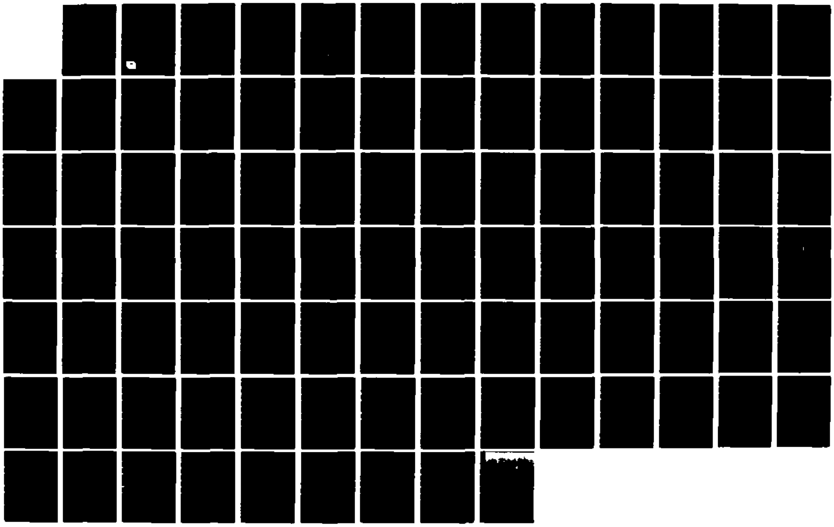
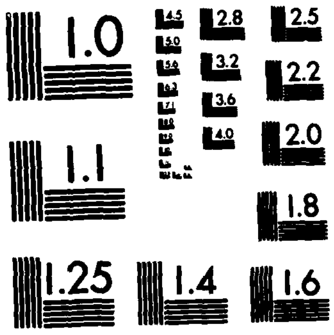
AD-A134 194 C3I LOCAL AREA NETWORKS -- AN ASSESSMENT(U) INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA SCIENCE AND TECHNOLOGY DIV T C BARTEE ET AL. JUN 83 IDA-P-1715 1/1

UNCLASSIFIED IDA/HQ-83-25416 MDA903-79-C-0320 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A134 194

Copy 10 of 60 copies

2

IDA PAPER P-1715

C^3I LOCAL AREA NETWORKS -- AN ASSESSMENT

T. C. Bartee
O. P. Buneman

June 1983

Prepared for
Assistant Secretary of Defense
(Communications, Command, Control & Intelligence)

INSTITUTE FOR DEFENSE ANALYSES
SCIENCE AND TECHNOLOGY DIVISION

83 10 26 032 IDA Log No. HQ 83-25416

The work reported in this document was conducted under Contract MDA 903 79 C 0320 for the Department of Defense. The publication of this IDA Paper does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER / 2. GOVT ACCESSION NO. AD-A134194	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) C^3I Local Area Networks -- An Assessment	5. TYPE OF REPORT & PERIOD COVERED Final -- Oct 1981-Sept 1982
	6. PERFORMING ORG. REPORT NUMBER IDA Paper P-1715
7. AUTHOR(s) T.C. Bartee, O.P. Buneman	8. CONTRACT OR GRANT NUMBER(s) MDA 903 79C 0320
9. PERFORMING ORGANIZATION NAME AND ADDRESS Institute for Defense Analyses, 1801 N. Beauregard Street, Alexandria, Virginia 22311	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Task Order D-32
11. CONTROLLING OFFICE NAME AND ADDRESS Director, Information Systems, OUSDRE (C^3I), The Pentagon, Washington, D.C. 20301	12. REPORT DATE June 1983
	13. NUMBER OF PAGES 85
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

None

18. SUPPLEMENTARY NOTES

N/A

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

local area networks, security, data base management systems, encryption, distributed data base, broadband, baseband, PBX, concurrency

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Command and control systems are now beginning to use local area network technology. In many of the applications, classified data must be handled. This report discusses specific DoD considerations for local area networks with emphasis on security. The accessing and updating of data bases in local area networks is also discussed and directions for further research and development presented.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

IDA PAPER P-1715

C^3I LOCAL AREA NETWORKS -- AN ASSESSMENT

T. C. Bartee
O. P. Buneman

June 1983

A

INSTITUTE FOR DEFENSE ANALYSES
SCIENCE AND TECHNOLOGY DIVISION
1801 North Beauregard Street
Alexandria, Virginia 22311

Contract MDA 903 79 C 0320
Task Order D-32

ABSTRACT

Command and control systems are now beginning to use local area network technology. In many of the applications, classified data must be handled. This report discusses specific DoD considerations for local area networks with emphasis on security. The accessing and updating of data bases in local area networks is also discussed and directions for further research and development presented.

CONTENTS

	Abstract	iii
	List of Figures	vii
	Abbreviations	ix
I.	INTRODUCTION	1
	A. Background Material	1
	B. Overview of Study Areas	2
	C. Conclusions and Recommendations	6
II.	LOCAL AREA NETWORK TECHNOLOGY - SECURITY ARCHITECTURE CONSIDERATIONS FOR DDN	9
	A. Introduction	9
	B. Transport Media for Local Area Network	10
	C. Access Methods	18
	1. Carrier Sense Multiple Access (CSMA)	19
	2. Carrier Sense Multiple Access with Collision Detection (CSMA/CD)	20
	3. Comments on CSMA and CSMA/CD	21
	4. Token Passing	24
III.	DATA BASES IN LOCAL AREA NETWORKS	29
	A. INTRODUCTION	29
	B. Data Bases in Local Area Networks	31
	1. Local Area Network Design	32
	2. Distributed Data Bases	35
	3. Data Servers	42
	C. Personal Data Bases	50
	1. General Requirements	51
	2. Other Data Management Methods	53
	3. Data Base Integration and Security	54
	4. Security Considerations for Advanced Work Stations	56

D. Integrated Data Bases and Programming Languages 57
1. Query Language 57
2. Interfaces Through External Subroutines 58
3. Two Language Systems 59
4. Programming Languages with Integrated Data Bases 59
5. Data Base Extensions for Existing Programming Languages 59
6. Languages for Multiple Data Bases 60
7. Data Bases for Multiple Languages 60

APPENDIX A-1
REFERENCES R-1

FIGURES

1a.	Architecture for Network with Classified Data	3
1b.	Terminal Communication Security	4
2.	Broadband Cable Layout	11
3.	Baseband Cable Layout	14
4.	Typical LAN	33
5.	Single-Machine DBMS	46
6a.	Software Organization of Back-End DBMS	47
6b.	Information Flow in Back-End DBMS	48
A-1.	Early Security Architecture for DDN	A-4
A-2a.	"IPLI Everywhere" Architecture	A-6
A-2b.	Layout for 2 Host, 3 Communications Link Node in an "IPLI Everywhere" Net	A-7
A-3a.	C^2I Network--Evolutionary Design	A-9
A-3b.	Washington Area Net	A-10
A-3c.	Security Architecture with Separate Nets for C^2I and Unclassified-Secret Hosts	A-11
A-4.	MILNET Backbone	A-12
A-5.	Security Architecture with Separate Nets which Can Be Interconnected when Necessary Using "Gates" or "Switches"	A-13
A-6.	Separate Nets with Interconnecting Devices	A-14

ABBREVIATIONS

ASD	Assistant Secretary of Defense
CCA	Computer Corporation of America
C^2I	Command, Control and Intelligence
C^3I	Communications, Command, Control and Intelligence
CSMA	Carrier Sense Multiple Access
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DBMS	Data Base Management System
DCA	Defense Communications Agency
DDN	Defense Data Network
DES	Data Encryption Standard
DoD	Department of Defense
DODIIS	Department of Defense Intelligence Information System
HDLC	High-Level Data Link Control
IC	Internetwork Computer
IC	Integrated Circuit
IMP	Interface Message Processor
IP	Internetwork Protocol
IPLI	Internet Private Line Interface
LAN	Local Area Network
LED	Light Emitting Diode
NSA	National Security Agency
PLI	Private Line Interface
PL/1	Programming Language One (IBM)
RAM	Random Access Memory
ROM	Read Only Memory
TAC	Terminal Access Controller
TCP	Transmission Control Protocol
WWMCCS	Worldwide Military Command and Control System

I. INTRODUCTION

A. BACKGROUND MATERIAL

In October 1981, the Institute for Defense Analyses was requested by the Director, Information Systems, ASD(C^3I), to study local network architectures and interfaces between these networks and global networks. Security issues and interoperability were to be key considerations in these studies, which also include data base management system usage on local and local-to-global networks.

The studies reported herein relate to security architectures for local and global networks and the interfaces between these networks. Also, there were several general security architectures which were proposed for the DDN and under study during the time period when our work was done, and we were to help study this subject. We reference this work and describe parts of it in Chapter I and in the Appendix.

For local area networks, the approach taken was to study existing and developing commercial systems as well as DoD developments and to propose and evaluate security strategies for the networks while also noting some features which were desirable or undesirable for DoD usage. This work is continuing.

In order to aid C^3I in implementing cost-effective data base standardization policies for command and control systems, our work concentrated on problems which arise in local networks. The distribution of data in these networks presents particular organizational concerns which need to be analyzed, and our initial work in that area is reported herein. Some study was also given the problem of a standard language for network usage.

B. OVERVIEW OF STUDY AREAS

When security architectures for local networks are considered, there are two broad categories into which these architectures may be divided: (1) end-to-end encryption, and (2) physical security. End-to-end encryption involves encryption at the data source, as shown in Figs. 1a-1b. In this case the encrypting device is located near the host so that data is encrypted before it enters the switch, Fig. 1a. For terminal protection the encrypting device is located at the terminal access controller (Fig. 1b) if IPLIs are used. The lower section of Fig. 1b shows link encryption only (for unclassified terminals) and also where an IPLI would be placed for a more secure system. In each case, the data is encrypted before transmission of the data. As a result, if the network transmission medium (cable, twisted pair, etc.) is tapped into or if data is delivered to the wrong user through error, or if a user for whom the data is not intended obtains the transmission, the data is encrypted and cannot be read. The header--which is attached to the data before transmission--is in the clear, however, because it identifies the intended destination and it must be read by the communication system in order for the proper receiver to obtain, decrypt, and pass it to the correct user. The encryption of data while leaving the header clear is performed by a device called a Private Line Interface (PLI) or Internet Private Line Interface (IPLI). When physical security is used, the transmission medium is physically secured (generally using special conduit) and by enclosing the transmitters and receivers in a "safe" enclosure. This provides physical protection against intruders; however, the receiver must also be trusted not to read and deliver improper messages to users.

More technical information on this subject may be found in later sections; for now, the following points should be noted.

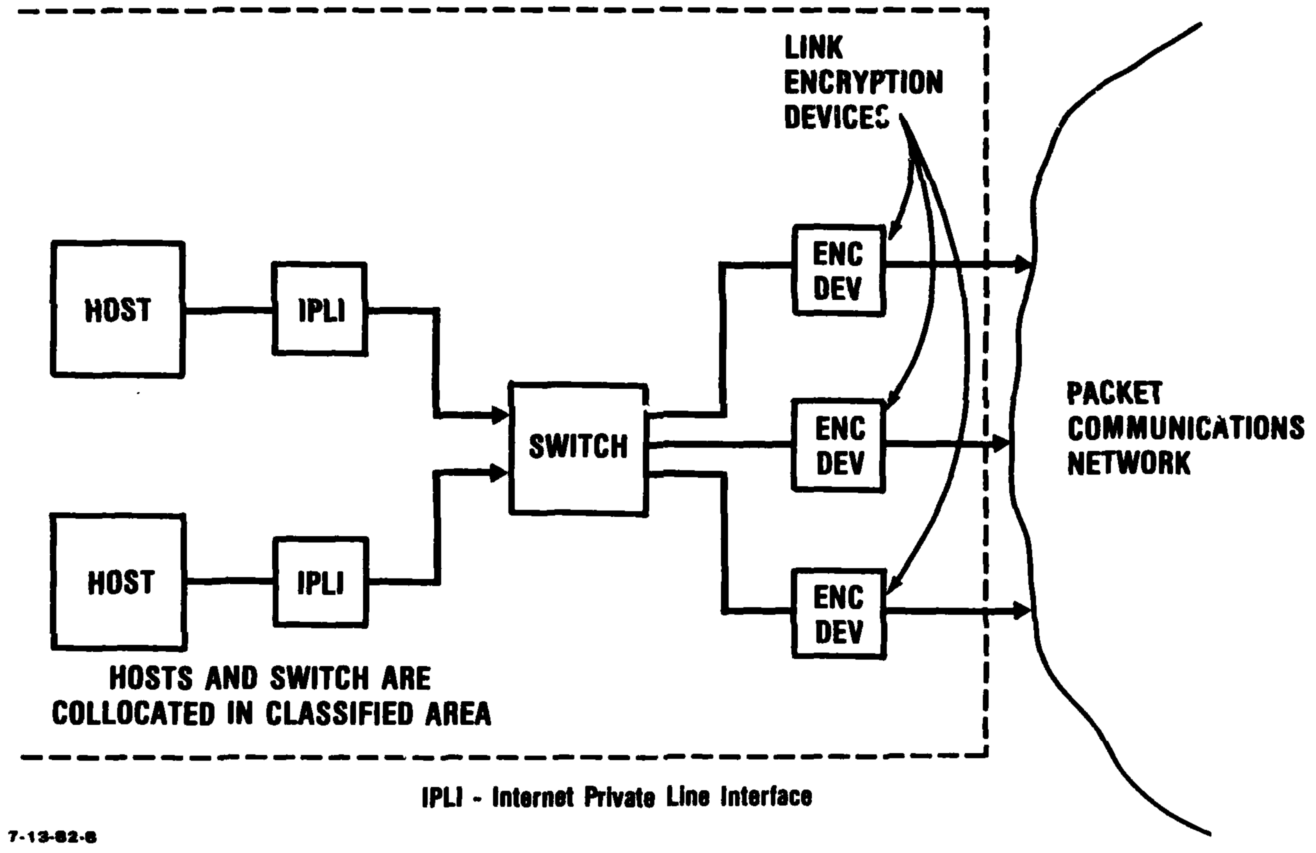


FIGURE 1a. Architecture for Network with Classified Data

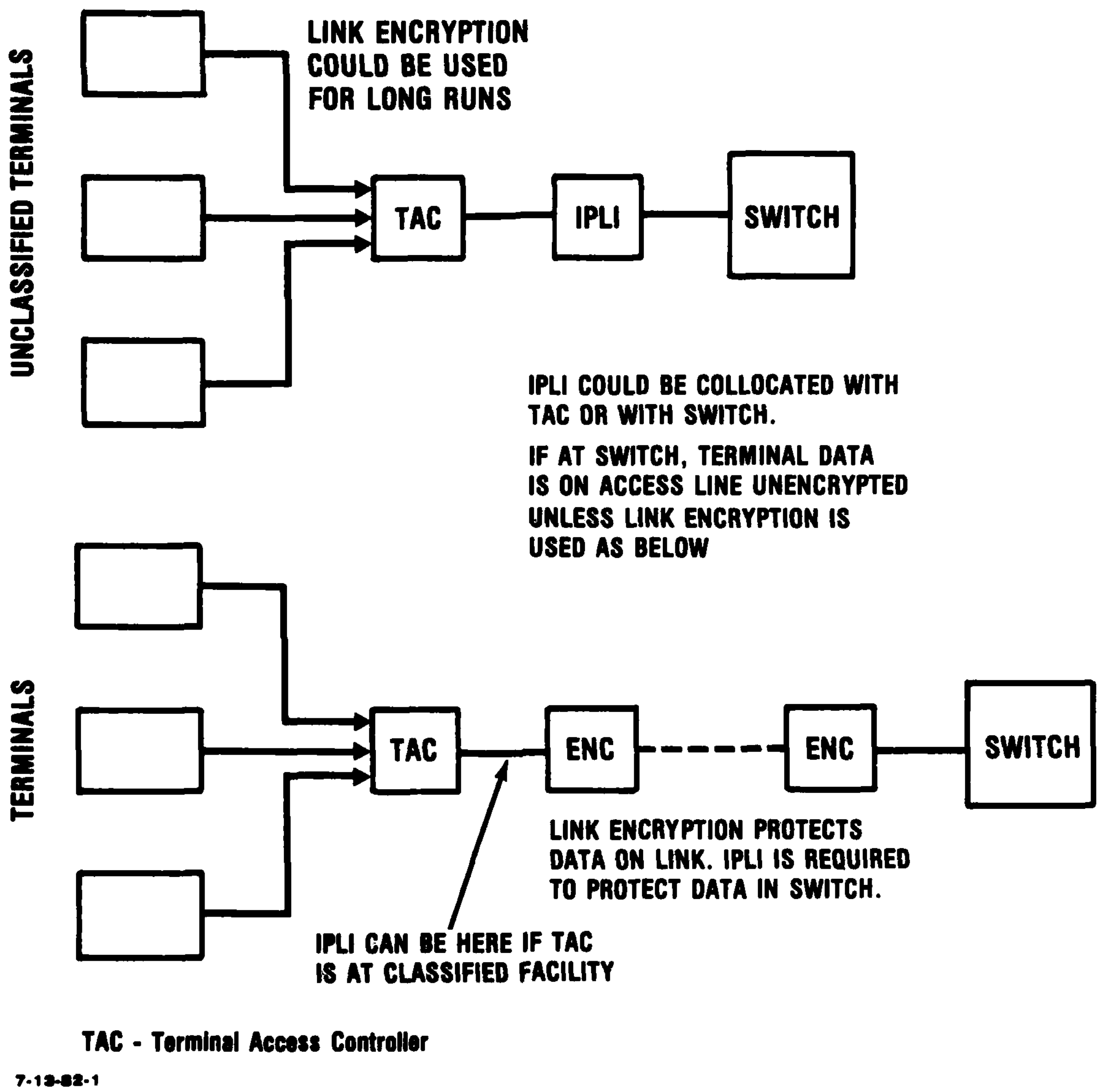

UNCLASSIFIED TERMINALS
LINK ENCRYPTION COULD BE USED FOR LONG RUNS
TAC
IPLI
SWITCH
IPLI COULD BE COLLOCATED WITH TAC OR WITH SWITCH.
IF AT SWITCH, TERMINAL DATA IS ON ACCESS LINE UNENCRYPTED UNLESS LINK ENCRYPTION IS USED AS BELOW
TERMINALS
TAC
ENC
ENC
SWITCH
LINK ENCRYPTION PROTECTS DATA ON LINK. IPLI IS REQUIRED TO PROTECT DATA IN SWITCH.
IPLI CAN BE HERE IF TAC IS AT CLASSIFIED FACILITY
TAC - Terminal Access Controller
7-13-82-1

FIGURE 1b. Terminal Communication Security

1. When end-to-end encryption is used, the data sources and destinations are partitioned into "communities of interest" by the encryption used. If all transmission and receiving stations have the same type of encryption devices, different communities use different encryption keys (Refs. 1 & 2). If these keys must be distributed and inserted manually, a given transmitter or receiver might require several encryption devices if different levels or different communities are involved (Refs. 3 & 4). This could be expensive and complicated; it is, however, the way present networks using PLIs operate. Fortunately, present networks have well-defined user communities for the most part, so there is no particular problem.

   Future security systems are now in development wherein an access controller identifies a source and proposed destination, checks the right for transmission of data at the level specified and for the community proposed, and then forwards to a key distribution device the information necessary so that the sending and destination encryption devices can be sent keys to be used during the transmission. The exact strategy for this is still being worked on, and further observations will be made in later sections.
2. When the physical security approach is used, the part of each system which puts headers on data (packets) and which reads these packets must be trusted or the data could be sent to an unauthorized destination by the transmitter placing a wrong header on the packet (perhaps duplicating the packet and sending a correct packet as well). Also, the transmitter could forward all packets, or all packets of a certain unauthorized class, to some user. Some local networks have units which monitor traffic, bypassing defective nodes and

correcting network problems of various kinds. These units must also be trusted or they might misaddress packets or bring the network down on command (whenever a special packet is read, for instance). The receivers must also be trusted, since packets with addresses to users other than from those connected to a given receiver will be received by these receivers.

3. Much, if not all, software for security architectures using end-to-end encryption must be trusted. There appears to be little difference in the amount of software which must be trusted in end-to-end and in physically secured networks (Ref. 5). For example, bringing a local network down is direct for most present commercial and DoD system local nets if software trap doors (Ref. 6) are used, unlike global networks, where switches (IMPs) are distributed and data is not broadcast or passed through all modes (Ref. 7). More detail on this aspect of security follows.

During the performance of our studies on Local Area Networks, IDA was also asked to study architectures and their security considerations for global networks. An appendix on this subject is included. Recommendations on this subject are included in the next section.

C. CONCLUSIONS AND RECOMMENDATIONS

Our studies have reached a point where some general recommendations can be made and where several conclusions concerning strategies for the present can be made. The general list follows, while supportive evidence and reasoning can be found in Chapters II and III.

1. The best security architectures for the DDN designers to pursue are those called Options 2.2 and 2.3 (see Appendix). For the present, the development of a

separate C^2I net and an unclassified network seems the best policy. Connecting links could then be implemented when the 2.2 and 2.3 options have been clearly analyzed, and the advantages and disadvantages weighed so that a final strategy can be set forth.

2. Local area networks under construction or beginning construction at this time should use physical security rather than end-to-end encryption. The technical risk in end-to-end encryption is much higher, and costs have escalated when end-to-end encryption strategy was planned.
3. Although DoD networks have emphasized broadband cable as a medium for local area networks, the emphasis has been on baseband cable for the largest makers of commercial systems. The less inexpensive interfaces are an important part of commercial considerations; however, for DoD users security considerations having to do with the complexity of securing broadband systems may be an even more important factor. Also, if audio and video services are included as part of the overall cable service, the extra complexity (from a security viewpoint) and the enlarged group of users, often at lower security levels, along with the more complex interfaces, head end, etc., may make it more pragmatic to have a second broadband (CATV) cable, and a baseband broadband (CATV) cable and a baseband data cable rather than combining the two. For these reasons and because of the consideration of wide commercial usage, baseband data systems may be more practical for DoD usage.
4. Protocols for local area networks need to be standardized as soon as practicable. (The IEEE 802 is a good start.) The need for internetting makes this essential before too much development occurs and future interoperability becomes difficult to achieve. TCP/IP are

outstanding candidates for LAN usage in C^3I and with no industry standards at present these should be seriously considered for standardization at this time.

5. Trusted software to control access to a local area network should be developed in a common user language as soon as possible. Similarly, trusted software which will correctly label and check outgoing packets must also be developed.
6. The development of advanced work stations and local area network interfaces will result in the development of a new generation of flexible data base management systems. It is likely that relational systems will dominate in this area. Future data base design and design of data base interfaces should be geared to this development.
7. Since local area networks will inevitably comprise existing hardware and data base software, communication tools such as remote procedure calls will need to be developed. An attempt should be made to standardize these.
8. The high transmission rates and low response times available on local area networks will diminish the complexity of concurrency control for distributed data bases. However, the need for replicated data (a complicating factor in concurrency control) is less acute on local area networks. The simplest methods of concurrency control should prove adequate.
9. The problem of multiple data base languages is being compounded by the development of new programming languages, especially Ada. Guidelines or recommendations should be prepared now, in order that substantially more complicated software problems will not develop within C^3I programming efforts.

II. LOCAL AREA NETWORK TECHNOLOGY--SECURITY ARCHITECTURE CONSIDERATIONS FOR DDN

A. INTRODUCTION

The following sections present some state-of-the-art information on local area networks, followed by some analysis of interrelations between local area network technology and architecture and security. Important considerations in a local area network design include (1) the choice of the transport media--baseband or broadband cable or twisted pairs, (2) the topology of the network--ring, star, and bus are the primary choices if PBX is not used, (3) access protocol--token passing, CSMA or CSMA/CD are the major choices, and (4) security architecture--physically secure or end-to-end encryption are the major choices. Within each of these choices are many smaller considerations. For example, if broadband cable is used, the frequency bands on the channel can be assigned to terminals and computers or the transceivers can be frequency agile. Some of these considerations are also presented.

The material presented here includes a considerable amount on commercial systems, in line with DoD's stated policy of using commercial products wherever possible. This appears particularly advantageous when IC chips are concerned, since chip families are being developed around the major manufacturers' systems.

B. TRANSPORT MEDIA FOR LOCAL AREA NETWORK

Deciding the transport medium for a local area network is one of the most important choices made during system design. There are two general types of media now in use--twisted pairs and coaxial cable, and there is also a future possibility of fiber optics links. The coaxial cable now used fits into two general categories--broadband and baseband cable.

This section gives some operational characteristics of transport media, including some DoD considerations concerning use of these media.

Broadband coaxial cable was the transport medium used in early DoD developments. This was because of its bandwidth and because it was readily available, as were receivers, transmitters, termination devices, taps, etc., all because broadband coaxial cable is widely used in CATV systems. The basic advantages of broadband cable are its low cost, relatively high bandwidth, extra shielding, and the fact that it is manufactured in large quantities and therefore readily accessible.

Broadband cable is suitable for long transmission distances (longer than baseband, for example), as much as 80 kilometers. Broadband supports data rates of 100 megacycles to 150 megacycles full duplex. When frequency division multiplexing is used, multiple services can be supported on different bands of the channel.

The physical topology of broadband is based on a single rooted tree because broadband is a directional transmission system. Each user device places transmission on the cable in one direction (upstream, also called the reverse channel) to the head end (refer to Fig. 2). The head end (translates and) rebroadcasts (downstream, also called the forward channel) to attached user devices. Wangnet, DCC, and Mitrenet use a dual-trunk twin cable in which separate cables are used for transmitting and receiving. Sytek and Amdax use a mid-split, two-way,

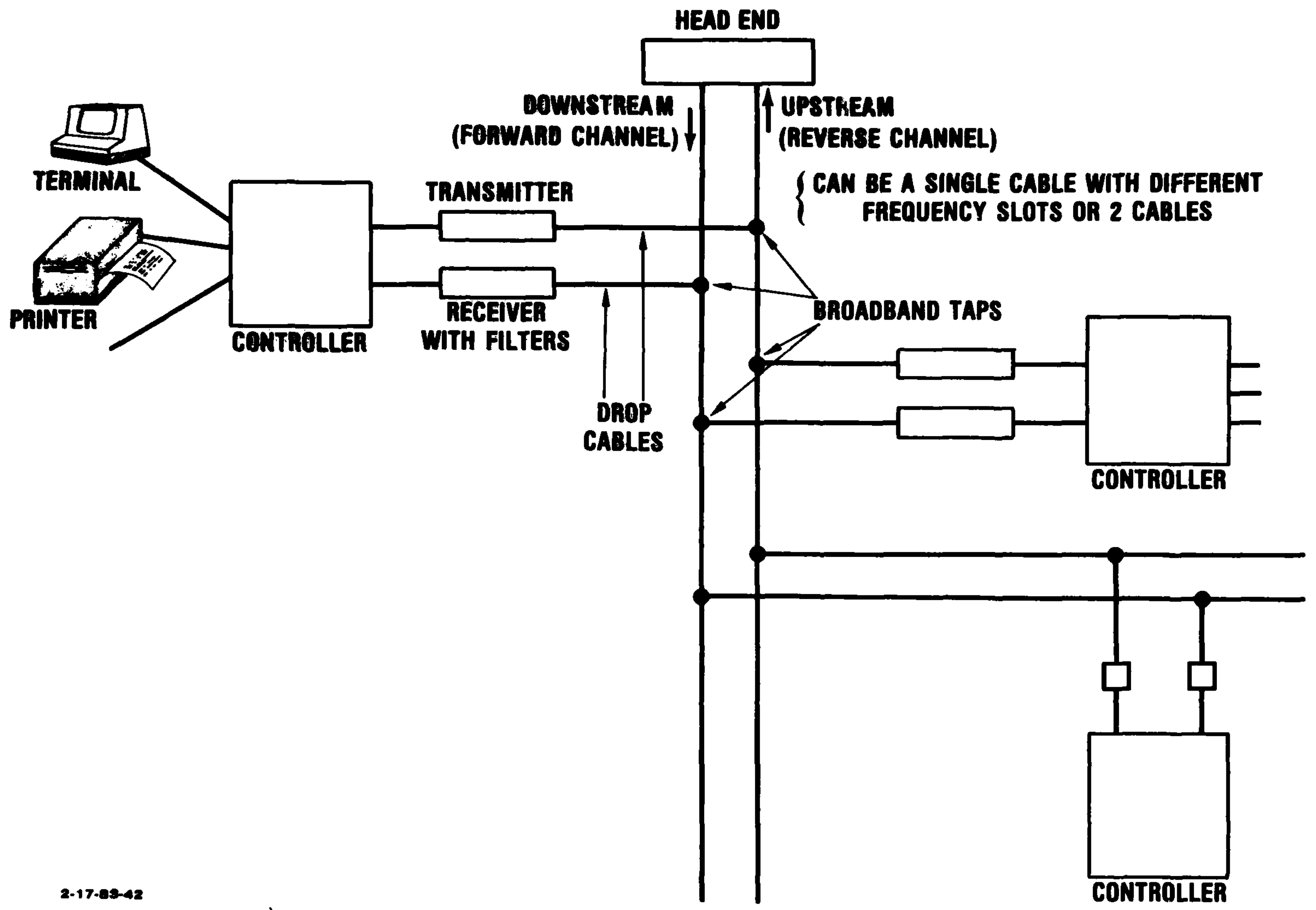


FIGURE 2. Broadband Cable Layout

single cable that is the CATV standard and also the IEEE 802 and the EIA standard.*

In these CATV-like networks, multiple channels can be used for data, video, and audio transmission.

The cable used is standard, low-loss, solid aluminum sheath coaxial 0.412-inch or 0.5-inch 75-ohm cable. The cable can be extended using connectors and by dropping branches from a trunk. Broadband systems support a 1/4-inch drop cable connected to a modem at the controller (refer to Fig. 2). Coupling to the 1/2-inch (approximately) trunk cable is made with a connector and a passive tap. The tap is a passive directional coupler that provides a mechanical interface between the trunk coaxial and the drop cable. The tap couples a predetermined fraction of the rf trunk power to or from the modem. Multiport taps provide more than one drop cable in a location. The length between a tap and a modem is normally 50 meters maximum.

Multichannel systems that use broadband transmission often divide the frequency spectrum into over 100 channels. For example, Localnet can handle up to 256 nodes on each of 120 channels. This distributes the contenders for each channel and allows the expansion by adding channels, rather than increasing contention for a single channel.

Broadband systems often attempt to integrate a wide range of communications into the Localnet media. Wangnet is an example of this, and is intended to provide voice and video services on the same cable with digital transmission. Mitrenet also provides this facility.

Baseband local area network uses 50-ohm coaxial cable, repeaters, taps, transceivers, terminators, controllers, and transceiver cable. Sections of coaxial cable are interconnected with repeaters to give greater geographical coverage (refer to

*The IEEE 802 standard, incidentally, specifies the following data rates: 1 megabit, 5 megabits, 10 megabits, and 20 megabits per second.

Fig. 3). Repeaters can be anywhere on the cable, and take the position of a tap transceiver; however, the number of repeaters in a path between two users is limited to some maximum. Making connections to a baseband cable is very straightforward. The taps used to connect transceivers to the coaxial cable should be located as closely together as possible. Each tap is spaced at a precise interval of cable, starting with the end of each cable segment. Each cable segment is tapped with a terminator. Twisted pair wire is used to connect controllers to the transceivers (refer to Fig. 3). Normally, standard RS-232 interfaces connect user devices to the controller. As IEEE 802 standard specifies, each baseband cable segment must be less than 500 meters and operate at a 1, 5, or 10 megabit rate. Innertap cable spacing should be 2.5 meters. Also, the space between the tap and transceiver should be 3 centimeters, and the transceiver cable length less than 50 meters.

Since CATV broadband cables are radio-frequency modulated, they can accommodate significantly longer distance communications than baseband systems (the signals on baseband are simply electrical signal levels representing 0's and 1's). As examples, Micronet and Sytek's Localnet use an rf signal which results in maximum transmission distances of 15 kilometers. The physical difference between baseband and broadband cable is that the baseband cable has an insulated inner conductor surrounded by a woven copper mesh, while broadband has a copper or copper-clad aluminum inner conductor surrounded by dialectric and then a sleeve of extruded aluminum. Even more significant are the cable's data transmission data capabilities. Baseband coax can support as much as approximately 50 megabits in a half-duplex mode, whole broadband cable can carry more than 100 megabits in a full-duplex mode. Broadband coaxial cable also offers higher immunity to electromagnetic and radiofrequency interference than baseband. The use of coaxial cable for broadband and baseband local area networks is based upon

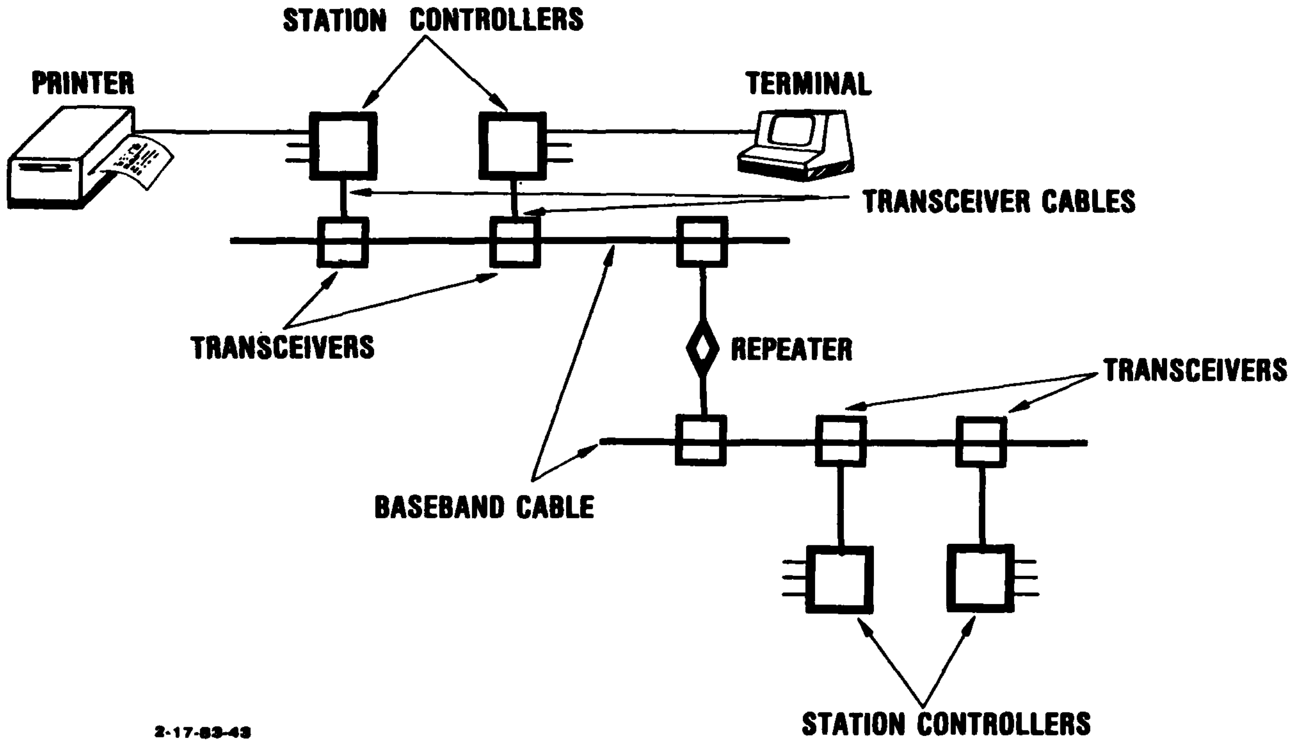


FIGURE 3. Baseband Cable Layout

the cost-effectiveness, flexibility, and performance of baseband and broadband coaxial cable, compared with the twisted pairs generally used in conjunction with PABX. High-speed, high-channel density and low error rates of coax are also advantages.

When twisted pair telephone switch technology is used, there are problems pulling additional cables through existing conduits and other tunnels in office buildings, and the proliferation of interconnected devices leads to interconnection problems in installing these systems. A single 1/2-inch diameter coaxial cable can replace 1500 or more twisted pairs. This gives cable a cost benefit in installing and maintaining a system. Also, the time and cost of sorting out and performing continuity checks on masses of twisted pairs can be a significant maintenance factor. Note, however, that if coaxial cable is broken, all terminations will be affected. However, diagnosis and repair can usually be done rapidly and simply, and can be expedited by using diversely-routed backup cables.

Fiber optic links are now being used by the telephone company in some places (long haul, not local area) but are not significant in the local area net market. Fiber optics advantages include low loss over long distances, high bandwidth, and immunity from electrical interference. However, connectors are expensive and difficult to attach, and it may be several years before fiber becomes a viable transmission medium for local area nets.

The signals used to drive fiber optics have generally been generated by LEDs, but sometimes lasers are used. As a result, long-haul fiber optic telecommunications links are helping to spawn development of semiconductor lasers capable of operating at 1.2 to 1.7 micrometer wave lengths, with trunk-line transmission rates of 140 megabits. The conventional approach of LEDs is limited to medium-range, narrow-band systems, operating at around 850 nanometers. As can be seen, true laser emitters go well beyond LED limitations. Plessey Research, Ltd. and Standard Telecommunications Laboratories in England are working on fiber optic telecommunications networks, investigating devices that will meet the requirements of fiber optics telecommunications networks, with emphasis on reliability and life. Lasers with continuous-wave room temperature thresholds as low as 19 milliamperes with stable single transverse longitudinal nodes have been fabricated by these English companies. Siemens, A.G. has developed a range of 1.3 to 1.65 micrometer high-radiance lasers that may support 140 megabits. Telecom has achieved 140 megabits over a length of 102 kilometers. Data rates conceivably could exceed 1 gigabit over greater than 100 kilometers with predicted device lifetimes of over 10^5 hours.

Despite the great promise, fiber optic links are still in the future. IC chips would need to be developed for access techniques, for example, and support technology (for maintenance) has yet to be developed.

From a DoD viewpoint, the advantages of broadband cable appear to be:

1. greater throughput,
2. multiple services (audio, video, and data),
3. widely used technology (CATV),
4. DoD has some experience with broadband, and
5. Greater shielding.

Some advantages of baseband are:

1. transmitters and receivers are simpler and less expensive,
2. the two major producers, IBM and Xerox (Ethernet), use baseband,
3. baseband IC chips are readily available, and
4. baseband is simpler and easier to maintain by digitally-trained maintenance personnel.

Twisted-pair PABX centered systems have these advantages:

1. widely used technology, and
2. readily available components.

The primary difficulty with twisted-pair systems versus cable systems lies in the many pairs which must be run in parallel to the switches versus the simple cable. Also, maintenance of cable systems should be more straightforward, and these systems are more readily expandable, more flexible, and should be lower in cost over the long run because of the lower transmission media costs.

The overall costs are also affected by security requirements in DoD. If cable is used, a system can be physically secured by placing conduit around the cable according to DoD standards (Refs. 8 & 9). The transceiver interface must also be physically secured (in a safe-like enclosure) and the software in this section must be trusted.* The cable conduit must be visible throughout the complete run, so panels must be placed on wall entries and apertures must be left open.

For end-to-end encryption-secured systems, several developments are required. An access controller (a microprocessor, probably) must check user identities and verify rights to communicate between users when a transaction is initiated. The access controller then notifies a key distribution center (in the microprocessor) that communication can begin, and the key distribution center sends keys to the users. Monitoring and record-keeping functions must also be performed by the access controller and key distribution center.

As yet, the necessary parts for the end-to-end encryption technique are not worked out. In a recent attempt to use this technique, costs escalated and the project was discontinued (this project appears to have been technically sound, but costs

*The Cl Division of NSA has developed levels of trust and descriptions for these levels, but as yet the levels for physically secured systems are not called out. Since these are in development, we assume the requirements can be accommodated.

throughout the system continued to rise until a fall-back system became the only viable way to meet deadlines and cost limitations).

As a result of the above, our conclusion is that using physical security is probably the most prudent policy for a LAN development at this time. If end-to-end encryption techniques are further developed and working models are produced, then a cost-benefit analysis might be done to weigh the advantages and disadvantages of these two approaches at a later date.

Security considerations may also prejudice DoD users toward baseband. The implications of mixing audio and video with digital are not fully understood, but this would widen the user community to include users and devices which may increase the overall risk. Being all digital, baseband is simpler to analyze, and the simplicity of baseband interfaces may be a substantial advantage when security is considered. There are, however, systems in development which use bandwidth allocation on broadband cable to partition users into communities. The use of band-pass filters to search out data needs to be analyzed as both sideband transmission levels at transmitters and attenuation levels at receivers must be carefully considered, as must strategies for moving filter frequencies.

C. ACCESS METHODS

The choice of a transport medium influences the choice of access method used for the network. There are three general categories of access techniques now in wide use, with variations of all three existing in commercial systems. The three access techniques are (1) Collision Sense Multiple Access (CSMA), (2) Collision Sense Multiple Access with Collision Detection (CSMA/CD), and (3) Token Passing.

This section discusses these access techniques, with comments on their possible use in DoD systems.

1. Carrier Sense Multiple Access (CSMA)

As has been mentioned, broadband cable networks use CATV-like cable layouts in which there is a head end containing frequency translation and repeater circuits. In tree-like fashion, the distribution of drop cables emanates from this head end and provides taps for attaching information processing equipment to the network. When a user broadcasts a packet of data, the packet travels inbound ("upstream") to the head end, and is then sent outbound ("downstream") to the waiting receivers. This two-way path is accomplished either by having two physically separate cables, or one cable in which the frequency is split, with low frequencies going "upstream" and high frequencies going "downstream," as has been mentioned.

If the network uses CSMA, when a terminal has a data packet ready for transmission it senses if a packet is being sent,* and if one is, it will defer until the current packet is completed and then start to transmit. This is the carrier sense phase of CSMA. Sometimes two nodes will begin to transmit virtually simultaneously, resulting in a collision. The collision occurs because each node transmits within the propagation time interval between the nodes, and hence each could not hear the others' transmission before trying to transmit. These collisions are resolved independently by each node. Each node calculates a random interval that it will wait before attempting to retransmit the packet. This random retransmission routine prevents two nodes from engaging in an unending series of collisions. When a pure CSMA system is used, only after the complete packet has been sent is the collision detected, generally by the lack of an acknowledgment from the receiver node.

*If the cable is divided into frequency bands, then the transmitter is using a particular frequency band and only receivers also using that band can receive the transmission.

2. Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

In order to cut down on the time lost due to collisions, a second technique, Carrier Sense Multiple Access with Collision Detection (CSMA/CD), is often used. In this case the transmitter monitors the channel as it sends, and if a collision is detected, each colliding node will discover the collision virtually as it occurs. Each node can then abort the remainder of the transmission, potentially saving time on the channel.

When CSMA/CD is used, stations needing to transmit must wait for current packets to traverse the net. Entire packets are sent once communication is initiated unless a collision is detected. All stations on the net must be capable of detecting the presence of another station's messages on the network cable, and this is called carrier sensing. In most implementations, a channel interface is ready to transmit once it has determined that no other carrier is present. A preamble of bits, just long enough to ensure that the first bit has had time to travel to the furthest point of the network, is sent. This is followed by source and destination addresses, a control field, data, and a cyclic redundancy check. When the transmission stops, a specified interframe time is observed, following which another interface is free to begin transmitting. The time required for a signal to make a round trip over the longest network path is called the slot time. When two or more stations begin sending their preamble at about the same time, a collision occurs and the bits become garbled. Each transmitting interface then recognizes that its message has collided with another one, and transmits a jam signal. At the same time, each station's controller executes a random number algorithm and calculates a random number of slot times for the station to defer before attempting another transmission. If a second collision follows the first, the random number algorithm produces a longer set of slot-time deferrals until a message is sent without collision.

A major attraction of CSMA/CD is that it permits network access control without the need for a central network controller.

Collision detection is difficult to implement on broadband systems, and often unreliable. As a result, baseband systems more often use CSMA/CD. The reason broadband systems have difficulty is because the receiver's filters affect the signal capture so that a strong signal is detected by a receiver even when a weaker signal also is present, and this can delude some users into thinking that only one data packet signal is being transmitted. Other intended recipients, however, may not lock onto the stronger signals when they sense a collision is occurring.

Present broadband implementations of CSMA/CD do collision detection by reading their own packet from the cable and comparing it with what was sent. If the comparison is successful, the transmitter concludes that no collision has occurred. However, when a collision does occur the packet may not be aborted until it is completed. The result is nearly the same performance as pure CSMA without any real benefits of collision detection.

3. Comments on CSMA and CSMA/CD

An important factor in CSMA throughput is the ratio of propagation delay to packet duration. If delays are short, the ratio will be less than one, so that the period of vulnerability of a packet with collision is short, resulting in high-channel throughput, where throughput is measured as the portion of channel time engaged in successful transmission. As the ratio approaches 1, the channel performance degrades, and the probable collision interval becomes as long as the packet itself. If the channel is pushed to its limit, nodes become engaged in retransmission of previously collided packets blocking the entrance of new packets onto the channel.

When a user located at the midpoint of the cable in a broadband system sends a packet using CSMA, that packet is vulnerable to collision for a time equal to 1.5 times the one-way maximum delay, which is the time it takes to reach the head end plus the time to travel downstream to the farthest subscriber. The worst-case subscriber is vulnerable to twice the one-way delay. A baseband layout uses two-way propagation on the cable so that worst-case delay is simply the end-to-end delay. For both CSMA and CSMA/CD, performance improves for longer packets and shorter propagation delays.

Major broadband cable local area network manufacturers are Wang, Sytec, and Digital Development Corporation. In some systems, time division multiplexing is superimposed on the elementary frequency division multiplexing of broadband transmission, allowing each channel to be shared by several stations. Stations sharing a single channel are then prevented from colliding with one another by transmitting only during time slots defined by a remodulator. Also, all packets in a given network will then have the same number of bits, and thus occupy the same time on the channel. The head end continually transmits packets on the downstream channel, some of which are retransmissions of packets originating at stations on the network. Addresses of specific stations are broadcast to all stations that share a particular channel. The remainder are empty packets whose sole function is to provide an interval time. When a station has a message to transmit it waits for an empty packet on the downstream channel, indicating an available time slot, and begins transmitting on the upstream channel. When traffic is light, the network can omit time slots and allow a contention between the stations. During contention periods a station recognizes a collision when its packet fails to return on the downstream channel in a reasonable time, or if the packet comes back garbled. In each case the station retransmits the packet. Also, as the data rate increases, the percentage of that rate that is usable decreases.

Computer Automation, Inc., makes a broadband system which uses a special form of CSMA/CD, which claims data rates of up to 10 megabits and error rates as low as 10^{-9}. Individual cables can be up to a kilometer in length, and the system can support up to 254 addressable nodes. CSMA networks can be either rings or open busses. All transmissions are broadcast in the sense that every station can receive them. User receivers will hear everything but only forward the transmitted data addressed to them. In one form of CSMA the receiver of a message is required to acknowledge it. This requires the sender to retain a copy of the message until the acknowledgment arrives, in case the message must be retransmitted. If two messages collide, the receiver does not recognize its address and does not accept the garbled message, and therefore does not acknowledge it. As a result the transmitter cannot distinguish between messages not received at all, messages received with errors not caused by collisions, or between messages lost in collisions, or even messages not received because the receiver's power is off. This form of CSMA also does not allow for loss of acknowledgment of a correctly received message. As a result, most networks such as Ethernet, Net/One and Z-net, use CSMA/CD. A possible disadvantage for CSMA/CD is that every transmitted frame must be at least twice as long as the propagation time to the most distant station in order to guarantee that a collision is detected before transmission is complete. For example, Ethernet is limited to 1 kilometer, so the one-way propagation time is approximately 6.6 microseconds, and frames must last at least 13.2 microseconds. Therefore, they must contain a minimum of 132 bits at 10 megabits. If a station at one end of the network begins to transmit, the station at the other end will not receive the transmission until 6.6 microseconds later. During this 6.6 microsecond period, if the station at the other end begins, it will transmit almost 132 bits before it discovers that someone is transmitting from the other end.

Probably the more important disadvantage for CSMA/CD is the inability to provide a maximum access time. Under light conditions a user can expect to get onto the network promptly; however, the chances of encountering collision after collision as a user tries to transmit can become large if the system is very busy. As a result, no guarantee of a maximum access time can be given.

4. Token Passing

In a token passing access system a particular bit configuration called a token is transmitted on the network between data packets. Examples of token networks are Tandy's TRS-80, the Datapoint ARCnet, and IBM's LAN. In these networks, no station can transmit unless it has just received the token. A station addresses the token to another station in the network following its completed reception. The token can be passed immediately after receipt if the station has nothing to send, or after sending a packet. Priority schemes can enable selected stations to keep tokens for some stipulated period, but not indefinitely. This makes it possible to guarantee a maximum time that a sender must wait before being given access to the network.

There are many variations of token passing. The network can be an open-ended bus or a closed ring. Token busses are broadcast systems where token senders place a specific address (that of the next station in the token passing sequence) on the token when transmitting it. Every station receives this transmission but only the addressed station gets the token. The token bus is physically ring-like in operation because of the fixed order of the token passing. A user that receives the token can transmit a packet of data addressed to any user in the network immediately. The station follows the packet with a token addressed to the next station in sequence. This approach works with either a bus or a ring. If the network is a ring network, each token contains a bit that identifies it as free or busy. When a user with a packet to transmit receives a free token, the user changes it to a busy token and then forwards it, followed by its packet. If the station has no data, it forwards a free token. A busy token makes one round trip and returns to the originating station. There are several strategies in use at this point. The user can wait for the entire packet to return, assuming that the user to which the packet was addressed read it, and after the entire packet is returned to the sender, if it is o.k., the token is changed back to free status and forwarded. A second choice is to change the returning token to free status and forward it at once (part of the just-transmitted packet may still be circulating). A third choice is to generate a new free token immediately after transmitting the packet. This allows two or more tokens to be in the network all at once. For these strategies the token must take longer to travel around the ring than to write the packet on the network. For a token bus system, a user receiving the token acknowledges reception to the sender. If the sender receives no acknowledgment, it considers the station to be inoperative, and retransmits the token to the user who was previously second in line. The propagation time around a ring is the sum of the delay in the communications medium, plus one bit time per station introduced by the accept and retransmit step. Some means of bypassing a failed station or link to avoid bringing down the entire network is needed, and this presents a problem to DoD users. Token passing has the advantage that no station can transmit without a token, so messages cannot collide, and a message can be as short as desired.

IBM uses a baseband, token-passing form of access, and has a monitor which is a station on the ring. The monitor watches for lost tokens, defective tokens, multiple tokens, and other pathological conditions, and takes action to correct them. (This monitoring activity can be centralized or distributed through the network.) IBM also has a technique for restructuring a ring in case of failure. IBM's system has two or more concentrators to which the user connections are attached. The concentrators contain active subsystems that, by examining transmissions, can locate a failure point and set up a bypass around failed stations.

A typical LAN early design (DoD) was for 2000 terminals and 150 printers. Broadband cable was chosen over twisted pair because of installation and maintenance expense for twisted pairs. Standard 0.5 inch aluminum outer conductor, copper inner conductor, coaxial cable was chosen and a two-cable system (one upstream--one downstream) was selected.

Terminal interfaces used a bus interface unit and there was also a computer interface unit. The cable was used multichannel with each channel having a 1.544-megabit data rate. Standard 40-channel CATV components were used and the bus interface units had frequency agile front ends made from modified TV tuners.

There was a log-on channel, and a system control computer polled key interface units continuously. When a user signs on, the system control computer verifies the user's identity and his rights to the computer connection address called for. The system control computer switches the rf tuner section of the key interface unit to the channel used by the computer. (Computers have fixed channel allocations, and their interface units are not frequency agile.)

The system controller also arranges for key distribution. The key interface units have log-on (and off) procedures in ROM but the communications protocols are down-line loaded in RAM.

The CSMA protocol is used to access the key channel to and from the computers and key interface units. X.25 and HDLC were used for levels 3 and 2 of the International Standards Organization model for open system interconnection.

In all, this is a carefully thought-out design for DoD-type use. The use of end-to-end encryption seemed reasonable, although considerable development was needed to implement this feature. The cost of a physically secured cable was about $3 million more than for a conventional cable installation. This was felt to be a further justification for end-to-end encryption. In retrospect, however, the escalation in cost for encryption units in the key interface units and the cost of developing and implementing the key distribution software and hardware appear to have exceeded the cost of physical security. Also, the cost and difficulty of implementing verification, and whether trusted key interface units would be necessary, and at what expense, have not been determined.

This system is relatively complicated for a local area network. A major consideration here would be whether a baseband physically secured system would not involve less technical risk than the broadband physically secure system. At the time the design was made, however, IBM was not in the baseband arena and Xerox was not in its present strong position. As a result, using DoD-type technology would certainly have appeared the most prudent approach. Now, however, with the proliferation of IC chips implementing CSMA/CD, baseband transceivers, HDLC, and even X.25, a formidable array of components can be assembled. By adding verification software, a designer would be well on the way to a complete design and the risk appears low, while interface unit costs should be very reasonable.

III. DATA BASES IN LOCAL AREA NETWORKS

A. INTRODUCTION

Probably the most significant advance in commercially available computer hardware over the past two years has been the development of "personal work stations". The term is very vague, for it can encompass anything from a limited word processor to a machine with the processing power of a large minicomputer or small mainframe. Based on these developments, a number of proposals within C^3I and related organizations have been made that will distribute existing computational loads over a network of personal machines and special-purpose processors, file servers, long-haul network interfaces, etc.

One of the main functions of such a network is to provide users with a variety of data base interfaces. There are two obvious extremes:

a. The data server performs all the processing, and the user work station acts as a regular terminal. In this case, the network is being used simply to support terminal-to-host communication. Other than that, the software to support data base access is exactly the same as is required for a regular configuration of terminals and host.

b. At the other end of the spectrum, the user work station may have the storage and processing ability to maintain a substantial data base, and the data server's function may be only to provide backing store and to support a few shared files that may be independently updated from various sites.

Somewhere between these two extremes lies the practical approach. However, to discover what this approach is, the probable characteristics of the personal work stations involved must be examined. Typically, these computers are considerably more powerful than conventional microcomputers in three important ways:

1. The address space, or potential memory size, is substantially higher than the conventional 64k available on the earlier microcomputers.
2. Secondary storage is a fixed "hard" disk (Winchester). Capacities of 30 mb are common. Compare this with the usual maximum of 1 mb for a dual floppy disk system.
3. The screen interface is often more sophisticated than that on a conventional terminal. Bit mapped screens and graphics software may be provided.

All three of these will have a dramatic impact on the way data bases may be used. The first two of these make possible the implementation of a substantial data base management system on a user work station (the performance of a floppy-disk-based micro is inadequate for any substantial data base work). The last point will make possible new and better user interfaces, especially for large amounts of textual data or for certain other "nonstandard" data, such as pictorial or geographical displays.

The price of such work stations should justify their use within C^3I networks. Machines such as those described above are now available commercially in the $8k-30k price range, approximately the same (in constant dollars) as a terminal ten years ago. Examples include the Apollo, Fortune, Sun "terminal", Apple's Lisa, Wicat, and some of the larger IBM and HP work stations. Most of these systems are based upon the 68000 processor, and many have integrated software to support some kind of high-bandwidth networking. In a somewhat higher price range, but worth noting in this context, are the Lisp machines developed by

Symbolics and Xerox. These have been designed in the hope that "expert" systems can be designed for personal machines, but it is quite possible that such systems will be desirable in C^3I networks.

The impact of networks and advanced work stations on the use of C^3I data bases will be divided and analyzed in three main areas:

A. Data bases on local area networks.

B. Requirements for a "personal" data base management system.

C. Integration tools for existing data bases.

These topics require some initial comments. The first point deals with the problems of distributing (or not distributing) data on a high-speed network of processors. These are somewhat different from the problems of data base access on a low-speed geographically dispersed network. The second point deals with the data base tools that a user requires on a personal machine. Any user who downloads a file to such a machine has of necessity created a local data base, and since this will inevitably happen, it is better that it happen in a controlled fashion. The last point deals with that in integrating, in a local data base, existing data bases with similar, but not identical, structure. Finally, in continuation of preceding studies we include a survey of developments in data base/programming language integration.

B. DATA BASES IN LOCAL AREA NETWORKS

Previous sections have described Local Area Networks (LANs) of processors geographically separated by distances of the order of a several hundred meters and having high communication rates. Local area networks are distinguished from the long-distance networks, such as the ARPANET, by the type of communication technology that is exploited. The distinction between a local area network and a conventional computer configuration using a bus as the medium of communication is less clear, and is more a matter of the philosophy surrounding the kinds of message passing that are used. A "single wire bus" configuration is often used for the implementation of a local area network.

A typical design of a local area network is shown in Fig. 4. It contains a number of work stations, a high-speed (and perhaps high-quality) output device, a file/data base server, and a connection with some other network. The bus-like topology of this diagram is one of the possible topologies for such a network that has been described. Such configurations are being widely considered for military applications; in particular, DCA has similar designs for site configurations when WWMCCS is upgraded.

A feature of nearly all such designs is that some form of file or data base server is included. This section of this report will concentrate on the design and implementation of data bases on such a network. At first sight, the only critical feature of local area networks to affect this issue is the communication rate; however, the network topology and the communication protocols do play some part in the physical limitations on data base design. It is therefore worth reviewing them here.

1. Local Area Network Design

For data base design considerations, the most important distinguishing feature of a local area network is the transmission rate. Characteristic transmission rates are in the range of 1-100 megabits/second in many designs, which exceeds the rate at which a data base management system can deliver or absorb data in most cases. One of the arguments for a high-level network query language (Ref. 10), that of reducing network traffic, no longer holds. The other argument, that of having a uniform interface, is still relevant.

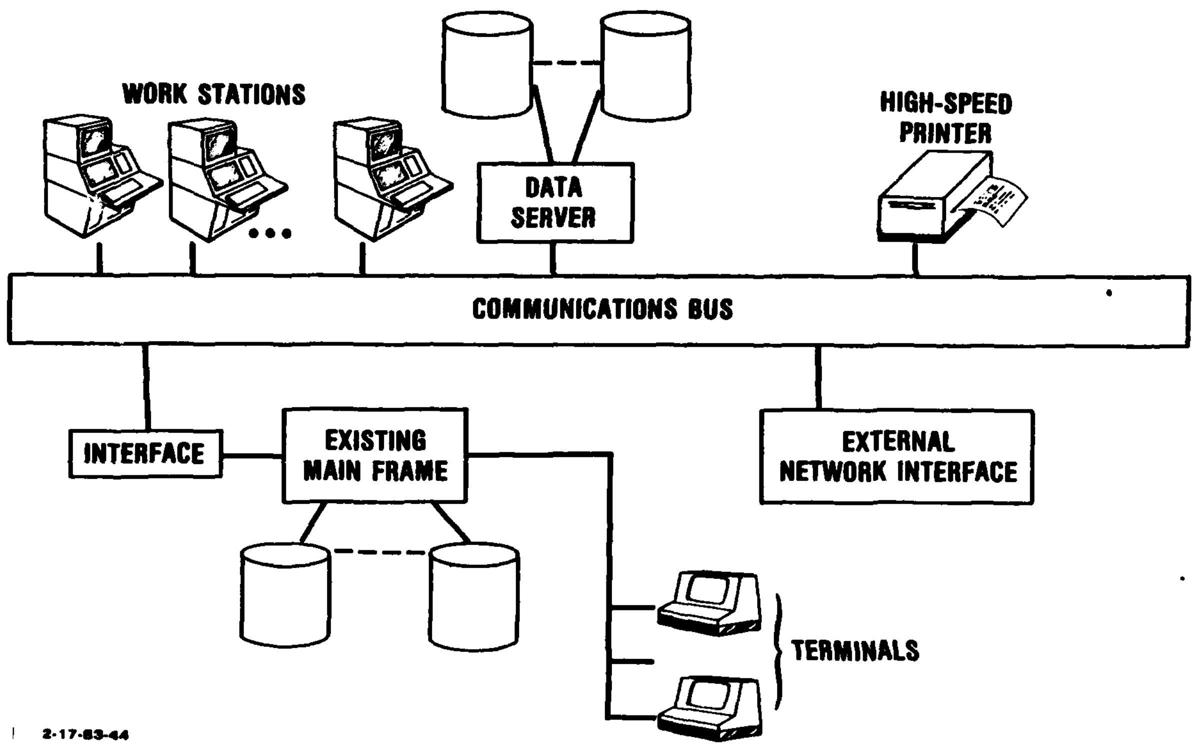


FIGURE 4. Typical LAN

The details of how this transmission rate is achieved--network topologies, network protocols, etc.--are less relevant to the discussion of data base design, but they are not independent issues. Factors such as reliability play an important part, both in network design and in the provision of shared data bases. In this section we shall, therefore, briefly review some of the details of Local Area Network design.

There are four possible topologies for a LAN: star, ring, linear bus, and network. Network topologies, while having obvious advantages of reliability for long-haul networks, are not normally considered for local area networks. The linear bus architecture is simply an extension of the bus architecture frequently used in designing single computer configurations, i.e., to connect a processor to its peripherals.

As mentioned, two similar transmission protocols are ad hoc standards for bus networks. These are the Xerox Ethernet protocol (Ref. 11) and the IEEE-802 standard (Ref. 12). In these protocols a host transmits a message (whose length has a predetermined maximum) by first listening to see if the network is in use. If not, it transmits; if it is in use, it waits a "random" time and tries again. This allows the possibility that two or more messages could collide. The collision may be detected at a low level (within the analog circuitry) and if it occurs, the contending hosts each wait for a "random" period and try again. The overriding advantage of this method is that the operation of the network is not dependent on the reliability of any complex processing at some node, and individual stations may simply be "unplugged" without affecting the reliability of the network.

In ring networks, a set of message "slots" is kept in constant circulation around the ring. Each node on the ring must forward the messages that are not intended for it and must move messages that are addressed to it. The question of such a network is clearly dependent on the continual operation of each node. In error states, or when the network is reconfigured, a distinguished node is used to restore the protocols. For further discussions of the network protocols, see Ref. 13; see Ref. 14 for surveys.

Finally, another form of star topology is used in IBM's System Network Architecture. In this, a central node is responsible for the interconnection of a "subarea"--a local area network. All messages between nodes (whether or not they are both in the same subarea) flow through such a central node, which is also responsible for initiating communication.

In the last two architectures, there is one node whose reliability is essential to the reliability of the network. Such a node is an obvious candidate for any kind of shared storage and in many networks (HP, for example), the central node is also a data or file server for the rest of the network.

2. Distributed Data Bases

A number of problems associated with distributed data bases have been extensively studied, and we shall review them here in order to investigate which are relevant to the problem of data bases on local area networks. The term "distributed," when applied to data bases, has varied meanings.

Large data bases have traditionally been "distributed" over several secondary storage devices, and all the logical problems that occur when a data base is more highly distributed (e.g., at several nodes of a computer network) also occur with a data base that is controlled by one processor. There are, however, some physical details that need to be taken care of. On a long-haul network, the communication rates are very much slower than those normally associated with a disk. It is therefore appropriate to try to cut down the amount of network traffic associated with a given data base query by performing more work at each processor. For example, the processors themselves may be capable of interpreting relatively high-level languages, rather than doing low-level i/o. This means that the study of query (and update) languages (Ref. 10) and query optimizations is important.

Since on a local area network transmission rates are comparable with those on a disk, high-level languages and query optimization are not required for distributed data bases (though they may well be required for other reasons--e.g., heterogeneous data bases). Therefore, a data base distributed on a local area network presents few problems, and it is often a relatively easy task to take an existing DBMS and make it "distributable" over a local area network. We shall discuss how this can be done in the next section.

Much work on distributed data bases has been directed toward the problem of <u>redundant</u> data. In a DoD environment it is generally advisable (Ref. 15) to have several copies of the same data geographically dispersed over a network. In this way, data access will not be dependent on one processor, and the reliability of many data base applications should be guaranteed even if parts of the network are destroyed. The efficiency of data base access may also be improved by having multiple copies, and also be improved by having "local" copies of certain frequently used data.

The major difficulty with redundant data is keeping a consistent version of all copies in the presence of concurrent transactions. As a very simple example of this problem, suppose that two update transactions U_1 and U_2 both update the "same" piece of data x. However, two copies of x, x^1 and x^2 exist at different locations on the network. The two updates may be issued from different sites in the network and, because of message transmission time, they may arrive in the order U_1, U_2 at the site that holds x^1 and in the order U_2, U_1 at the site that holds x^2. The two copies of x will now be inconsistent with one another.

This problem has been studied extensively and we shall briefly review the problems of concurrency in distributed systems here.

When two transactions are run against a data base there is a danger that they will interfere with one another and leave the data base in a state which is in some sense incorrect. The purpose of concurrency control is to ensure that this kind of interference does not take place. Before summarizing concurrency control algorithms, it should be noted that the semantics of data base are of paramount importance. For example, a common case in which two transactions can interfere with one another is in concurrent updating of an account file. Consider two programs A and B that both attempt to change a BALANCE in a data base.

Program A	Program B
A1 Read BALANCE from data base	B1 Read BALANCE from data base
A2 Add $500 to BALANCE	B2 Deduct $1000 from BALANCE
A3 Store BALANCE in data base	B3 Store BALANCE in data base

If program A is performed entirely before B or entirely after it (i.e., the transactions are in the order A1, A2, A3, B1, B2, B3 or in the order B1, B2, B3, A1, A2, A3) then the result will be to decrement (correctly) the BALANCE by $500. Suppose, however, the programs are run concurrently and the order in which the transactions take place is A1, B1, A2, B2, A3, B3. In this case, program A will have no effect and the BALANCE will be (incorrectly) decremented by $1000.

Compare this with a similar scenario in which two intelligence analysts might simultaneously attempt to re-estimate the position of some moving object.

<u>Analyst A</u>	<u>Analyst B</u>
A1 Read POSITION from data base	B1 Read POSITION from data base
A2 Add (x,y) to POSITION	B2 Add (x',y') to POSITION
A3 Store POSITION in data base	B3 Store POSITION in data base

In this case Analyst A and Analyst B are both attempting to correct the data base in a similar fashion, unaware of each other's actions. In this case it might be appropriate to add either (x,y) or (x',y') to the data base but not both. The point to be made is that concurrency analysis can only proceed from careful semantic analyses of the updates, and should new updating transactions be introduced into a system, the analysis will have to be reworked. This may prove especially difficult in the relatively flexible data base systems that we shall consider later.

Most commercially available large-scale data base management systems provide some form of <u>locking mechanism</u> that allows one process to read-lock or write-lock a record or other data base component. If, for example, a process obtains a read-lock on a record, no other process may read that record until the read-lock is released. Locking mechanisms are needed for implementing concurrency control, but the problems of implementation arise above that level. Only in a few cases (we note these later) do any commercially available systems provide any higher level of concurrency control. This is usually left to the implementers of a particular set of applications programs, and is usually the most problematical part of the program specification.

The basic method of providing a higher level of concurrency control is <u>two-phase locking</u>. A process that updates the data base first obtains the appropriate (read or write) locks on all the relevant record. It then performs the update and releases those locks. The problem with this is that <u>deadlock</u> between two processes can easily result.

Process A	Process B
A1. Read(x)	B1. Read(y)
A2. Write(y)	B2. Write(x)

Suppose that A1 and B1, respectively, require (read) locks on x and y. If these are obtained before the execution of B1 and B2, then neither B1 nor B2 will be able to proceed. The solution to this problem is for one process to back out entirely and wait for the other to complete. A more general version of this method is to subdivide each process (this is not always possible) into a _growing_ and _shrinking_ phase. During the growing phase, the process only acquires new locks, while during the shrinking phase it can only release locks. A process that completes its growing phase may progress to completion, while one that is halted in its growing phase must back out of all its locks and restart later.

The next problem is to detect deadlock. This can be done by a graphical analysis. A directed graph is constructed whose nodes are the processes and an edge is drawn between node A and node B if A is waiting for a record that B has currently locked. If this graph contains a cycle, there is a deadlock which can only be broken by one of the processes backing out. There are various methods for deciding which process should back out, and there are also methods that, by prior analysis of the transactions, can prevent a transaction ever starting if it is likely to cause a deadlock. See Refs. 16, 17 & 18 for details.

Another possibility in a centralized data base (or a distributed data base with centralized control) is to use a method similar to that employed by Ethernet. Any process, if it cannot obtain all its locks, must back out of all locks, wait a random time, and try again. This solution is not feasible for a long-haul network when transmission delays can be substantial, but for classes of simple transactions, it is perfectly feasible for local area networks.

The major problem in concurrency is the preservation of data consistency. The simplest case of this is when data are replicated at different sites. It is often deemed useful, either for reliability (insurance against failure of one or more nodes), or for efficiency (when local access is much faster than remote access) to have multiple copies of certain data. The maintenance of replicated data so that the users have a consistent view of the data is the major thrust of the research behind SDD-1, currently being developed by Computer Corporation of America. The details of this are reported in Ref. 15.

Replicated data is only one form of consistency that is important in concurrent access. More generally, data may not be replicated, but there may be integrity constraints that exist among data bases. To our knowledge, little work has been done in concurrency control for this more general form of data consistency.

There are other forms of concurrency problems with data consistency that exist in C^3I data bases. These are concerned with data consistency in "centralized" systems. For example:

(a) There is a common practice of keeping "stripped" files which replicate information in a larger data base. This is done for two reasons: one is for efficiency of processing a large number of similar queries against the data base. The other reason is that often there are better tools for querying a simple rectangular file than there are for a more complex data base. The concurrency problem is usually solved (a) by creating the stripped file from the data base relatively infrequently and in such a way that ensures that any transaction runs against exactly one version of the stripped file, and (b) by only allowing queries (as opposed to updates) against the data base.

(b) A single data base will often contain data structures that have to be kept consistent. A relatively common example is with data bases that hold (among other things) geographical information. The coordinates for a record are usually stored within that record. At the same time (for efficient geographical search), it is often necessary to erect some indexing structure, or other data structure) on top of these records. In a Codasyl data base, certain geographical clusters will be combined into a set. The set structure must be kept consistent with the coordinates, and geographical search algorithms could be seriously endangered if no attempt is made to preserve this kind of consistency.

A good survey of concurrency algorithms to preserve data consistency in distributed data bases is to be found in Ref. 19. Many of the more complicated algorithms appear to be intended to long-haul networks in which there may be substantial delays in message passing between sites and where transmission rates may be inadequate.

It is not clear that the more complicated methods will have immediate usage for local area networks within C^3I. Reasons are (a) the concerns for reliability and robustness may be less important for local area networks, (b) the flexibility of data base access may defy the kinds of pre-analysis required to define the locking algorithms described above, and (c) where some sort of locking is required, it may be possible to implement an extremely crude time-stamping algorithm that will take care of conflicts such as those described above. It should also be noted that the transmission rates on local area networks are so high that it is possible to copy a large fraction of a data base in a relatively short time.

Nevertheless, for long-haul networks and for local area networks on large military platforms (e.g., aircraft carriers) where the data base applications are relatively predictable, we believe that the work on concurrency in distributed redundant data will prove extremely important.

Other than the research projects described above, we know of few commercial products that incorporate any sophisticated degree of concurrency control or manage to implement adequate tools for distributed data bases. Exceptions are a distributed Codasyl for Apollo computers (Ref. 20) and the multiple copy data bases used by Tandem computers.

3. Data Servers

Nearly every design for a local area network contains some node whose function is the maintenance of large quantities of data. In many cases this node is viewed as dedicated to the maintenance of a (physically) centralized data base, in which case it is often called a data server (Ref. 21) or a "back-end" data base system. The degree to which such systems are specialized can vary. Such a node might serve in one of the following ways:

a. A File Server. This is by far the most common of data servers and exists on a number of implemented networks. To a user program running at a work station, the file server provides the same functionality as the file i/o system of single machine operating systems. Moreover, since the transmission rates are so high, there is little or no difference in the performance of a file server when compared with conventional operating systems. The reasons for having a dedicated file server are twofold:

(1) For reasons of economy. It is presently cheaper to provide a central (large) mass-storage device rather than local mass storage for each work station. However, the economics may change in the near future. We have seen that work stations (or "personal machines") with a built-in Winchester disk system, capable of

providing fast access to tens of megabytes of storage, are now appearing in a price range that is competitive with existing word processors and home computers.

(2) For reasons of network design. The file server may also be the central node on a star network and may also be responsible for handling communications between work stations and may also maintain other programs (such as a mail-forwarding system for other networks) that are of general use and need to run independently of the condition of the individual work stations. In some networks (e.g., the Hewlett Packard), the file server is no more than a specialized work station with added mass storage.

b. <u>A General-Purpose Data Base Manager</u>. In this scenario, the data server is designed to perform data base functions. It may also be the case that the hardware and software of the data server are optimized for this function; it may be a "data base machine." This means that the work stations must communicate in some kind of intermediate language that could be anything from the low-level FINDs and STOREs (of Codasyl [Ref. 22], for example) to the high-level operations of the relational calculus or some other query language (see Ref. 10 for details). Whether or not this node can also perform the other functions of a central node described above is a matter of design. But it may be that the performance of these functions will impede its efficiency as a data server.

c. <u>A Special-Purpose Data Manager</u>. In this case the data server is restricted to maintaining a predefined data base and the communication with other nodes is even more restricted. This case is worthy of study since in many developing networks an existing computer is added as a node. This machine must support a number of existing applications programs that exploit an existing data base. To other nodes on the network, this machine may be treated as a special-purpose data base manager. It appears that early efforts to develop a back-end data base machine (Ref. 23) concentrated on providing such a service with a fixed schema data base management system.

Back-end data base systems are dealt with extensively in Ref. 24. The main conclusion of this paper is that the machine supporting these systems must support a multi-programmable operating system: it must be possible to run several programs simultaneously in much the same fashion as a time-sharing system can support simultaneously the demands of several users. Even in the case of an existing machine being added as a data base server to a network, some form of multi-programmable interface is desirable. In the case that the software is unavailable or inconvenient to run on the existing machine, some separate multi-programmable interface node is desirable. Such a node could, with minimal disruption to the existing machine, run in terminal emulation mode--much as is done in the Adapt (Ref. 25) projects.

One of the practical problems concerns what combinations of software best achieve a natural multi-programmable interface to data bases. Before discussing this, consider the simplest case of a data server--that of a file server--since it illustrates in a simple way a few of the problems of distribution of control between the server and the machine that is communicating with it (the user). In its simplest form, communication between the user and server is simply a reflection of the subroutines normally found in operating systems for file i/o:

(1) OPEN(<filename>, <control>). Note that at the server, <filename> must have an indication of the user. If accounting is a problem, account information must be supplied.

(2) READ(<filename>, <control>, <data>) and WRITE(<filename>, <control>, <data>) operate as usual, as does

(3) CLOSE(<filename>, <control>).

The only immediately obvious problems are (a) that acknowledgments and error states must be communicated back from the user to the host via messages rather than through values returned from subroutines or through modification of control blocks, and (b) that security may prove a problem. The second of these will be dealt with later; the former can easily be achieved and the file interface can, in fact, be made transparent to an applications program by making a set of subroutines transmit and receive the messages. This interface is then at the seventh ("application") layer of the ISO layer of communication protocols.

There are problems that are not completely solved by this simple view. Most of these have to do with "meta-data" or schema information. A simple example is that of default subdirectories. In most operating systems, a user may specify that his working environment is a subdirectory. All file names are then automatically prefixed with that subdirectory name. There may be some additional secondary storage structures to enhance the manipulation of subdirectories. Where is the knowledge of the current subdirectory to reside? If it resides at the server, additional communication protocols must be set up to define and create subdirectories. If it resides at the host, the host must maintain checks to make sure that such subdirectories exist. A similar problem will exist, if it is desired, in the combined user-server operating systems to maintain logical (as opposed to physical) file names.

These problems are special cases of the problem of schema distribution. Figures 5 and 6a-6b, taken from Ref. 24, show the distribution of data base functions across a network.

In Figure 5, a conventional single-machine data base architecture is shown. This diagram is based, more or less, on the assumption that a Codasyl DBMS is in use. It is not entirely accurate as a description of relational data base architecture.

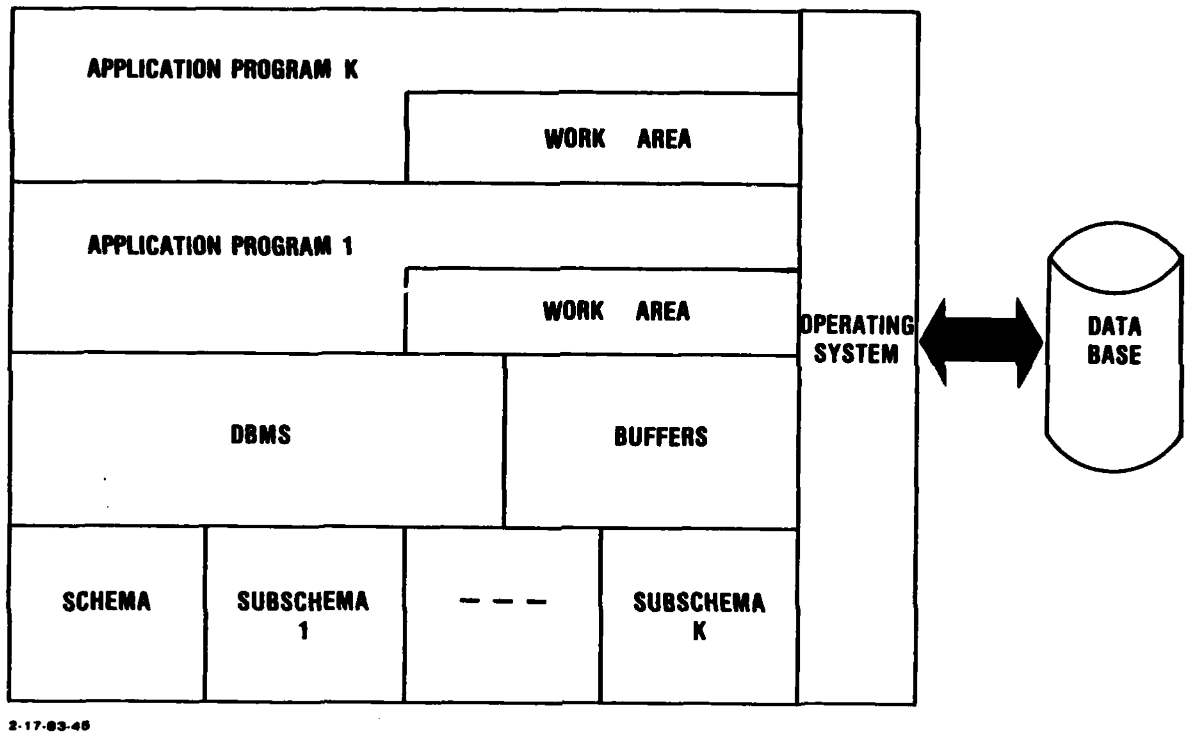


FIGURE 5. Single-Machine DBMS

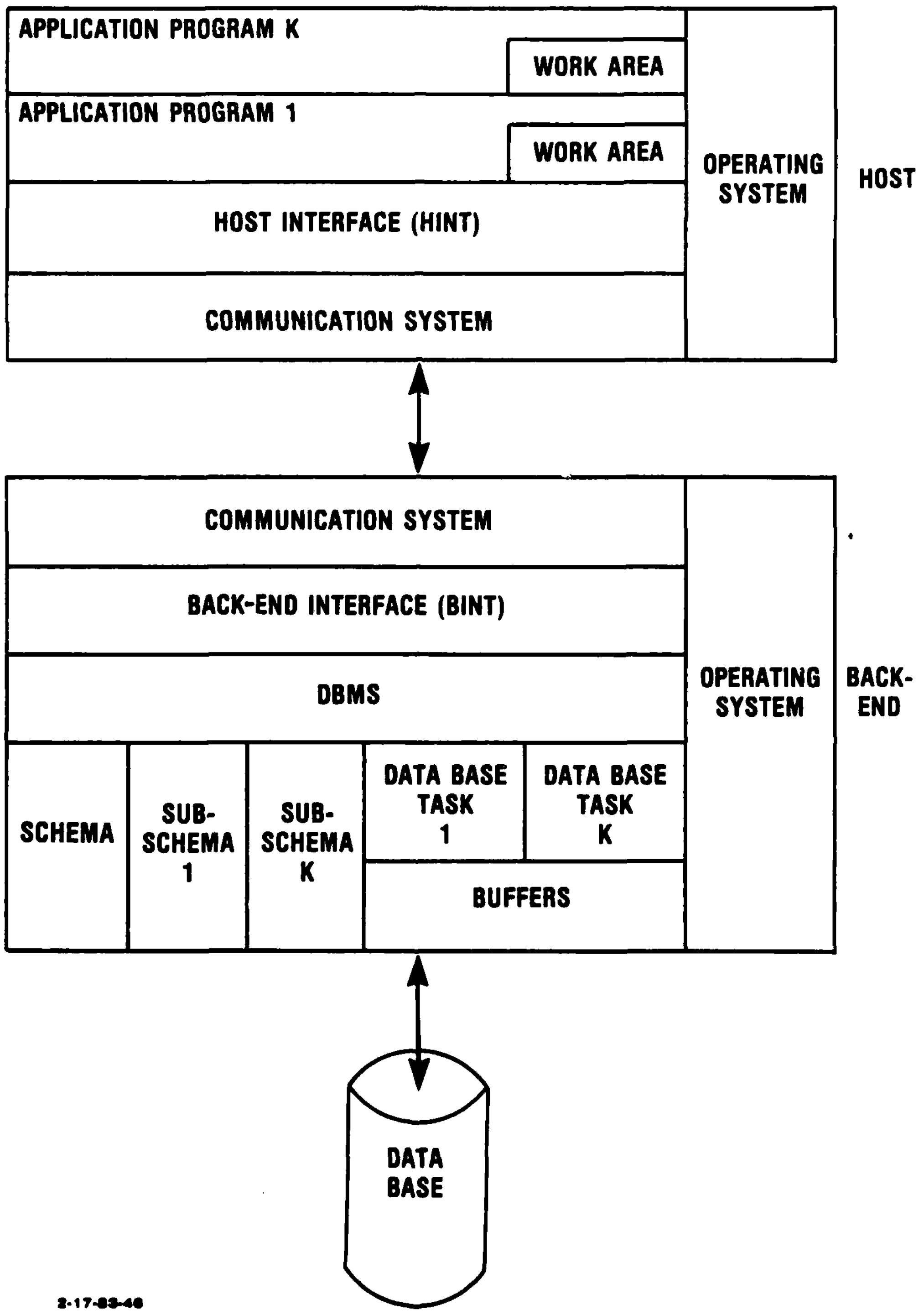


FIGURE 6a. Software Organization of Back-End DBMS

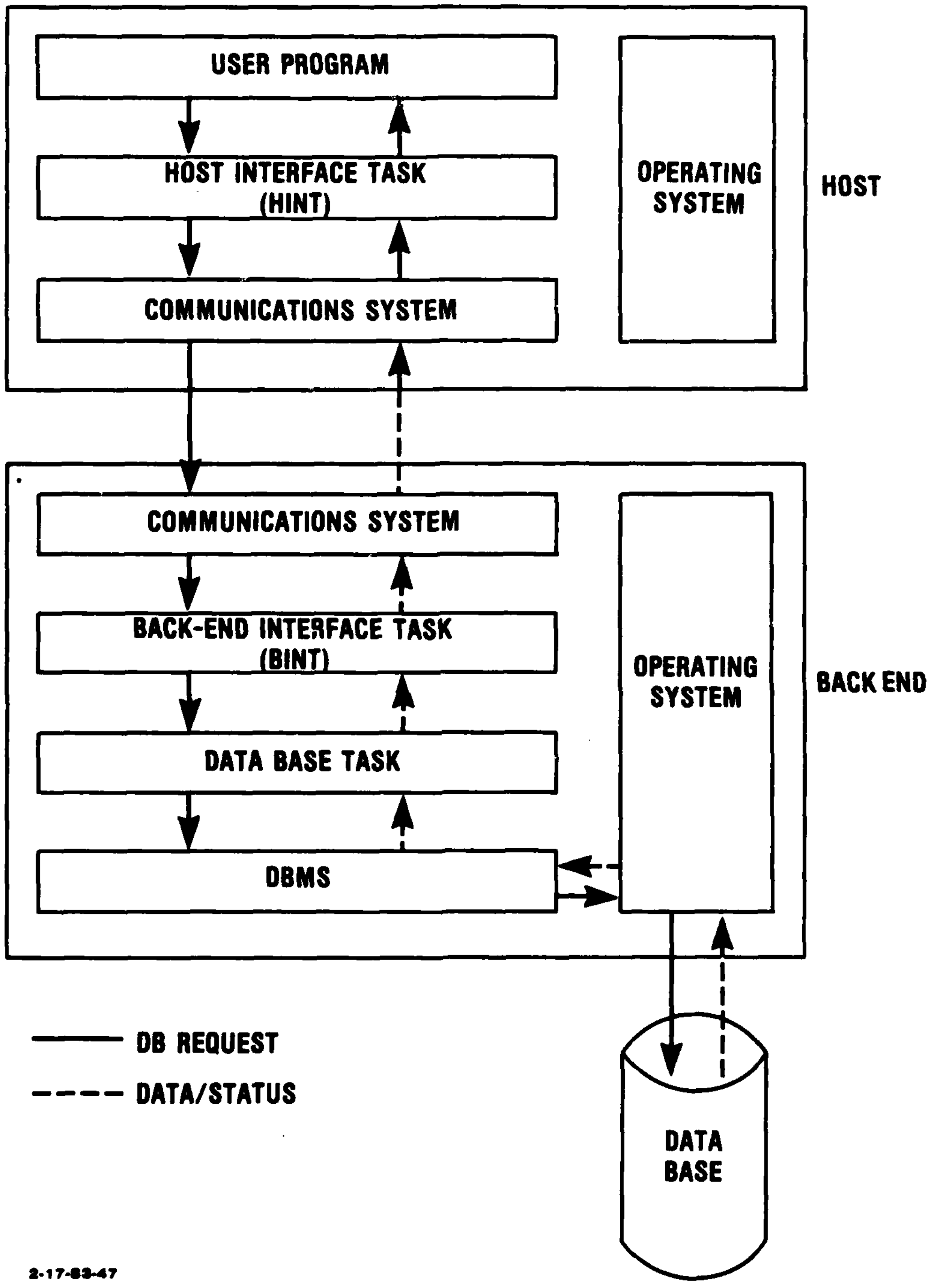


FIGURE 6b. Information Flow in Back-End DBMS

In a Codasyl (Ref. 22) data base management system,the applications program communicates with the data base through a fixed set of "registers"--one for each record type or set type in the data base. A number of "data manipulation" subroutines are available to transfer data to and from secondary stroage. These registers are usually called the user working area (UWA) and it is these that reside on the user machine in most elementary kinds of distributed architecture. All other data base access functions are performed by the server. In fact, as far as an application program is concerned, the data manipulation subroutines could appear to be exactly the same as they are in conventional Codasyl DBMS, the only difference being that they are initiating and receiving messages rather than calling directly subroutines that are excuted on the user machine.

Figures 6a-6b are not, however, a complete picture of what is needed in many cases. It shows the structures needed to support and run an application program once the data base has been established. There are several other practical considerations:

(1) A substantial amount of data base software is concerned with the creation of new data bases. These include programs to compile data definition languages, initialize files, etc. At the same time there are a number of data base restructuring tools for mapping (logically or physically) the data base. Programs to carry out these tasks must be callable on the network.

(2) It is likely, given that a back-end machine can act as a data server, that its services as a repository for large files will also be desirable. (Some data base management systems have a built-in provision for file storage, but this is unusual.) A further group of file creation and access commands, of the form described above, would need to be implemented.

(3) In this design, concurrent access is treated exactly as it is in a single-machine DBMS. This usually

takes the form of (a) data base locking (only one program can write to the data base), (b) area locking, in which physical areas of the data base may be independently write-locked, and (c) record locking. Although these are, in the order given, increasingly powerful tools, they have the associated difficulty of providing for more sophisticated concurrency control mechanisms.

If the first two of these problems are solved by the data server itself, then the function of the back-end machine is effectively the same as any operating system. What has been achieved is a degree of distributed processing that permits (though there is no evidence that this was ever attempted) the operating system of the server to be optimized for data base work.

Implementations of this sort call for "remote procedure calls." These are procedure calls that call upon resources on some remote processor and that are implemented by message passing. Recent work at Xerox Parc has resulted in the development of such calls, and what is particularly relevant here is that these have been implemented across different languages and processes and, to a certain extent, preserve data types across languages.

The survey (Ref. 24) includes reviews of some data base back end projects, including several developed for military contractors. They include XDMS (Ref. 23), IDMS (Ref. 26), KSU (Ref. 27), and MADMAN (Ref. 28).

C. PERSONAL DATA BASES

It was mentioned earlier that users of personal work stations who are interested in a variety of data bases are likely to want to perform their own manipulation of data by constructing their own "local" data bases. Examples of this are already common: most small computers support software that enables the computer to emulate a terminal and simultaneously to record the conversation between the terminal and remote host in a local file. By having the host "type" a file on the terminal, the file may be moved from the host to the processor to be subsequently manipulated. This is a crude but effective method of information interchange between machines and can be used as the basis of building simple textual data bases. Another example of something that approximates a local data base is the "local file" of Adapt. A user may direct the output of a data base query into a file, and that file may subsequently be used on other queries.

In considering the impact of local data bases on the end-user, it should be borne in mind that the user of a personal work station is likely to have a much more friendly, responsive and simple computing environment than has been traditionally available on time-shared mainframes. Experience of the home computer market has shown that unsophisticated users are quite willing to learn new programming techniques to develop their own applications rather than use existing packages that fail to do exactly what was required. An important consideration in C^3I environments is the threat to security posed by this increasing sophistication of end users.

In this section we shall review the potential uses for "personal machines" in C^3 networks, paying particular attention to the problem of security.

1. General Requirements

A number of data base management systems are currently available in the micro-computer market. These are mostly "flat-file" manipulation packages, and although they are often sold as relational systems, they cannot be regarded as such because, logically, they do not properly support join operations and, physically, they usually have inadequate indexing methods.

For anything but the simplest forms of manipulation, these systems are inadequate. They indicate, however, that what is required in a "personal" data base management system differs from conventional systems in a number of crucial respects.

1. Most important is extreme flexibility. Users cannot be expected to design a data base in advance. It must be incrementally built from various sources of data, and have good restructuring tools.
2. It must be possible to load and unload portions of the data base into text files. Data are seldom added by the end user if it is possible to use an existing source such as a report. It should be possible to incorporate new data classes by editing a report and loading it into the data base. Similarly, it is extremely important, for the purpose of generating reports, to create text files and incorporate these into reports.
3. There must be a user-friendly data manipulation language. Discussions of this subject, including the IDA report (Ref. 10), have been devoted to the nature of these languages and the trade-off between the power and ease of use of the language.

The only systems that currently satisfy these criteria are the relational systems. Ingres (Ref. 29) has been developed for a PDP-11 under Unix and is thus likely to be developed for the many other personal work stations that run this operating system. It is also likely that similar data base management systems, probably based on the relational model, will be developed for this kind of system. The main problem commonly cited with Ingres and other relational systems is that of efficiency: support for data bases with relations containing 100,000 records may prove difficult. Part of the problem is with PDP-11 architecture and the way in which (for reasons of limited memory) Ingres is implemented through multi-tasking. These limitations will be removed on more advanced machines. The other limitation on performance is that flexible indexing methods (usually B-Trees) are, generally speaking, less efficient than the more constrained methods such as hash tables. This is a necessary consequence of the desire to have a flexible data base management system and will only be improved by using new machine architecture and, possibly, by using another data model.

2. Other Data Management Methods

Recently, other data base management systems have emerged for the personal work station. For example, a version of SEED, a Codasyl system, is available for the Apollo. This was mentioned earlier as one of the few distributed data base management systems available for local area networks. While SEED has a variety of restructuring tools, and has a remarkably simple query language, it still cannot compete in flexibility with the relational systems. It will probably prove more efficient for large data bases.

It should also be noted that many users perform relatively sophisticated tasks by the manipulation of text files. Again, under the Unix operating system, there are a variety of software "tools" that are mostly text formatters. They are sufficiently powerful that most of the operation of a relational data base management system can also be performed through manipulation of text files. Examples of commonly used tools are a sorting program, a file merging program, and a regular expression matcher that can be used to filter out and reformat lines of a text file. Experts in such a system regularly use these tools for keeping "personal" data bases (address books, etc.) and also for large data manipulation problems. The efficiency limitations of this method of data manipulation are considerably more severe than those of relational systems, and much more sophistication is required to use these tools properly.

3. Data Base Integration and Security

As common data base interfaces are being introduced into the C^3I community, especially through the adoption of ADAPT (Ref. 30) in certain areas, there is an increasing problem of data base integration. Two extant data bases may have considerable overlays in two possible ways:

1. There may be an overlap in the data contained in these two data bases. Usually, this is partial.
2. The same kinds of data are to be stored in the two data bases even if there is no overlap in the actual data elements.

As an example, consider the personnel data maintained by two organizations. If one organization is a part of the other, we may expect the records in one to be related to records in another. However, the records in the two data bases may have differing structures (different fields). Therefore, in merging two data bases, one has to decide on a common representation and to determine what (if any) data elements are common to both data bases. One can imagine either a physical merge in which the two data bases are physically combined into a new data base or a virtual merge, in which the two data bases remain physically separate, but the user perceives, through the interface, a single uniformly standard data base.

The problem of integration becomes more complicated as the physical structures that support a data base become more intricate (e.g., two different Codasyl designs for the same body of data. The problem of providing tools for the merging and restructuring of data bases has only recently been studied (Ref. 31) and, to our knowledge, no useful tools have yet emerged. The general results of the research so far indicate that the richer the data model, the more likely one is to be able to perform a merge automatically or semiautomatically. What this means is that if the data base is, for example, conceptually structured as a relational data base with no information, such as functional dependencies, about relationships between domains, there is little hope of finding any general-purpose integration method. If, however, the conceptual model is richer and involves the various relationships (such as functional dependencies, subtypes, etc.) that we commonly used to describe richer data structures, then, to some extent, the integration problem can be solved automatically.

A particular example of this need has come to our attention that involves the merging of intelligence data files. There are usually files of documents, or other text fragments, with some kind of indexing structure. There are three kinds of indexing in common use.

1. Each document is assigned a set of key words. A user wishing to retrieve a document specifies a set of key words and those documents whose key word set is "close" to the user's set are retrieved. The measure of closeness is usually taken to be such that a manageably small number of documents are retrieved on a first pass.
2. Hierarchical indexing. A hierarchy (taxonomy) of subject matter is constructed and each document is assigned a position in this hierarchy. In some systems it is possible to place a document at two positions in the hierarchy.
3. Content addressing. With sophisticated hardware, it is possible to retrieve documents by specifying one or more text fragments in those documents. For example, "smart" disk handlers have been constructed that will search a text file of the order of 10^8-10^9 bytes in a second. By increasing parallelism, it should be possible to perform this search even faster.

Methods 1 and 2 are in widespread use, and a typical example of a merging problem is to have two different hierarchical descriptions of the same document set. It is possible that the same term used in the two hierarchies will not be used in the same way. It may be impossible, therefore, to find a unifying hierarchy for both document sets. The best that can be hoped for is a set of good interactive tools that allow an analyst reclassifying documents in one hierarchical system to be able to use the other as a starting point. It should be relatively straightforward to construct an interactive program to do this.

4. Security Considerations for Advanced Work Stations

There are two considerations that may prove important in the use of advanced work stations on a network when they are being used as "smart" terminals.

1. Because of the ability to manipulate data locally: to edit, search, and cross-correlate by the methods described above, the need for hard copy is reduced. That is, there should be less need to take notes or printed forms away from the work station, thereby eliminating one of the most common threats to security in many commercial establishments.
2. Since data may be "sucked" out of a central data base and manipulated locally, the user has access to all kinds of aggregate information that may normally be regarded as sensitive or classified even when the individual data items may be unclassified. Note that in this situation security is normally guaranteed only by lack of efficiency of the users or by the speed of the system. This is clearly unsatisfactory.

These two considerations, respectively, agree for and against user-programmable work stations. Commercially, the increased productivity resulting from their introduction usually outweighs the threat posed for security. The decision in military establishments may, in the short run, go the other way.

D. INTEGRATED DATA BASES AND PROGRAMMING LANGUAGES

Previous reports have concentrated on the problem of a unified network query language. Since there are attempts to standardize both programming languages and data bases, it appears appropriate to append a brief survey of the various methods used to produce integrated programming/data base environments.

1. Query Language

As noted in Ref. 10, it is only relatively recently that interactive systems have been made simple enough--and that users have been deemed competent enough--that end users have been able to perform their own general-purpose data base manipulation. These systems are called query languages (a misnomer because they usually allow update) and have been developed both for relational and Codasyl data bases. Among the relational query languages SEQUEL (Ref. 32) and SQL are probably the best known. These are nonprocedural in the sense that the user does not have to describe a specific control structure for his query. Another notable non-procedural system for relational data base access is QBE (Query-By-Example) (Ref. 33), in which the user interacts with a set of tabular displays on a CRT screen to form relational queries.

Query languages have also been developed for Codasyl and other nonrelational data bases. These range in power from semi-procedural languages that provide the kind of interactive programming one has with BASIC, to sophisticated non-procedural (Ref. 34) languages that attempt to determine the correct paths in the schema needed to resolve a query, when this has not been completely specified by the user.

We shall see that most of these languages, while being extremely convenient for end-user interactions, lack the power of conventional programming languages to perform arbitrary computation of calls to the operating system. These limitations will be discussed later.

2. Interfaces Through External Subroutines

In contrast to the simple forms of data base interaction provided by query languages, most data base applications for network and hierarchical data bases are written in a lower-level language such as COBOL, FORTRAN, or PL/1, where the data base traversal consists of setting up certain conditions and then successively calling a "find" subroutine that extracts the records (or other data base components) that satisfy those conditions. Such a program is compiled, having possibly first been run through a preprocessor, and then linked to the appropriate precompiled data base subroutines.

Apart from the obviously increased complexity of the programming environment, it is unfortunate that the programming language interface for Codasyl was originally defined with Cobol in mind: a programming language without parameterized subroutines. This means it is straightforward to write simple data base traversals through the use of such subroutines, for such programs resemble sequential file access, for which Cobol was designed. On the other hand, complex traversals are extremely cumbersome, requiring the programmer to worry about "data currency" and other side effects that are needed to communicate with unparameterized subroutines. It may well be that it is these rather peripheral annoyances, rather than lack of programming ability, that have traditionally placed a barrier between the applications programmer and the end user.

A general problem with such interfaces is how the data base structures--records, sets, etc.--should be represented as data types within the host language. The answer is simple in a language with a simple type system such as FORTRAN: the data base can only provide integers, reals, arrays, and possibly character strings. The problem becomes more acute when one tries to perform a clean integration with the type systems of languages, such as PASCAL, with a richer type system.

3. Two Language Systems

In order to achieve the convenience of a high-level query language and the power of a programming language, a hybrid system exploiting both kinds of language is often used. An example of this is the relational data base interfaces that allow PL/1, say, to communicate with a relational data base through a query language such as SEQUEL. Here, a SEQUEL query is formed as a character string within the PL/1 program. This is then passed to a processor that creates, if the query returns output in the form of a relation, a structure that may be sequentially traversed in much the same fashion as a sequential file.

4. Programming Languages With Integrated Data Bases

A number of attempts have been made to design new languages with integrated data bases. Plain (Ref. 35), and Rigel (Ref. 36) are examples with an Algol-like syntax, and Taxis (Ref. 37), which also fall into this category, has a particularly interesting data model and also provides facilities for coping with the kinds of exceptions that are common in data base applications.

5. Data Base Extensions For Existing Programming Languages

It was noted earlier that it was often difficult to reconcile the type system of a programming language with the implicit type system of data base. One way out of this is to extend the programming language to support new data types that correspond to the data base structures. Thus, for example, Pascal-R (Ref. 38) extends Pascal with a new relational data type, and makes available within Pascal the relational operators and allows data to be transferred between tuples of a relation and other Pascal data structures. Also under development is Adaplex

(Ref. 39), in which an extension to the type system of Ada permits access to a "functional" data base (see below).

6. Languages For Multiple Data Bases

Since it is quite usual to find users who require access to several data bases of differing structure (relational, network, etc.) there is an obvious need for a language which provides reasonably uniform access to these. It is not at all clear that, in view of the differing data models, any simple language could be constructed in which the various access methods could all be represented. However, several independent suggestions have been made that indicate that a functional data model can go some way toward representing the various commonly used data models and can capture some of the more recent "semantic" data models. This is used in Daplex (Ref. 40) (upon which Adaplex, mentioned above, is based), and is the interface for functional programming systems (Ref. 41). For the latter, interfaces to both Codasyl and Relational data bases have been built.

7. Data Bases For Multiple Languages

By contrast to the preceding section, there is also a need to have a data base management system that can look like different systems to different applications. The reason that this is needed is that one often wants to replace an existing data base with a new (and perhaps more flexible or efficient) system, and maintain existing applications programs. For this reason, one would like to have a data base that can look, say, like a Codasyl data base to an applications program that supports a Codasyl data base and like a relational data base to an end user who wants to used a relational query language.

The purpose of this brief survey is to indicate that there is even more heterogeneity among data base/programming language interfaces than there is among data base query languages. This is an area that urgently needs further study.

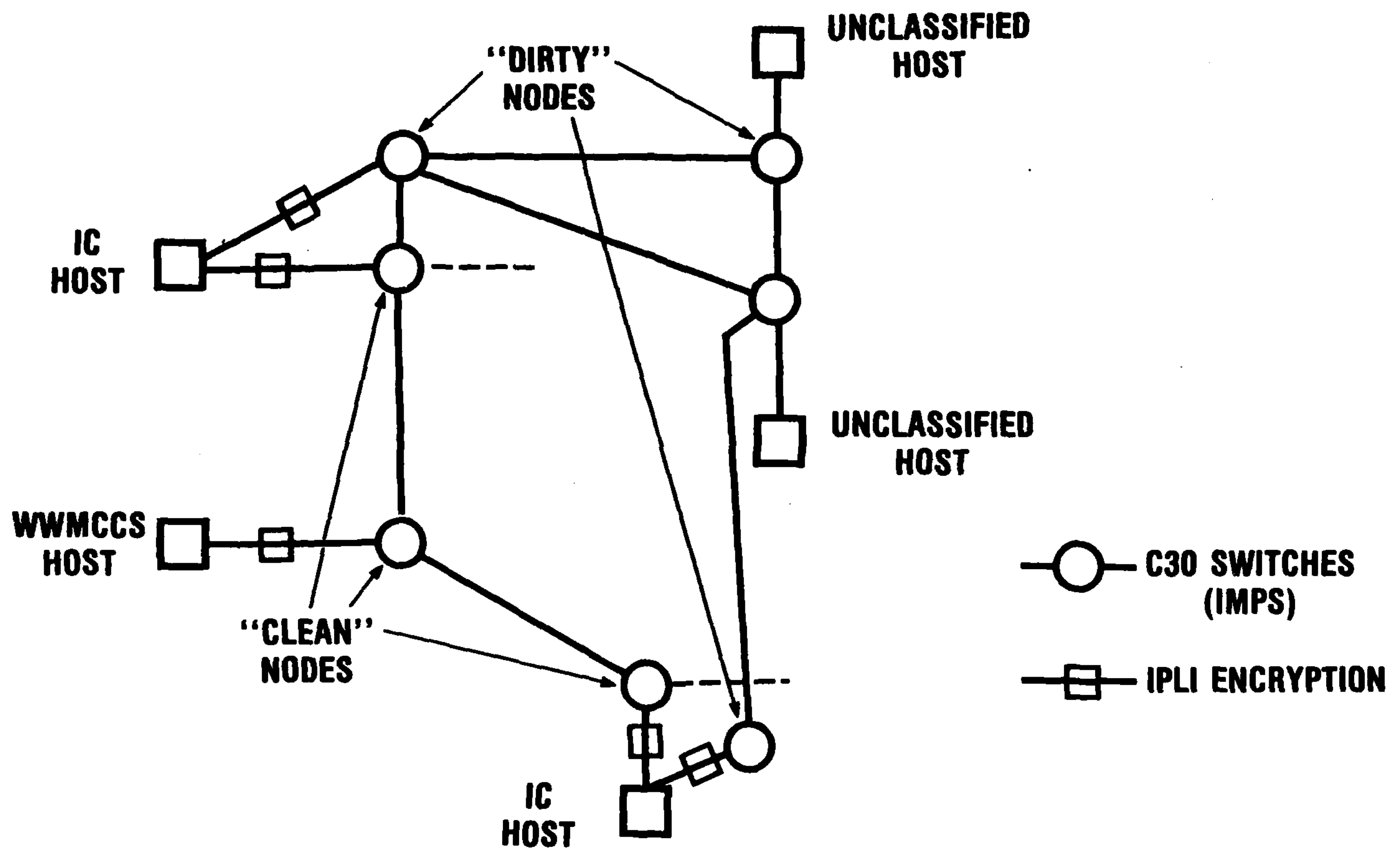
APPENDIX

SECURITY ARCHITECTURES FOR DDN

APPENDIX
SECURITY ARCHITECTURES FOR DDN

The following considerations present pros and cons for several architecture options for security in the Defense Data Network, and lead to the selection of two as the best developmental procedure. The basic recommendation is to proceed with construction of two separate nets and then to interconnect them if and when the development strategies prove out.

We now present a description of the options for a security architecture for the global network for what is now called the Defense Data Network (DDN). Several designs for the DDN have been made by DCA. In the earliest design, a single network was planned with two kinds of nodes (switches)--which were called "clean" and "dirty" nodes. This is shown in Fig. A-1. Here the C^2I host computers (such as WWMCCS and IC computers) are connected to "clean" switches which only interconnect to other C^2I host computers and which are all operated at a TS level. The "dirty" nodes are part of a subnetwork which interconnects unclassified hosts as well as C^2I and Secret hosts. There are connections between the clean and dirty nodes; however, these connections would be used only when circumstances force the mixing of unclassified and C^2I traffic. (The routing tables in the switches form this partition.)



NOTE: All switch-to-switch links are link-encrypted with KGXXs at each end of each link.
12-30-82-1

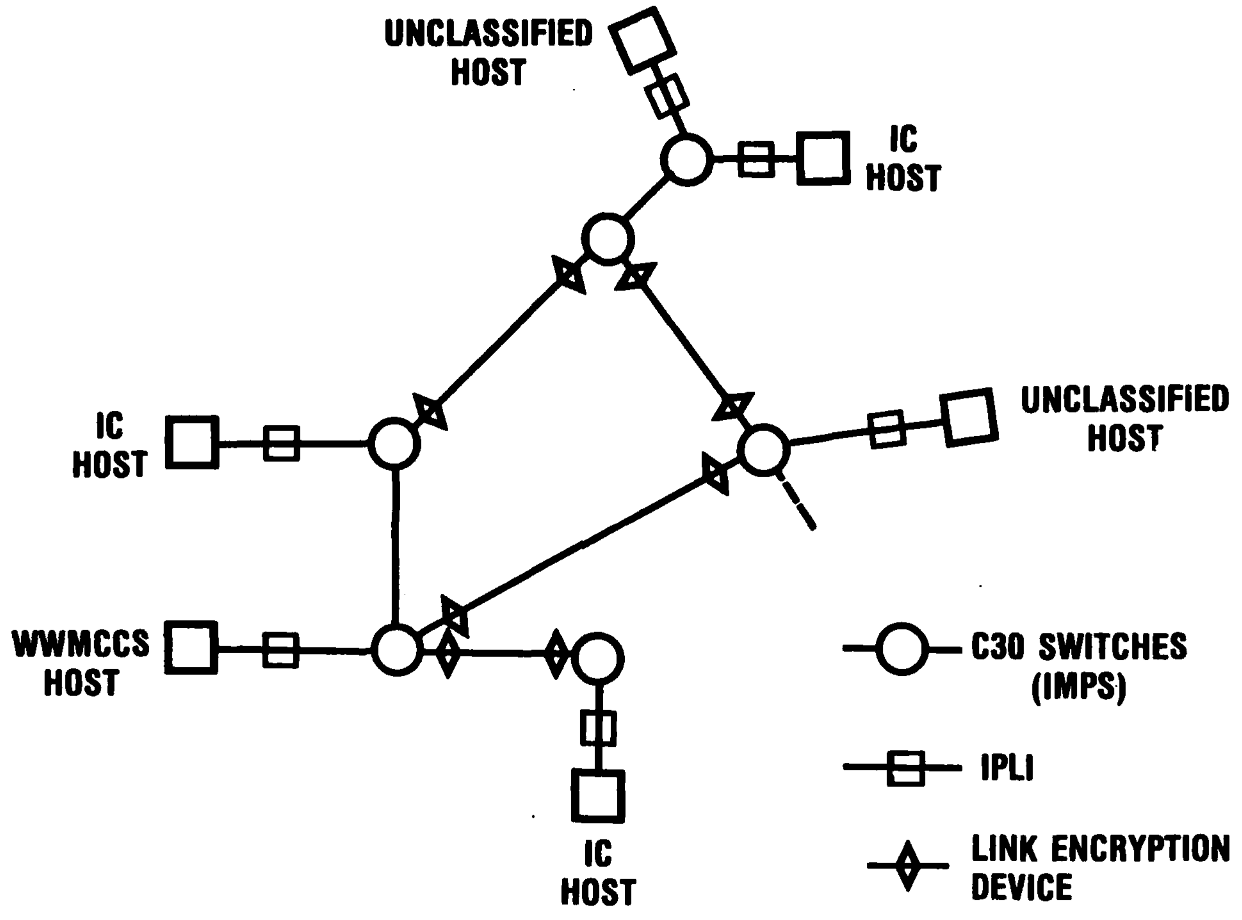
FIGURE A-1. Early Security Architecture for DDN

This architecture was later criticized by NSA, who objected to the ability of unclassified hosts to access both the dirty nodes and perhaps the clean nodes through the network. As a possible problem, if hidden software routines designed to disable nodes existed in switch software, an unclassified host could call these routines and bring the network down.

In order to alleviate this problem, NSA proposed placing an IPLI (Internet Private Line Interface) on the line to each unclassified host. The IPLIs were to be collocated with the switches, which were all to be operated at the Secret level. The encryption of the data by the IPLI would make passage of signals to the network software difficult (although the header would be only simply encrypted and length, frequency, etc., could also still be used in a relatively unsophisticated scheme).

The placing of IPLIs on unclassified users' input lines meant that all users would now use IPLIs (Secret and C^2I hosts always use IPLIs in mixed designs). The resulting architecture was called the "IPLIs everywhere" architecture. (It is also referred to as "Option 1" in a letter written by Dr. Latham, directing DCA and NSA to investigate the possible architectures [Ref. A-1].) When IPLIs are used everywhere, DCA designs then call for a single network with all nodes operated and maintained at the Secret level. This is shown in Figs. A-2a-A-2b. Figure A-2a shows the general layout while Fig. A-2b shows a specific layout for a two host, three communications line node. A major objection to this architecture concerns the cost, since unclassified users would require IPLIs. Also, nodes would have to be operated at a classified level, which is expensive, and this expense, along with maintenance costs and procedural problems which might arise, made the architecture appear expensive for unclassified users.

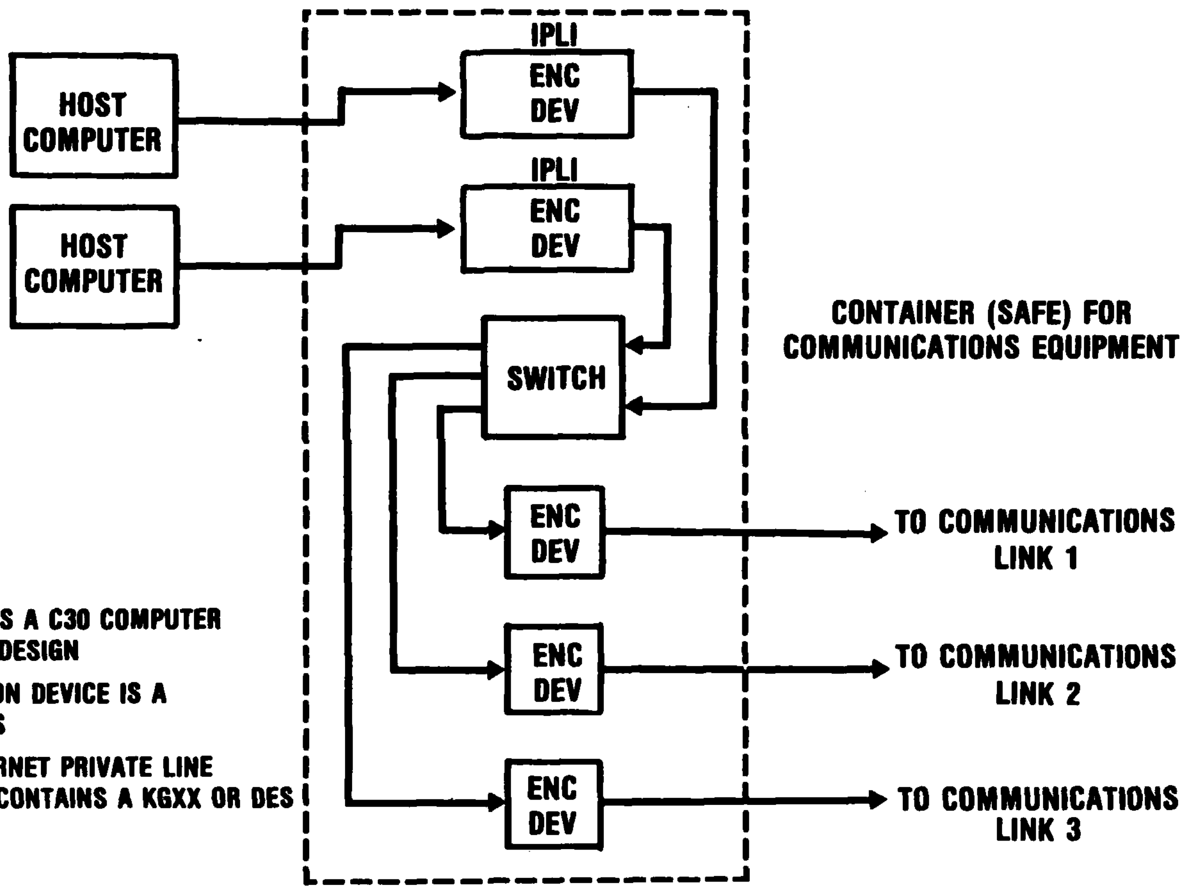
From the C^2I users' viewpoint, there is a further objection, since unclassified traffic is passed through switches at C^2I nodes and, for example, SCI and unclassified data are being



NOTE: IPLIs for unclassified hosts are collocated with C30 switches. All switch-to-switch links are link-encrypted.

12-30-82-2

FIGURE A-2a. "IPLI Everywhere" Architecture



LEGEND:

THE SWITCH IS A C30 COMPUTER IN THE DDN DESIGN

AN ENCRYPTION DEVICE IS A KGXX OR DES

AN IPLI (INTERNET PRIVATE LINE INTERFACE) CONTAINS A KGXX OR DES

7-13-82-6

FIGURE A-2b. Layout for 2 Host, 3 Communications Link Node in an "IPLI Everywhere" Net mixed throughout the network. This could present a threat from a security viewpoint, as well as causing problems with service, traffic flooding, etc.

Figures A-3a-A-3c and Fig. A-4 show designs for "C^2I" and a "Milnet" backbone network, which are separate.* Using two networks alleviates security and access problems for C^2I users in a crisis by simply not allowing unclassified users access to the nodes. The net can then be designed to the requirements of C^2 users with regard to survivability, traffic handling, etc. The unclassified "Milnet" network can also be designed to the specific needs of unclassified users, permitting economy and other considerations to be emphasized without C^2I requirements intruding.

DCA has proposed several variations or options on the two separate network architectures. The first of these is shown in Fig. A-5. In this the two networks are normally separate, but can be connected together in case of crisis. The connection is by means of some device with the properties of a "knife switch" or gate which can be closed on command. The routing tables in the switches must also be updated at that time, and there are considerable security considerations and unclassified network message blocking problems to be worked out. There is also one clear major objection--the unclassified user again has access to C^2I nodes after the switches are closed. This option is often referred to as "Option 2.1" (Ref. A-1).

There are two more variations on Option 2 which are being considered. Both of these are shown in Fig. A-6. In this case, however, the interconnections between networks consist of devices which appear to the subnetworks as either links, hosts, or

*These are from an IDA letter report entitled, "AUTODIN II Alternatives," (December 1981). Only "backbone" or beginning of Milnet is shown. Many more users would be added.

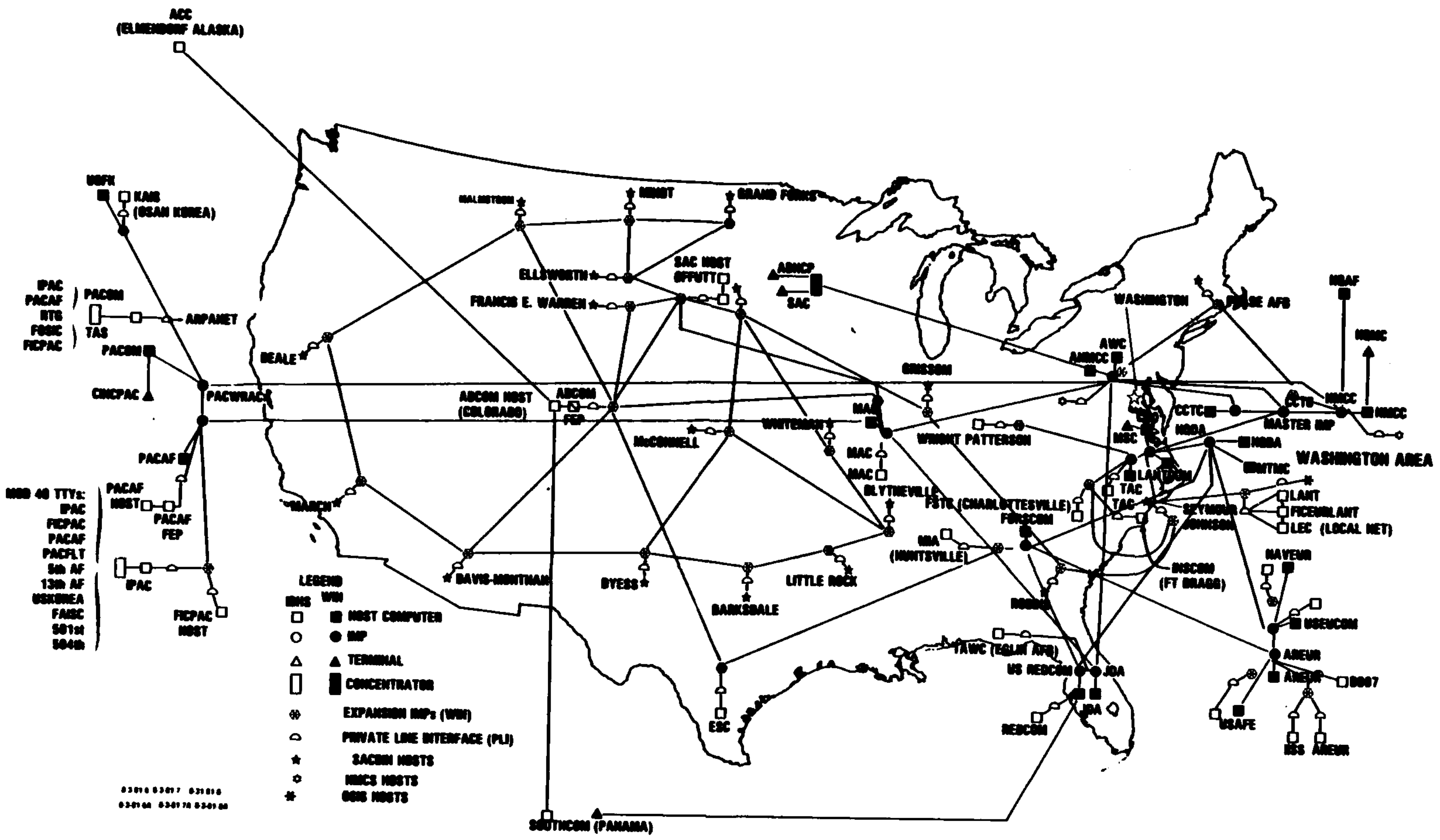


FIGURE A-3a. C^{2}I Network--Evolutionary Design

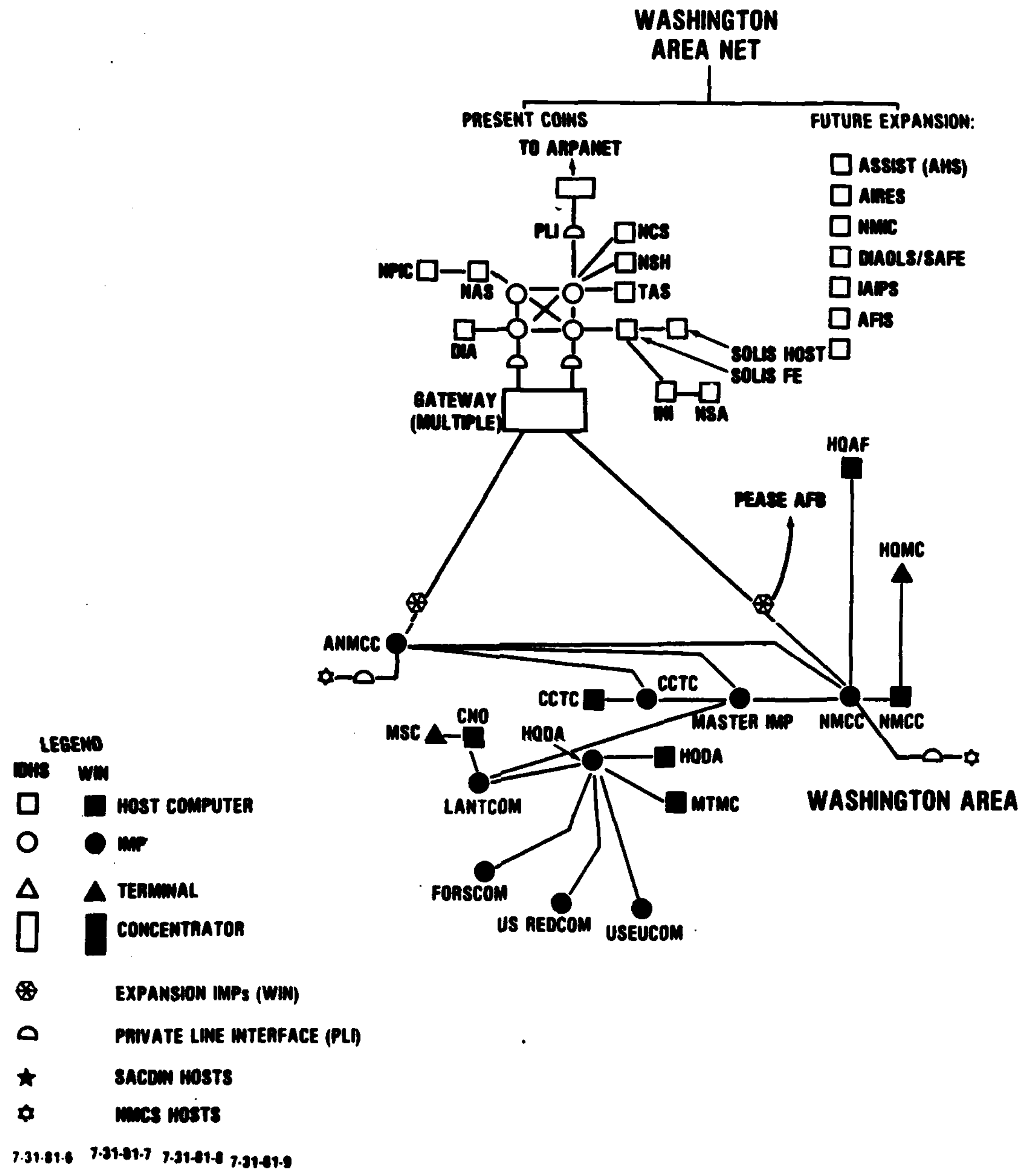


FIGURE A-3b. Washington Area Net

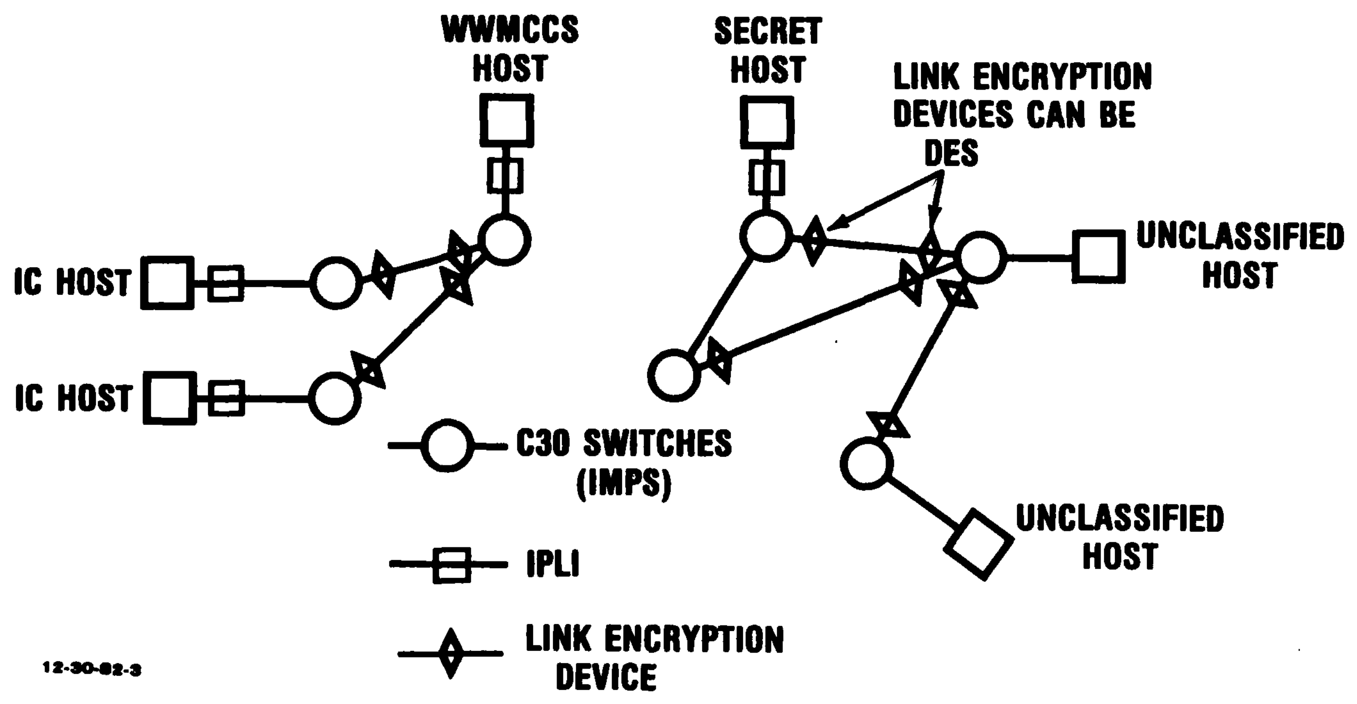


FIGURE A-3c. Security Architecture with Separate Nets for C^2I and Unclassified-Secret Hosts

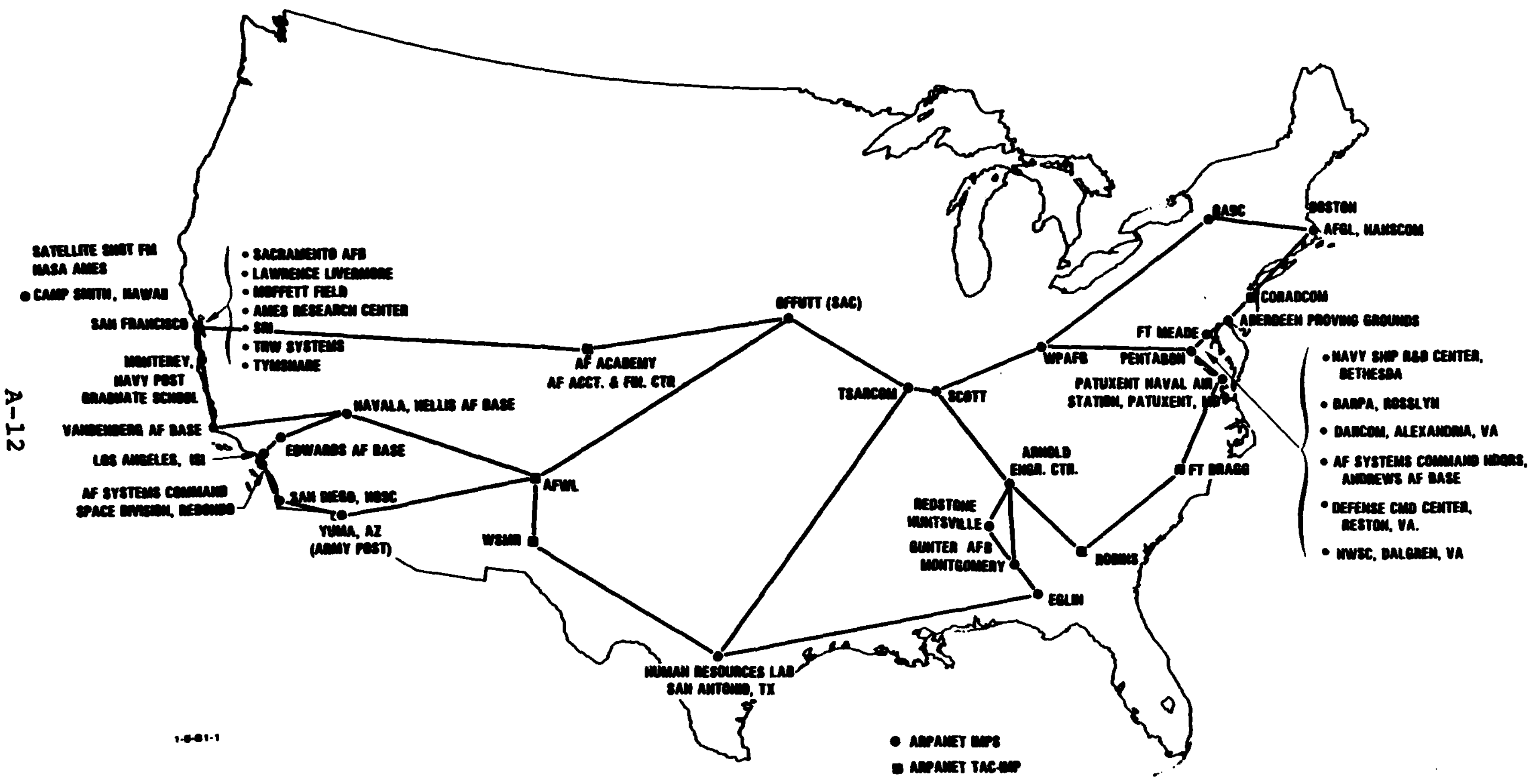
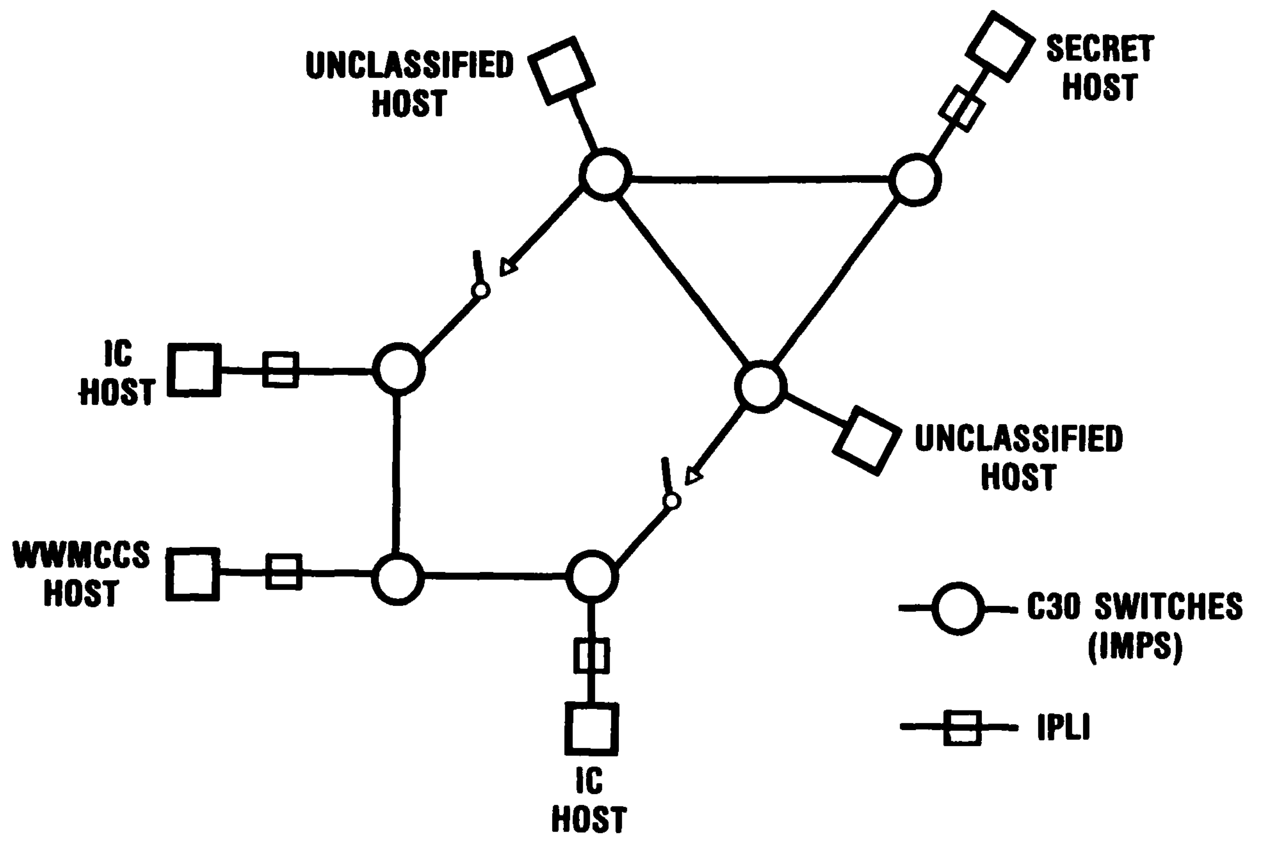


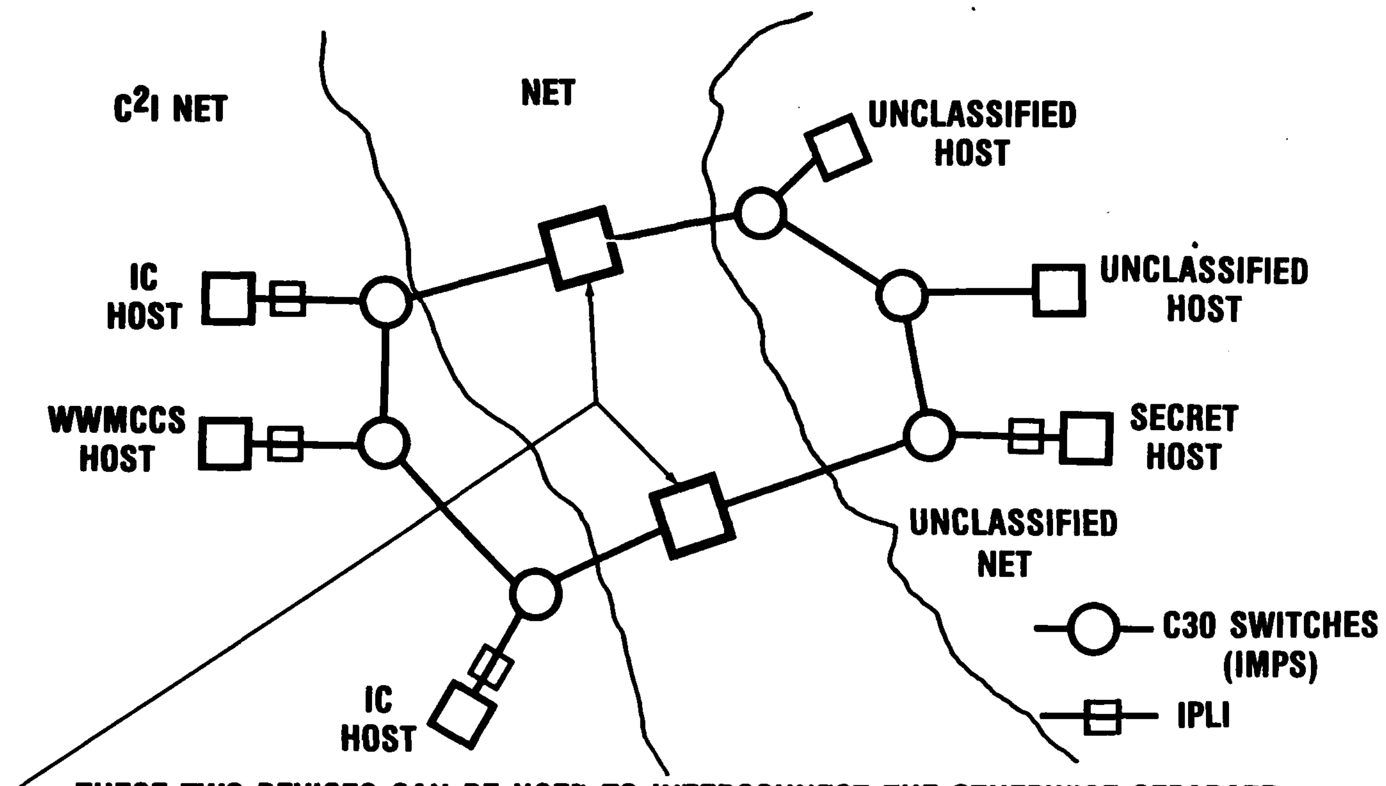
FIGURE A-4. MILNET Backbone



NOTE: In normal operation, unclassified and C^2I nets are separate; however, in a crisis they can be connected by closing the ''switches'' (gates). This is also referred to as Option 2.1.

12-30-82-4

FIGURE A-5. Security Architecture with Separate Nets which Can be Interconnected when Necessary Using "Gates" or "Switches"



THESE TWO DEVICES CAN BE USED TO INTERCONNECT THE OTHERWISE SEPARATE NETWORKS. THERE ARE TWO STRATEGIES — IN BOTH, THE DEVICES APPEAR AS LINKS TO C^{2}I SWITCHES. IN ONE THEY APPEAR AS HOSTS TO SWITCHES IN THE UNCLASSIFIED SECTION AND IN THE OTHER THEY APPEAR AS SWITCHES.

NOTE: This configuration is also referred to as Option 2.2 and Option 2.3.

12-30-82-6

FIGURE A-6. Separate Nets with Interconnecting Devices switches. In each case, the two subnetworks are operated normally; however, the routing tables are adjusted so that the C^2I subnetwork "sees" the bridging devices as links to selected switches and can use them to give extra survivability or traffic handling capability. Using this scheme, the bridging devices can then pass C^2I messages through the unclassified net to other bridging devices. The bridging devices must therefore have some routing and routing table updating capabilities to coordinate these messages properly. Also, the packets will be encrypted by the bridging devices and new headers placed on them when they come out of the C^2I net and into the unclassified section, and these headers will be removed when the transition back into the C^2I net is made. To switches in the unclassified section, the bridging devices appear as either hosts or switches. DES can be used in the unclassified section (Refs. A-2 through A-5). These two options are sometimes referred to as Options 2.2 and 2.3.

The security aspects of these two variations are now being investigated by NSA. The routing and routing table strategies are being worked on by DCA. Since there has been no experience with this kind of network, the present work is original and much research needs to be done before actual nets are constructed.

Our basic lists of considerations follow.

Option 1: The single network mixing C^2I, secret, and unclassified users (shown in Figs. A-3a-A-3b has these advantages:

1. All users can potentially be interconnected. Only the IPLI keys separate the different communities.
2. The single network could be more survivable, since the additional nodes supplied by the unclassified secret users will provide nodes and links. This must be weighed against the threat to the C^2I part of the system introduced by the large number of unclassified users with access to the net and traffic passing over the net.

3. It was originally felt the single network might be more economical. This is not clear, however, particularly since unclassified users require IPLIs; the geography is not really conducive to reducing link lengths and numbers, and the mixed requirements for different classes of users must be accommodated.

   The necessity of converting normally unclassified user sites to classified operations is probably the overriding economic cost factor. Users from unclassified sites make very high cost estimates in this area and these costs would probably be prohibitive for, as examples, hospitals, personnel, and some logistics users.

Some disadvantages follow:

1. Unclassified users have access to C^2I nodes. Even with IPLIs at each input, signals can be entered into the system using the headers which are only encoded by a single mapping. Also, the large number of nodes increases the community of maintenance men and others with access to the system and its data. While all nodes are secret and IC-(C^2I) data is protected by IPLIs, the dangers are increased by the number of personnel involved. NSA's current position is that this option is undesirable if IPLIs are not used everywhere and marginal if they are.
2. The network is larger, and if DDN becomes more widely used, would be very large. This could cause flooding and access problems due to the size of the network and variations in user requirements.

<u>Option 2</u>: (Figs. A-5 and A-6) - Separate networks. This alternative has these disadvantages:

1. All users do not have access to all system assets. There are those who think that in a crisis everything should be available to, for instance, WWMCCS users.

A possible solution is for such users to have a terminal connected to the second net.

2. Having two nets might lead to having more nets. The solution here is to have adequate management of DoD resources.
3. Separate nets might lower survivability, since the unclassified users generate extra nodes and links and thus alternate paths. However, studies to date show that the way to kill C^2I users is simply to attack the users' bases, and the small incremental survivability gained by the alternate paths is not significant. The C^2I net should be sufficiently robust to survive any attack which does not eliminate the users themselves. Also, not having unclassified users and a large number of users on the net has advantages from a survivability viewpoint, which appears to offset any small gain in link survivability.

Advantages of the separate networks are:

1. From a security viewpoint, each user is more secure than with any other scheme. This is particularly true for C^2I users who are not exposed to unclassified users. Also, the number of personnel having access to link encryption keys is less for each net, reducing the risk of key disclosure for the network.
2. The technical risk is less than that for a single all-secret network, since there are fewer nodes in each network.
3. The single all-secret net IPLI everywhere and two nets with bridging have been priced by DCA to be about the same (less than 1 percent difference); however, actual costs for converting unclassified nodes to classified are probably underestimated.
4. Packets from SCI and TS communities are not mixed with packets from unclassified users. Also, SCI and TS

packets do not pass through secret-level switches maintained by personnel from other communities.

Following are some considerations on Options 2.1, 2.2, and 2.3.

Option 2.1 (Fig. A-5)

Advantages:

1. This option has less technical risk than the options in Fig. A-6. It should be implementable.
2. This option should cost less than the options in Fig. A-6 from both a developmental and implementation viewpoint.

Disadvantages:

1. The option in Fig. A-5 gives unclassified users access to TS and SCI switches when the switches are closed. If IPLIs are used everywhere it is too expensive; if not, it is very vulnerable.
2. The technical risk is greater than for Option 1.
3. The switches are normally open, so when they are closed the network enters a new "not normal" operating mode. Experience says that such a procedure is dangerous.

Options 2.2 and 2.3 (Fig. A-6)

Advantages:

1. The primary reason for these alternatives is greater survivability. The risk of security attacks must be factored in, however.
2. Unclassified users are pretty well blocked from the C^2I subnetwork by the decryption in the bridging devices. The C^2I users have an extra level of encryption on packets while in the other part of the network.

Disadvantages:

1. The greatest technical risk lies in the two alternatives, since both are very new and in formative stages.
2. The network will be larger than two separate nets, and routing and access control problems might arise.

3. The two sections may lead to problems in network control from the network control centers and in bookkeeping the network.

Conclusion: A reasonable position to take when decisions are made appears to be as follows:

1. Two networks which are separate--a C^2I and an unclassified network--should be developed at this time. The present plans for WIN upgrade, followed by adding DODIIS and stripping out military nodes from ARPANET to use as a backbone for an unclassified network, are currently planned and we agree with this procedure.
2. Support for research and development work on the bridging devices approaches appears reasonable. Network designs and plans to use these approaches should be deferred until more is known and some testing has been performed.

REFERENCES

1. Kent, Stephen D., "A Comparison of Some Aspects of Public Key and Conventional Cryptosystems," in ICC79, Vol. 1, pp. 4.3.1-4.3.5, 5 ref.

2. Kent, Stephen D., "Protocol Design Considerations for Network Security," in Beau 79, pp. 239-259, 15 ref.

3. Matyas, S.M. and C.H. Meyer, "Generation, Distribution, and Installation of Cryptographic Keys," IBM Systems Journal, Vol. 17, No. 2, pp. 126-137, 4 ref., 1978.

4. Walker, E.L., "Cipher Key Management for Data Security Applications," IBM Technical Disclosure Bulletin, Vol. 19, No. 6, pp. 2168-2170, November 1976.

5. Wood, David C., "Local Networking at Mitre/Washington C-cubed Operations," The Mitre Corp., McLean, VA, MTR-81W00178, September 1981.

6. Padlipsky, M.A., D.W. Snow, and P.A. Karger, "Limitations of End-to-End Encryption in Secure Computer Networks," The Mitre Corporation, MTR-3592, Vol. I, Bedford, MA, May 1978.

7. Sidhu, Deepinder P. and M. Gasser, "Design for a Multilevel Secure Local Area Network," The Mitre Corp., November 1981.

8. Holmgren, S.F., A.P. Skelton and D.A. Gomberg, "FY 79 Final Report: Cable Bus Applications in Command Centers," The Mitre Corporation, MTR-79W00383, October 1979.

9. Skelton, A.P., J. Nabielsky and S.F. Holmgren, "FY 81: Cable Bus Applications in Command Centers," The Mitre Corporation, MTR-81W00248-01, December 1981.

10. Bartee, T.C. and O.P. Buneman, "Data Base Access in C^3I Computer Networks," IDA Paper P-1489, Institute for Defense Analyses, June 1980.

11. Metcalf, R.M. and D.R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," Comm. ACM 19, July 1976.

12. Riley, W.B., "LAN Standards Controversy .....," Electronic Design, September 1981.

13. Clark, D.D., K.T. Pogran and D.P. Reed, "An Introduction to Local Area Networks," Proc. IEEE, 66, November 1978.

14. Tanenbaum, A.S., "Network Protocols," Computing Surveys, 13, 4, December 1981.

15. Bernstein, P.A., D.W. Shipman and J.B. Rothnie, "Concurrency Control in a System for Distributed Databases (SDD-1)," ACM Trans. Database Systems, 5,1, 1980.

16. Gray, J.N., "Notes on Database Operating Systems," in Operating Systems: An Advanced Course, Springer Verlag, New York, 1978.

17. Menasce, D.A. and R.R. Muntz, "Locking and Deadlock Detection in Distributed Databases," IEEE Trans. on Software Engineering, SE-5, 195-202, May 1979.

18. Stonebraker, M., "Concurrency Control .... in INGRES," IEEE Trans. on Software Engineering, SE-5, 195-202, May 1979.

19. Bernstein, P.A., and N. Goodman, "Concurrency Control in Distributed Databases," Computing Surveys, 13, 2, 1981.

20. Germano, F., PhD. Dissertation, University of Pennsylvania, 1978.

21. Maryanski, F., "Data Server Design Issues," Proc. National Computer Conf., Houston, TX, 1982.

22. Codasyl Database Task Group Report, April 1971.

23. Canady, R.E. et al., "A Back-End Computer for Database Management," Comm. ACM, 17, 10, October 1974.

24. Maryanski, F., "Back-End Database Systems," Computing Surveys, Vol. 12, No. 1, March 1980.

25. Erickson, L.E., M.E. Soleglad and S.L. Westmark, "ADAPT I Final Functional Specification and Design," Logicon, Inc., January 1978.

26. Cullinane, J. et al., "Commercial Data Management Processor Study," Cullinane Corp., Wellesley, MA, 1975.

27. Maryanski, F. et al., "A Prototype Distributed DBMS," Hawaii Intl. Conference on System Sciences," January 1979.

28. Hutchison, J.S. and W.G. Roman, "MADMAN Machine," Workshop on Computer Architecture for Non-Numeric Processing, August 1978.

29. Held, G., M.R. Stonebraker and E. Wong, "INGRES: A Relational Database System," Proc. AFIPS NCC, Vol. 44, 1975.

30. ADAPT I Final Functional and System Design Specification, Logicon, Inc., 1978.

31. "Constructing Superviews," Proc. ACM SIGMOD, May 1981.

32. Chamberlin, D.D., and R.F. Boyce," SEQUEL: A Structured English Query Language," Proc. ACM SIGMOD, 1974.

33. Zloof, M.M., "Query By Example," Proc. NCC 44, May 1975.

34. Harvest Reference Manual, International Database Systems, Philadelphia, PA, 1978.

35. Wasserman, T., "The Data Management Facilities of Plain," Proc. ACM SIGMOD Conf., 1979.

36. Rowe, L., "User Views in Rigel," Proc. ACM SIGMOD Conf., 1979.

37. Mylopoulos, J., P.A. Bernstein, and H.K.T. Wong, "A Language Facility for Designing Database-Intensive Applications," ACM Trans. Database Systems, 5, 2, 1980.

38. Schmidt, J.W., "Some High-Level Language Constructs for Data Type Relation," ACM Trans. Database Systems, 2, 3, 1977.

39. ADAPT I Trial Functional and System Design Specification, Logicon, Inc., 1981.

40. Shipman, D.W., "The Functional Data Model and the Data Language DAPLEX," ACM TODS, 6:1, 1981.

41. Buneman, O.P., R.E. Frankel and R. Nikhil, "An Implementation Technique for Database Query Languages," ACM TODS, June 1982.

REFERENCES FOR APPENDIX

A-1. Latham, Donald C., "Defense Data Network--Security Architecture Options," memorandum, 10 May 1982.

A-2. National Bureau of Standards, "Data Encryption Standards," Federal Information Processing Standards Publication 46, U.S. Dept. of Commerce, January 1977.

A-3. National Bureau of Standards, "Data Encryption Standard Modes of Operation," Federal Information Processing Standards Publication 81, U.S. Dept. of Commerce, December 1980.

A-4. "Interoperability and Security Requirements for Use of the Data Encryption Standard in the Physical and Data Link Layers of Data Communications," proposed Federal Standard 1026, 21 January 1982.

A-5. General Security Requirements for Equipment Using the Data Encryption Standard, Federal Standard 1027, 14 April 1982.