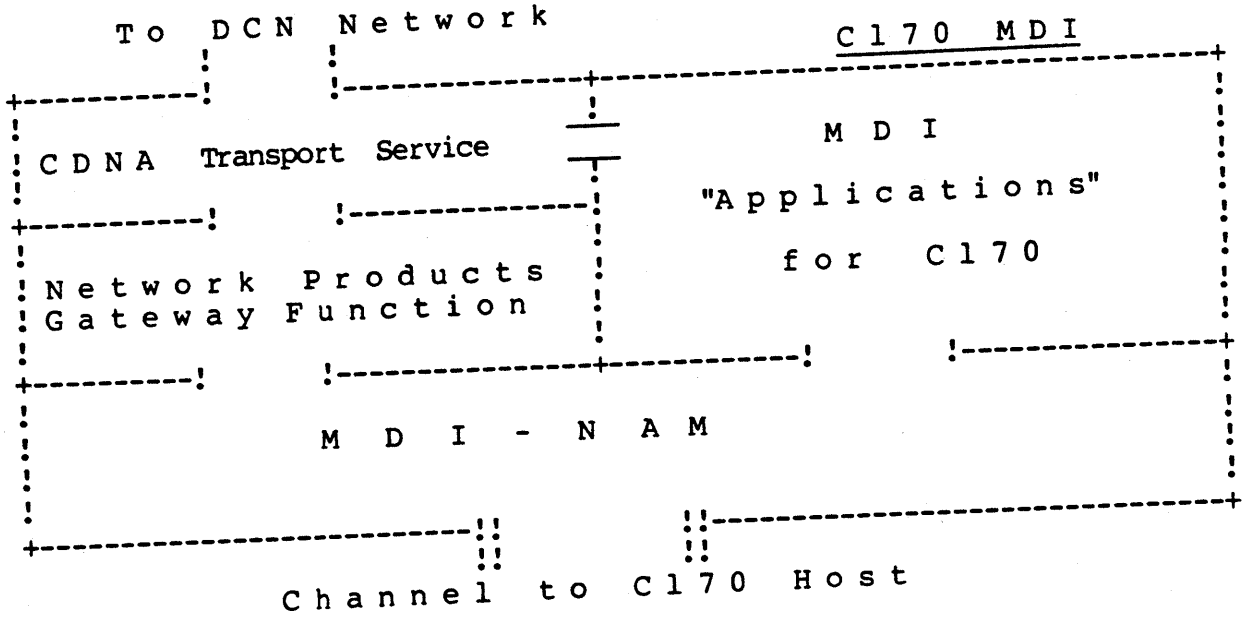
3.5.4 MDI-resident Software

MDI-resident software within the C170 MDI serves two purposes:

it allows user access, via the DCN, to applications and features in the C170 host.

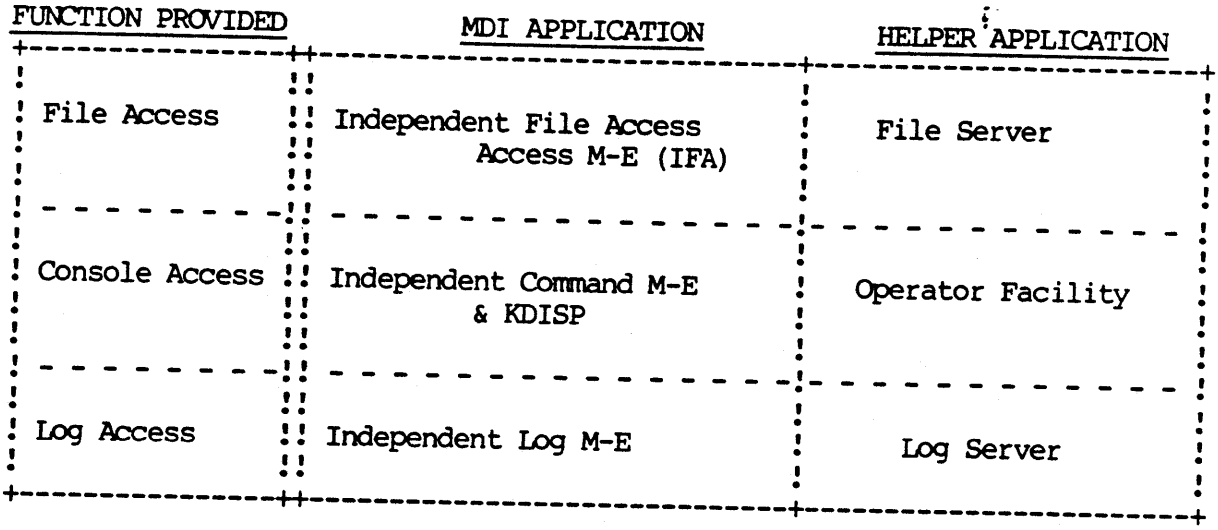
it provides the DCN itself with access to C170 secondary storage and host console. The DCN does not support its own secondary storage or operator console for network control functions at Release 1 time.

The primary component of the MDI-resident software is called MDI-NAM. MDI-NAM interacts with its "users," as depicted in the figure below.



Implementation of MDI-NAM made it possible to define the concept of an MDI "application" within the MDI. An MDI "application" uses the services of MDI-NAM to communicate with a corresponding helper application within the C170 host. These application pairs communicate over Network Products A-A connections using Network Products protocols and serve to provide access to C170 host resources for the DCN. The three application pairs defined for Release 1 are as follows:

3.5.4 MDI-resident Software (continued)



Note that the above application pairs are implemented such that the MDI "application" half of the pair is physically contained partially or completely within an independent management-entity rather than existing as a separate module; the concept of MDI "application" is retained simply to provide a mental model of the functionality and to maintain consistency with existing overview documentation of the system.

Additional information on MDI "applications" appears later in this section and in the section on "Network Management."

Helper applications are described in the section on "C170-Resident Software."

The NP gateway function uses MDI-NAM to map Network Products protocol to and from the X.25 Support Layer protocol; i.e., the NP protocol is said to be "terminated" in the MDI.

3.5.4.1 MDI-NAM

The C170 Network Products (NP) Block Protocol defines the concept of "logical links" imposed upon physical links; for example, a logical link between a host and frontend 255X NPU is imposed upon the physical link consisting of a CYBER channel. Each logical link contains up to 256 logical connections, one of which is known as the service channnel. The service channel is used to communicate commands to establish and terminate other connections and to regulate the logical link; the other 255 connections are used to transmit data between communicating applications.

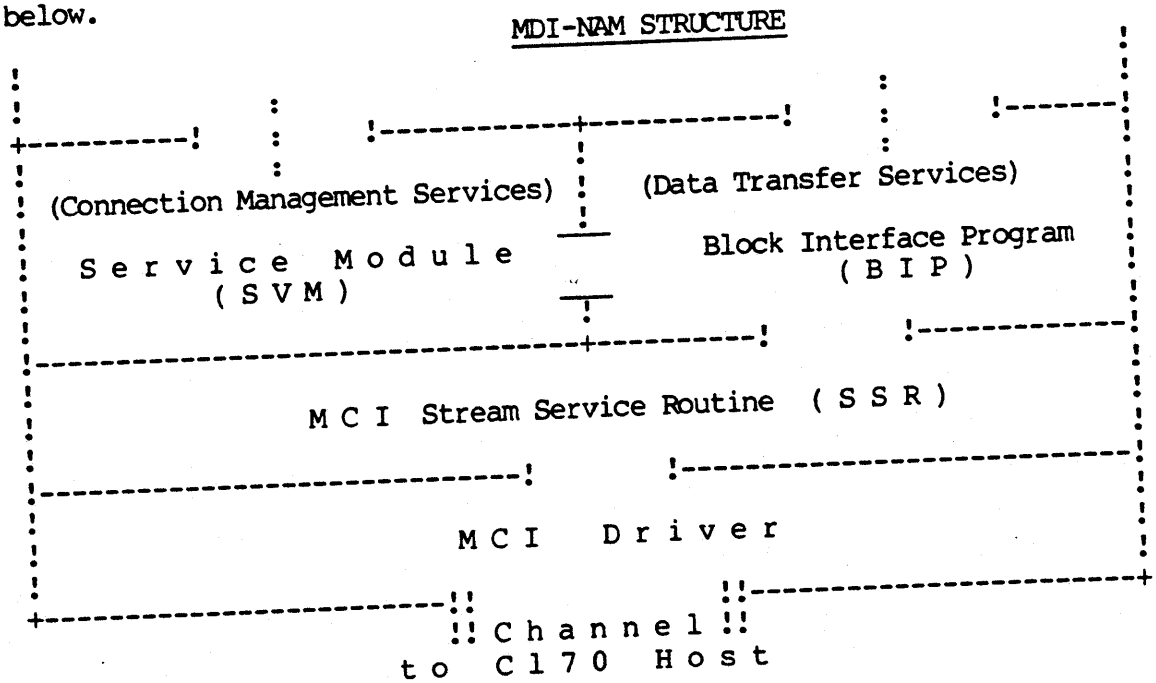
MDI-NAM presents virtually the same interface to the C170 host as does the 255X Communication Control Program (CCP); in fact some of the CCP module names and functions are carried over into MDI-NAM.

The MDI Block Protocol Interface Program (BIP) and Service Module (SVM) use the NP Block Protocol to communicate with NAM in the C170 host. The protocol used by the MDI SVM is a subset of that used by the CCP SVM; no special C170 software is required to interface to the MDI SVM and MDI BIP.

MDI-NAM does, however, use a new communications protocol at the link level. This protocol is processed by the MCI Stream Service Routine (MCI SSR) and MCI driver in the MDI and by an enhanced version of the 255X PIP in the C170 PPU.

MDI BIP depends upon the services of the MCI SSR and driver for physical transmission of data across the coupler to the channel-connected C170 host. MDI BIP and SVM add to the services of the MCI SSR to provide a "transport-like" service for their "users," the MDI "applications" and the NP gateway function. This interaction is illustrated in the figure below.

MDI-NAM STRUCTURE



/0068y

3.5.4.1 MDI-NAM (continued)

Services provided by SVM include:

Logical connection management - accomplished through SVM's interfaces to BIP and to its "users." "Users" call SVM to establish and terminate connections with NAM; SVM then calls BIP to send the necessary service messages and protocol elements to NAM. Messages from NAM are passed to SVM by BIP.

Logical link status maintenance - MCI-SSR will inform SVM of a change in the status of the logical link by building a service message and routing it through BIP to SVM. SVM will change the status of the Logical Link Control Block for that logical link.

BIP communicates with NAM in the C170 host via the NP Block Protocol to provide data transfer services across an established Network Products connection. BIP additionally supports a local interface with SVM for the exchange of connection management messages between SVM and NAM.

All messages from the C170 to the MDI arrive at BIP from the MCI-SSR. Connection management and logical link status messages are delivered to SVM for processing. All other messages are delivered to the appropriate "user."

All messages from the MDI to the C170 leave BIP through the MCI-SSR. Such messages may originate either from SVM in the case of connection management messages or from a "user."

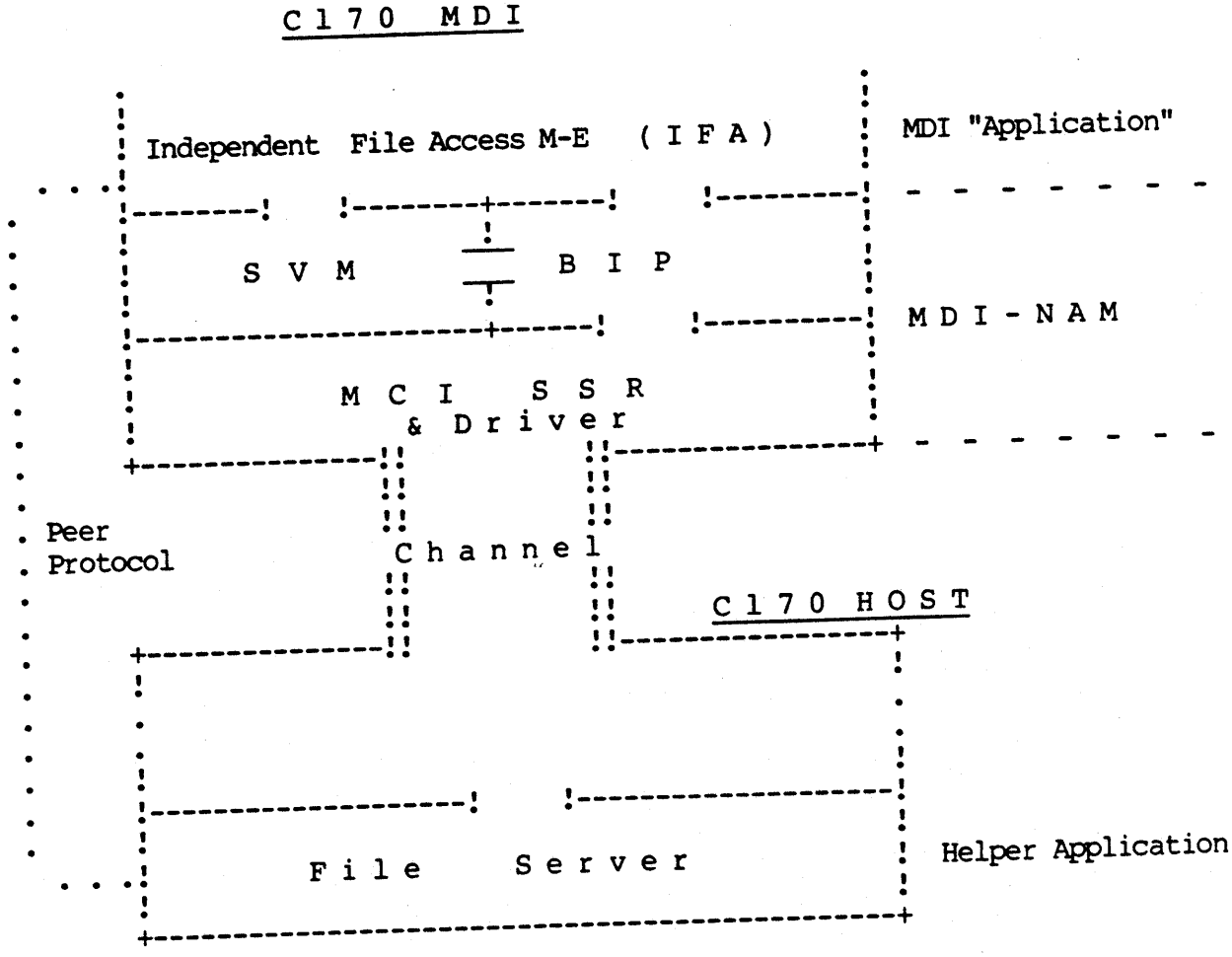
3.5.4.2 MDI "Applications"

MDI "application" is a logical concept that does not necessarily imply existence of a separate physical entity. It was more efficient, in some cases, to implement MDI "application" functionality as part of a management-entity.

MDI "applications" reside within a C170 MDI and communicate with peer helper applications in the C170 host via a Network Products A-A connection. MDI "applications" and their C170 host helper peers support three network management functions, for which the DCN is dependent on external sources.

File Access

Access to C170 files is provided by the Independent File Access (IFA) management-entity. IFA is described in the section entitled "File Access M-E." The MDI-resident IFA program provides the DCN interface to secondary storage for all DCN files except Log files.



3.5.4.2 MDI "Applications" (continued)

Console Access

The NAM K-Display at the C170 console is used to manage the DCN. DCN commands are entered and command responses are received by the operator at the NAM K-Display.

Access to the NAM K-display, is provided by two modules resident in the MDI:

the K-display Supervisor (KDISP)

the Independent Command M-E

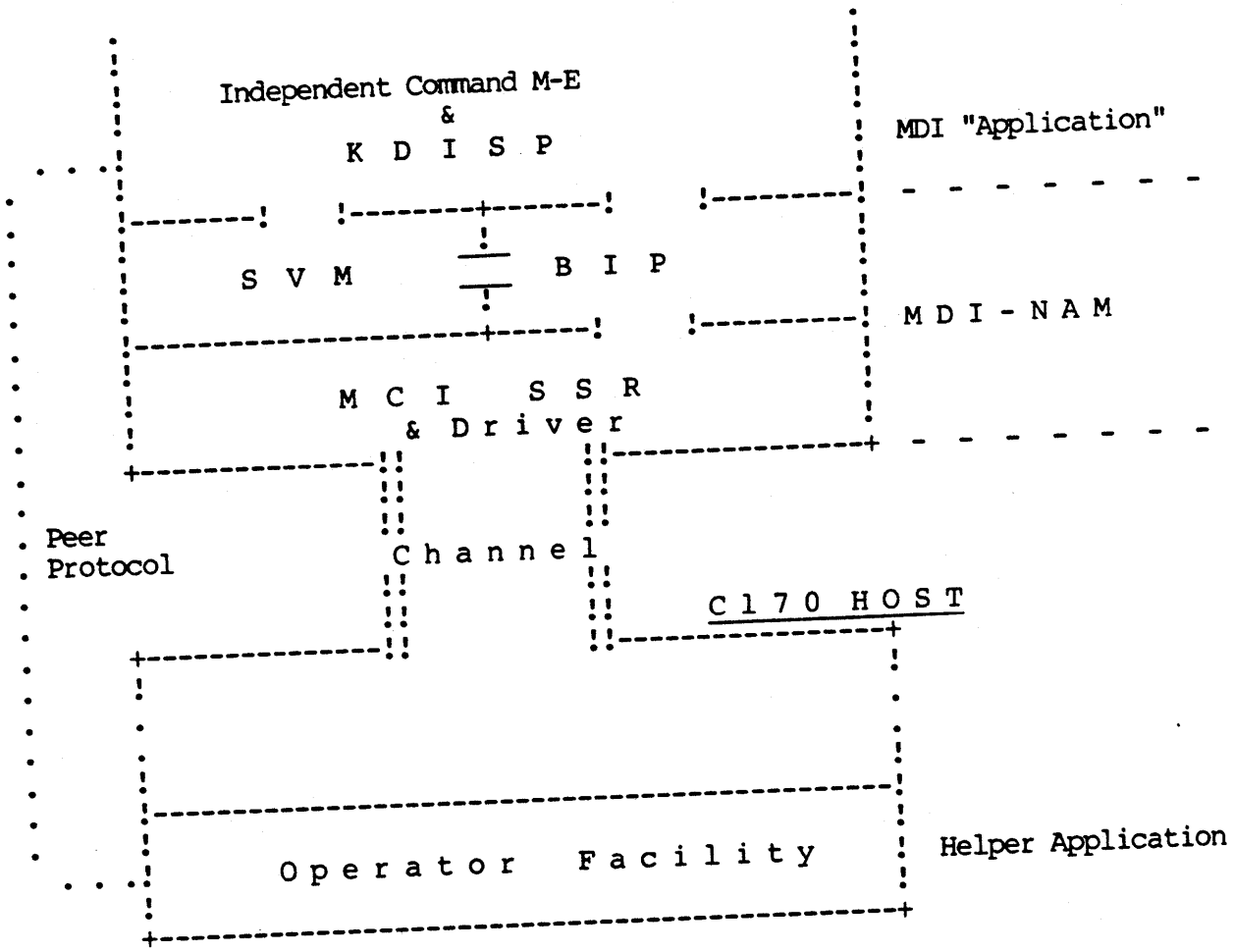
For purposes of discussion within this section of the ERS, these two modules will be considered jointly to be the MDI "application" for console access, as illustrated below. (It might be noted here that, in other DCN documentation, the Independent Command M-E is occasionally called the "Operator Support Application (OSA)" in recognition of its dual role as both a management-entity and an MDI "application.")

KDISP establishes and supports a Network Products A-A connection, using SVM, to communicate with the C170 Operator Facility helper application. During initialization, KDISP initiates an A-A connection request, supplying INCALL application name information to identify the Operator Facility as the requested helper application. The Operator Facility is started by NAM and the connection is brought to a fully operational data transfer state. KDISP will reinitialize the connection if it detects an A-A protocol error or if it receives notification of an unsolicited TCN/AP/R connection termination.

The Independent Command M-E (aka Operator Support Application) in the MDI uses KDISP to transmit command traffic between the network and the C170 console, as described in the section entitled, "Command M-E."

3.5.4.2 MDI "Applications" (continued)

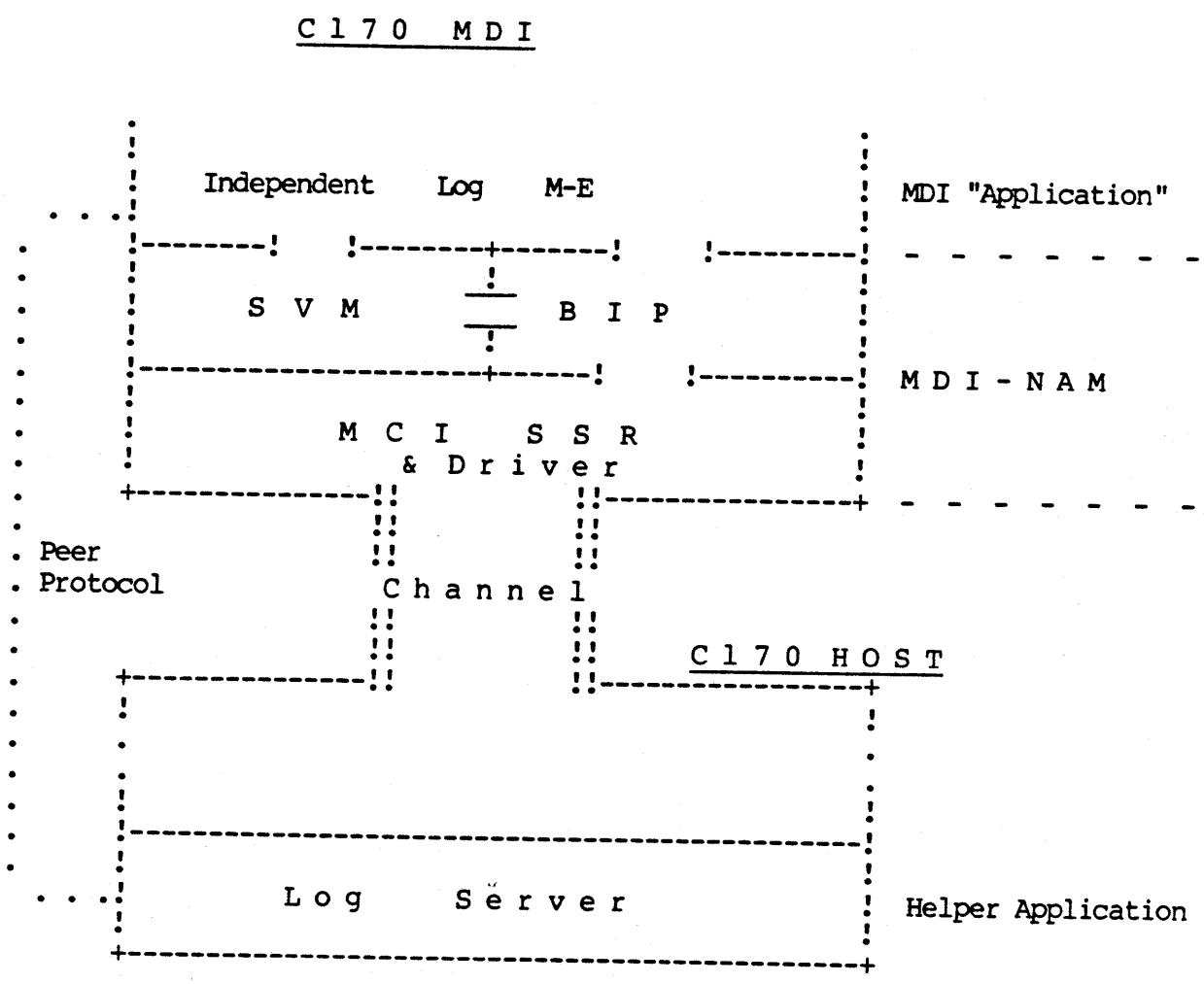
Console Access (continued)



3.5.4.2 MDI "Applications" (continued)

Log Access

Access to C170 secondary storage for log information is provided by the MDI-resident Independent Log M-E; a description of the Independent Log M-E may be found in the section entitled "Log M-E."



3.5.5 C170-resident Software

This section describes C170-resident software components that relate to the operation of DCN. C170-resident software provides the following functions:

- builds load and configuration files for the DCN
- loads a C170 MDI as the first step in loading the network
- provides network management function support via C170 helper applications
- analyzes DCN network performance
- analyzes DI dumps

3.5.5.1 Load and Configuration Files

(****TO BE SUPPLIED****)

3.5.5.2 MDI Loading

INITMDI is the C170 software responsible for the initialization of channel-connected C170 MDIs. INITMDI functions as an independent Initialization M-E and is comprised of a C170 CP program and a PP helper. The CP program is activated as the result of an initialization request from the MDI. The CP program is responsible for executing the Initialization protocol with the MPB ROM bootstrap of the requesting MDI. The CP program requests the PP program for sending and receiving initialization protocol data units.

INITMDI uses the NOS EST interlock to lock out all other C170 processes from trying to use the loading MDI. Once the entire load file is transferred to the MDI, the EST interlock is cleared and the EST for the MDI enters an operational state. The operational state is noted by NAM and a PIP is assigned to the MDI. Error and status information pertinent to initialization are reported to the operator by an INITMDI interface to the NAM K-Display.

3.5.5.3 C170 Helper Applications

The Operator Facility maintains an A-A connection with the MDI "application," KDISP. Together, these two provide the capability of managing the DCN from the NAM K-Display.

The Operator Facility is initialized when NAM receives an A-A incall from the KDISP MDI "application." AOF validates the connection by requiring that a specific text message be received from the calling application as the first data unit after the connection is established. After validation, the connection can be used to send DCN commands and receive DCN command responses. The operator uses standard NAM K-Display procedures for interfacing to the Operator Facility.

The File Server Facility in the C170, along with the independent File Access M-E in the MDI, provides secondary storage access capabilities for DCN. The File Server is capable of providing multiple simultaneous file accesses to the same or different NOS-resident files. All file accesses are multiplexed across one A-A connection. File uniqueness is accomplished by assignment of a file identifier (FID) to each active file. The FID is assigned by the independent File Access M-E when the file is opened. The FID remains associated with the file until the file is closed.

The File Server supports the file functions of open, create, delete, read, write, position, and close.

NOS-resident files are used by DCN to support the network management functions of DI loading, DI configuration, DI dumping, and network user validation.

The Log Server and the independent Log M-E in the MDI provide a network logging and alarm capability. The Log Server receives log messages from the independent Log M-E and writes them to a formatted log file.

The Log Server examines each log message to determine if it is also currently selected as a network alarm. In addition to being written to the DCN log file, alarm messages are displayed. The mechanisms for processing alarms are currently being defined.

3.5.5.4 Network Performance Analysis

The Network Performance Analyzer (NPA) is both a batch report generator and interactive processor of the DCN log file. Network management personnel use NPA to gather and analyze the performance aspects of the DCN. NPA generates reports to highlight trends that aid in early detection of imminent hardware failure, present and future configuration requirments, and network capacity, bottlenecks, and throughput.

3.5.5.5 DI Dump Analyzer

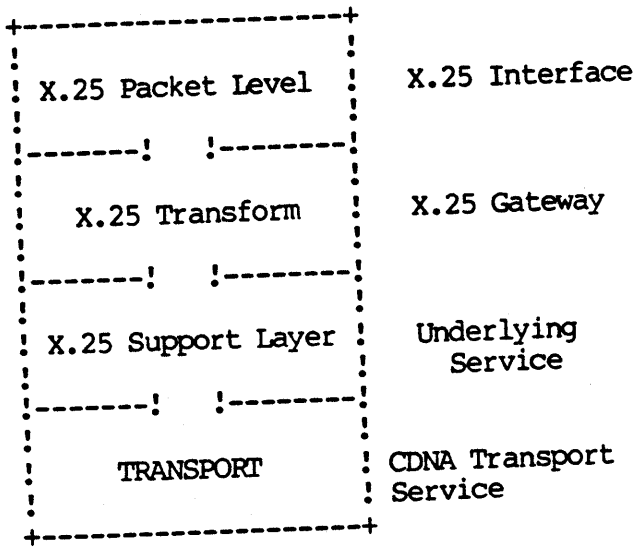
(****TO BE SUPPLIED****)

3.6 DCNS AND X.25

NDIs may be configured to interface DCN to an X.25 network, via a software component called the "X.25 Transform." The X.25 Transform interfaces X.25 virtual circuits and protocols to DCN X.25 Support Layer connections and protocols. The X.25 Transform performs the X.25 gateway function; the two terms can be used synonymously.

The X.25 gateway is unlike the NP gateway in that it does not contain multiple specialized components. The X.25 gateway only supports A-A Transparent connections.

The diagram below illustrates the relationship of the X.25 Transform to other system components.



Like NP gateway functions, the X.25 Transform must register its Title in the network Directory. A Title is required for every logical interface to an X.25 Public Data Network (PDN) or X.25 Trunk. A logical interface to a PDN or X.25 trunk is defined as one or more physical links to a PDN or X.25 trunk.

The Title is used to select a logical interface to a PDN or X.25 trunk, not a physical link; physical link selection is accomplished via specification of a source DTE address.

The Title is also not used for selecting a destination host or application; destination host selection is accomplished via the destination DTE address. Destination application selection is accomplished at the destination host.

3.6 DCNS AND X.25 (continued)

When C170 applications initiate an A-A connection that traverses both DCN and transparent X.25, the C170 OUTCALL block is configured with a "port" field that is mappable to the Title of the X.25 Transform providing the desired X.25 interface. Mapping of this parameter into the appropriate Title and searching of the directory will be performed by the NP gateway function in the MDI associated with the calling host.

The NP A-A Transparent Transform will obtain the "port" field from the text portion of an ICN/AP/R service message. If the value of the "port" field is zero, the requested connection is not to traverse a transparent X.25 interface; in this case, the NP A-A Transparent Transform will use the "dhost" field within the ICN/AP/R text as a Title for locating another NP A-A Transparent Transform.

When the X.25 gateway receives an incoming call indication, the eighth and ninth octets of the call user data are used as a Title for locating the NP A-A Transparent Transform that services the requested host. By Network Products convention, these two octets of call user data contain the ASCII equivalent of a C170 host node number; this convention will be used in the initial release of DCN so as not to affect the existing Network Products design in this area.

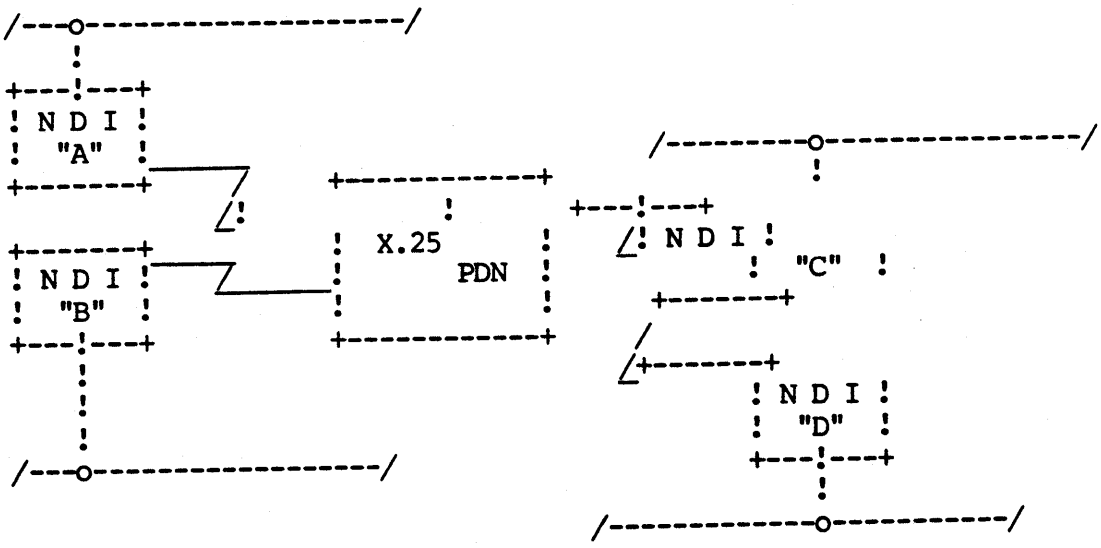
Because the external design of the X.25 Support Layer is almost identical to the external design defined by DCN for the X.25 Packet Level, the mapping of connection and data transfer protocols between the two entities is a relatively straightforward process. The X.25 Transform maintains connection pairings just as the NP gateway function does. The connection pair consists of an X.25 Support Layer connection and an X.25 virtual circuit. The X.25 Transform receives service indications on a connection/virtual circuit and issues the corresponding service request on the paired connection/virtual circuit.

In the initial release of DCN, an X.25 interface is available in support of both DCN itself and also in support of C170 A-A connections that traverse an intermediate X.25 virtual circuit link.

> DCN utilizes X.25 virtual circuits as CDNA network solutions, connecting two NDIs via a public data network (PDN). The virtual circuits carry DCN Internet datagrams, complete with all CDNA protocols.
>
> When X.25 virtual circuits are used as intermediate links in a C170 A-A connection, data units appearing on the virtual circuits are free of any CDNA protocols. Since only the application's data units themselves are present on the virtual circuit, the virtual circuit is commonly called a Transparent X.25 virtual circuit.

3.6 DCNS AND X.25 (continued)

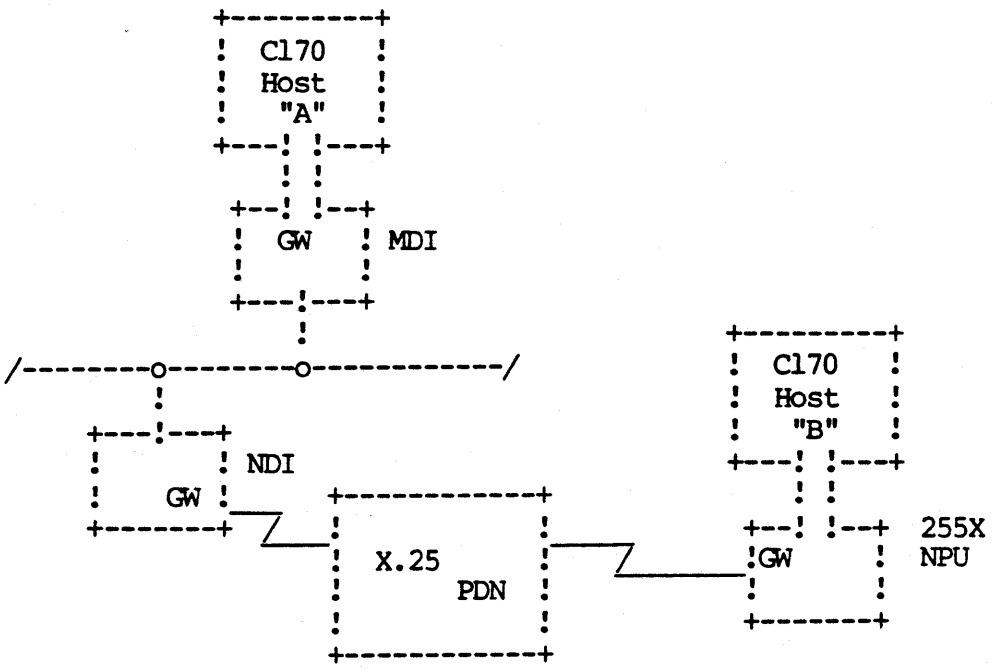
The diagram below illustrates use of X.25 virtual circuits as CDNA network solutions. Four NDIs are pictured, each with an X.25 interface to a common PDN.



Assuming the configuration is fully connected, each NDI pictured above maintains one Ethernet network solution and three X.25 virtual circuits as point-to-point network solutions; e.g., NDI "A" uses one virtual circuit to communicate with NDI "B", the second to communicate with NDI "C", and the third to communicate with NDI "D". These virtual circuits are used by Internet entities within the NDIs for sending and receiving Internet datagrams.

3.6 DCNS AND X.25 (continued)

The diagram below illustrates use of an X.25 virtual circuit as an intermediate link in a C170 A-A connection. A Transparent X.25 virtual circuit is always used by DCN when communicating over an X.25 interface with "foreign" systems; i.e., systems that do not implement the DCN protocols.



C170 host "A" is connected to an NDI that interfaces to X.25; C170 host "B" is connected to a front-end 255X that interfaces to X.25. The C170 A-A connection in the diagram is really comprised of four distinct intermediate connections joined together by gateway functions:

ENTITIES CONNECTED			INTERMEDIATE CONNECTION TYPE
C170 host/	MDI		NP connection
MDI	/	NDI	X.25 Support Layer connection
NDI	/	255X	Transparent X.25 virtual circuit
255X	/	C170 host	NP connection

Gateway functions exists in the MDI, NDI and the 255X. Since the 255X does not implement the DCN protocols, the NDI and 255X "gateway" communicate over a Transparent X.25 virtual circuit. Only the application's data units, or a portion thereof, appear in the packets that traverse the X.25 virtual circuit.

3.7 DCNS AND TERMINALS

DCN terminal support is implemented in a TDI component. Within the TDI, unique Terminal Interface Programs (TIPs) are required to support the various protocols/capabilities of the wide range of available terminal types. Despite dissimilarities between terminal types, many of the functions that TIPs perform are common. To take full advantage of this commonality and to provide a basis for common and simplified TIP design, a set of common TIP functions is implemented and called "Terminal Support Software."

Terminal users supply Terminal Support Software in the TDIs with host select information during initial interchange. The host select information is used to construct a Title that is the ASCII equivalent of the C170 node number. This Title is used for querying the network Directory for the Transport SAP address(es) of the T-A Transform(s) servicing the selected host.

- The Transport SAP address of the Transform that has a "directly accessible service" of ITS is used for establishing ITS associations in support of the terminal's interactive device(s).

- The Transport SAP address of the Transform that has a "directly accessible service" of BTS is used for establishing BTS associations in support of the terminal's batch device(s).

Terminal Support Software is implemented in two major modules:

- the Line Control Module (LCM)

- the Terminal Data Services Modules (TDSM).

A TIP interfaces to LCM and TDSM via shared control blocks that are created and deleted under the control of LCM subroutines. The control blocks are organized in a hierarchical fashion. At the highest level, the Configured Line Control Block (CLCB) maintains the configuration and current state of the line. Linked to the CLCB and at successively lower hierarchical levels are separate control blocks that maintain Cluster, Terminal Device, and Data Association information is used by both the TIPs and the Terminal Support Software.