ULTIMATE CORPORATION

ULTINET

IMPLEMENTATION PLAN

FEBRUARY 10, 1983
REVISION: ORIGINAL

ULTIMATE CORPORATION
ULTINET
IMPLEMENTATION PLAN

1. INTRODUCTION

1.1 OVERVIEW

THIS IMPLEMENTATION PLAN ASSUMES THAT THE CONCEPTS DESCRIBED IN THE ARCHITECTURAL DEFINITION PROVIDE THE BASIS FOR THE IMPLEMENTATION OF ULTINET. A PHASED APPROACH IS TAKEN SUCH THAT THE PROJECT MAY BE CONSIDERED COMPLETE AT THE CONCLUSION OF ANY OF THE PHASES OF DEVELOPMENT. EACH PHASE RESULTS IN A MARKETABLE PRODUCT. A LEVEL OF EFFORT HAS BEEN ASSIGNED TO EACH OF THE TASKS SO THAT ADDITIONAL MANPOWER REQUIREMENTS MAY BE ASSESSED EARLY AND NECESSARY STEPS BE TAKEN TO OBTAIN RESOURCES WHEN THEY ARE REQUIRED. NO ATTEMPT HAS BEEN MADE TO SCHEDULE THE PHASES AT THIS TIME. SCHEDULES WILL BE DEFINED AND ATTACHED TO THIS PLAN.

1.2 PHASES OF IMPLEMENTATION

PHASES OF IMPLEMENTATION HAVE BEEN DEFINED IN ORDER TO PROVIDE AN ORDERLY IMPLEMENTATION OF THE ULTINET PRODUCT. THE INTENT IS TO DEVELOP ALL THREE PHASES, HOWEVER, SINCE PRODUCT WILL FLOW FROM EACH OF THE PHASES, IT MAY BE POSSIBLE TO TERMINATE THE PROJECT AFTER ANY OF THE PHASES.

1.2.1 PHASE ONE OVERVIEW

PHASE ONE OF THE PROJECT IS INTENDED TO PROVIDE THE FACILITIES NECESSARY TO SUPPORT FILE TRANSFER BETWEEN ULTIMATE SYSTEMS. THE TRANSPORT MEDIA SUPPORTED WILL BE AN INTERFACE TO A PUBLIC DATA NETWORK (TELENET) AND, POSSIBLY, IN A LOCAL AREA NETWORK ENVIRONMENT.

THIS PHASE PROVIDES FOR THE EVALUATION OF THIRD PARTY SUPPLIED INTERFACE DEVICES TO TELENET AND LOCAL AREA NETWORKS. NO ATTEMPT WILL BE MADE TO IMPLEMENT THIS CAPABILITY DURING PHASE ONE. AT THE CONCLUSION OF THIS EVALUATION A DECISION WILL BE MADE TO EITHER BUY THE DEVICES FOR RESALE, OR TO RECOMMEND THAT THE DEALERS PURCHASE THE EQUIPMENT FOR INTEGRATION AS NEEDED. THIS IS A MARKETING DECISION, AND ENGINEERING WILL PROVIDE THE TECHNICAL AND COST INFORMATION NECESSARY TO MAKE THE DECISION. VENDOR SELECTION WILL OCCUR DURING THE FIRST PART OF THE PROJECT.

AT THE CONCLUSION OF THIS PHASE, IT WILL BE POSSIBLE FOR USERS OF ULTIMATE SYSTEMS TO TRANSPARENTLY ACCESS REMOTE FILES FROM TCL, BASIC, AND RECALL, BY USING TELENET AND, POSSIBLY, BY USING A LOCAL AREA NETWORK.

1.2.2 PHASE TWO OVERVIEW

PHASE TWO PROJECT IMPLEMENTATION WILL PROVIDE FOR DIRECT ULTIMATE TO ULTIMATE COMMUNICATIONS USING THE SWITCHED TELEPHONE NETWORK, LEASED LINES, PUBLIC DATA NETWORK (TELENET), AND, POSSIBLY, A LOCAL AREA NETWORK. IN ADDITION, REMOTE LOGON FACILITIES WILL BE PROVIDED, ENABLING THE USER TO LOGON

TO ANY OTHER PROCESSOR IN THE NETWORK.

THE TRANSPORT AND SESSION LAYER PROTOCOLS OF ISO WILL BE IMPLEMENTED DURING THIS PHASE, PROVIDING END TO END ASSURANCE OF DELIVERY OF DATA AND MULTIPLE SESSION SYNCHRONIZATION WITHIN EACH NETWORK NODE. IN ADDITION, RESOURCE MANAGEMENT WILL BE PROVIDED, ALLOWING USERS TO DIRECT OUTPUT TO DEVICES ATTACHED ANY WHERE IN THE NETWORK. THIS FACILITY WILL ALSO PROVIDE A BASIC ELECTRONIC MAIL CAPABILITY WITHIN THE NETWORK.

NETWORK LEVEL DIAGNOSTICS AND MANAGEMENT WILL ALSO BE PROVIDED AS A PART OF THIS PHASE. INTELLIGENT CONTROLLER ALTERNATIVES WILL BE INVESTIGATED AND DEFINED. THIS PHASE OF THE PROJECT WILL REQUIRE THESE CONTROLLERS IN ORDER TO PROVIDE THE THROUGHPUT THAT IS REQUIRED IN AN INTERACTIVE ENVIRONMENT.

THIS PHASE WILL RESULT IN THE PROVISION OF THE CAPABILITIES DEFINED ABOVE, AS WELL AS THOSE PROVIDED IN PHASE ONE. AT THE CONCLUSION OF THIS PHASE OF IMPLEMENTATION FULL NETWORKING CAPABILITY WILL EXIST FOR ULTIMATE PRODUCTS USING THE PHASE TWO CONTROLLERS FOR DIRECT COMMUNICATIONS AND THE PHASE ONE THIRD PARTY INTERFACES FOR TELENET AND LOCAL COMMUNICATIONS.

1.2.3 PHASE THREE OVERVIEW

PHASE THREE IMPLEMENTATION IS INTENDED TO PROVIDE FULL INTEROPERABILITY WITH COMPATIBLE ARCHITECTURES AND IMPLEMENTATIONS OF THE FULL SET OF ISO PROTOCOLS FOR OPEN SYSTEMS INTERCONNECTION. IN ADDITION, CONTROLLER FIRMWARE WILL BE DEVELOPED TO REPLACE THE THIRD PARTY TELENET INTERFACE. A LOCAL AREA NETWORK CONTROLLER AND FIRMWARE COMPATIBLE WITH THE IEEE 802 STANDARD FOR LOCAL AREA NETWORKS WILL BE IMPLEMENTED, REPLACING THE THIRD PARTY LAN INTERFACE.

THE DECISION TO PROCEDE WITH PHASE THREE WILL BE MADE AT THE CONCLUSION OF PHASE TWO. THIS DECISION IS A MARKETING DECISION AND WILL BE BASED ON THE SUCCESS OF THE THIRD PARTY PROVIDED INTERFACES, THE COST OF IMPLEMENTATION OF REPLACEMENT PRODUCT, AND THE DESIRE TO INTERCONNECT IN A FULL OPEN SYSTEM ENVIRONMENT. ANY PART OF PHASE THREE MAY BE A CANDIDATE FOR IMPLEMENTATION, SINCE THEY ARE LOGICALLY AND TECHNICALLY INDEPENDENT.

2. DETAILED DESCRIPTION OF THE PHASES

2.1 PHASE ONE IMPLEMENTATION

EACH TASK IS BRIEFLY DESCRIBED, AND AN ESTIMATE OF THE EFFORT REQUIRED TO PERFORM THE TASK IS GIVEN.

2.1.1 CHANGES TO EXISTING SYSTEM ROUTINES

THE FOLLOWING SYSTEM ROUTINES MUST BE MODIFIED TO RECOGNIZE THAT THE REQUESTED FILE IS REMOTE. IN ADDITION, THE 'OPEN' ROUTINE MUST ESTABLISH A LOGICAL CONNECTION IF ONE DOES NOT ALREADY EXIST. EACH ROUTINE WILL INTERFACE TO A VIRTUAL

PROGRAM WHICH EMULATES THE PRESENCE OF A TRUE SESSION LAYER PROTOCOL MACHINE. THIS WILL ENABLE THE ADDITION OF A SESSION LAYER WITH MINIMAL IMPACT ON THE CURRENTLY PLANNED MODIFICATIONS.

ROUTINE	PURPOSE
OPEN	RETRIEVE FILE POINTERS ESTABLISH LOGICAL CONNECTION
RETIX	RETRIEVE ITEM POINTERS
RETIXU	RETRIEVE ITEM POINTERS AND GROUP LOCK
RETIXUX	AS ABOVE, EXCEPT RETURN IF GROUP LOCKED
GETITM	BUILD LIST OF ITEM ID IN GROUP, RETURN NEXT SEQUENTIAL ID
UPDITM	BUILD NEW ITEM AND WRITE TO FILE AND CLEAR GROUP LOCK
GLOCK	PERFORMS GROUP LOCK
GUNLOCK	PERFORMS GROUP UNLOCK

ESTIMATED EFFORT TO COMPLETE THE ABOVE TASKS IS 6 MAN MONTHS.

2.1.2 DESIGN/IMPLEMENT NEW SYSTEM ROUTINES

NEW SYSTEM ROUTINES WILL BE NEEDED IN ORDER TO SUPPORT REMOTE FILE ACCESS AND SUPPORT THE LOGICAL RELATIONSHIP BETWEEN PROCESSES ON DIFFERENT NODES. THESE ROUTINES ARE IDENTIFIED BELOW.

ROUTINE	PURPOSE
RETIXI	RETRIEVE ITEM POINTERS AND ITEM LOCK
RETIXIX	AS ABOVE, EXCEPT RETURN IF ITEM LOCKED
ILOCK	PERFORMS ITEM LOCK
IUNLOCK	PERFORMS ITEM UNLOCK
CLOSE	CLOSE ACTIVITY ON REMOTE FILE PROCESSOR
CLOSEC	DISCONNECT SESSION. THIS MAY BE DONE BY USER, OR DURING 'WRAP-UP'.

ESTIMATED EFFORT TO COMPLETE THE ABOVE TASKS IS 6 MAN MONTHS.

2.1.3 REMOTE FILE SERVER

A REMOTE FILE SERVER WILL BE IMPLEMENTED TO PERFORM THOSE TASKS NECESSARY TO EFFECT A REMOTE FILE ACCESS AND TRANSFER. THIS SERVER WILL BE CAPABLE OF MAINTAINING MULTIPLE LOGICAL SESSIONS, WITH MULTIPLE FILE ACCESSES PER SESSION. THE EXACT NUMBER OF OPEN SESSIONS, AND OPEN FILES IS TO BE DEFINED. A REMOTE FILE SERVER WILL BE ACTIVE IN EACH NODE OF THE NETWORK AND WILL OPERATE AS A VIRTUAL PROCESS AT AN ASSIGNED PORT IN A MANNER SIMILAR TO THE CURRENT PRINTER SPOOLER. THIS FUNCTIONAL PROGRAM IS FREQUENTLY REFERRED TO AS A COMM SPOOLER.

ESTIMATED TIME TO COMPLETE THIS TASK IS 12 MAN MONTHS.

2.1.4 EXTERNAL DEVICE DRIVER

AFTER AN EXTERNAL DEVICE TO SUPPORT TELENET ACCESS HAS BEEN IDENTIFIED A SYSTEM LEVEL ROUTINE WILL BE IMPLEMENTED TO INTERFACE THE DEVICE TO THE ULTIMATE SYSTEM. AN EFFORT WILL BE MADE TO IDENTIFY A DEVICE WHICH INTERFACES THROUGH THE

ASYNCHRONOUS PORT OF BOTH THE HONEYWELL AND THE DEC BASED MACHINES, THEREBY MINIMIZING THE EFFORT REQUIRED TO SUPPORT A TELENET INTERFACE.

ESTIMATED EFFORT TO PERFORM THE ABOVE TASK IS 6 MAN MONTHS.

IF IT IS DEEMED DESIREABLE TO SUPPORT A LOCAL AREA NETWORK, AN ADDITIONAL DEVICE INTERFACE DRIVER WILL HAVE TO BE IMPLEMENTED FOR THIS PURPOSE, AGAIN USING THE ASYNCHRONOUS INTERFACE TO MINIMIZE THE EFFORT.

ESTIMATED EFFORT TO PERFORM THE ABOVE TASK IS 6 MAN MONTHS.

2.1.5 NETWORK SIMULATOR

A NETWORK SIMULATOR WILL BE DEFINED AND IMPLEMENTED IN ORDER TO AID THE CHECKOUT PROCESS IN A SINGLE MACHINE. THIS SIMULATOR WILL INTERFACE TO THE SESSION LAYER EMULATOR AND THE REMOTE FILE SERVER. UPON COMPLETION OF CHECKOUT IN THIS MANNER, THE SIMULATOR WILL BE REMOVED FROM THE SYSTEM AND THE EXTERNAL DEVICES AND ASSOCIATED SOFTWARE WILL BE INSERTED TO PERFORM THE FINAL CHECKOUT.

ESTIMATED EFFORT TO PERFORM THIS TASK IS 3 MAN MONTHS.

2.1.6 ACCESS SECURITY

IT MAY BE NECESSARY TO PROVIDE ADDITIONAL ACCESS SECURITY MECHANISMS IN THE SYSTEM TO PREVENT UNAUTHORIZED ACCESS TO USER DATA. THIS BECOMES CRITICAL IN A NETWORKING ENVIRONMENT AND MUST BE CONSIDERED. THE MECHANISMS TO BE DEVELOPED ARE UNDEFINED AT THIS TIME, AND THIS DEVELOPMENT MAY SPAN PHASE ONE AND PHASE TWO OF THE DEVELOPMENT, DEPENDING ON THE COMPLEXITY OF THE ALGORITHM CHOSEN.

2.1.7 QUEUE MANAGEMENT

DURING THE COURSE OF THE DESIGN IT MAY BECOME OBVIOUS THAT A COMMON QUEUE MANAGEMENT FUNCTION IS NEEDED TO SIMPLIFY THE INTERACTION OF VARIOUS SYSTEM AND VIRTUAL PROCESSES. THE FIRST PHASE WILL AVOID THIS CONDITION IF AT ALL POSSIBLE IN ORDER TO REDUCE THE OVERALL EFFORT REQUIRED FOR FIRST PHASE IMPLEMENTATION.

2.1.8 FINAL INTEGRATION AND TEST

AFTER COMPLETION OF LOCAL MODULE DEBUG, A FINAL INTEGRATION AND TEST CYCLE WILL BEGIN, USING THE NETWORK SIMULATOR, AND FINALLY THROUGH TWO TELENET PORTS. TO SIMPLIFY THIS EFFORT, THE TELENET PORTS WILL BOTH BE LOCAL INITIALLY. WHEN THE TEAM IS SATISFIED THAT THE PHASE ONE PROJECT IS WORKING SATISFACTORILY A THIRD PORT WILL BE ADDED FOR NEW JERSEY AND LONG DISTANCE TESTING. DEALERS DESIRING THIS PRODUCT CAPABILITY WILL BE ABLE TO OBTAIN TELENET PORTS AND USE THE TEST SOFTWARE DEVELOPED FOR THIS PRODUCT TO CONDUCT DEMONSTRATIONS WHILE CONNECTED TO NEW JERSEY VIA TELENET.

ESTIMATED EFFORT TO PERFORM THIS TASK IS 8 MAN MONTHS.

2.1.9 TEST EQUIPMENT REQUIREMENTS

IT MAY BE NECESSARY TO PURCHASE A PROGRAMMABLE LINE ANALYSER TO AID THE DEBUG EFFORT. SUCH LINE ANALYSERS TYPICALLY COST IN THE NEIGHBORHOOD OF $20,000. IT WILL CERTAINLY BE REQUIRED FOR THE NEXT PHASE OF DEVELOPMENT.

2.1.10 SUMMARY OF DEVELOPMENT EFFORT

THE DEVELOPMENT EFFORT FOR THE ABOVE TASKS IS SUMMARIZED BELOW:

TASK ITEM	EFFORT	
2.1.1	6 MM	
2.1.2	6 MM	
2.1.3	12 MM	
2.1.4	6 MM	(X.25)
	6 MM	(LAN)
2.1.5	3 MM	
2.1.8	8 MM	

TOTAL (X.25 OR LAN): 41 MM
TOTAL (X.25 & LAN): 47 MM

THE IMPACT OF DOING 2.1.6 OR 2.1.7 IS UNKNOWN AT THIS TIME.

2.2 PHASE TWO IMPLEMENTATION

THE TASKS IDENTIFIED FOR IMPLEMENTATION DURING PHASE TWO ARE DESCRIBED BELOW. THE ESTIMATED LEVEL OF EFFORT FOR EACH TASK IS ALSO IDENTIFIED.

2.2.1 SYMETRICAL HDLC

THE SYMETRICAL VERSION OF HDLC (CURRENTLY THE SUBJECT OF STANDARDIZATION WITHIN ISO) WILL BE DEVELOPED TO PERFORM THE LINK LEVEL FUNCTIONS OF THE DATA LINK LAYER OF THE ULTINET ARCHITECTURE.

ESTIMATED EFFORT FOR THIS TASK IS 12 MAN MONTHS.

2.2.2 X.25 PACKET LEVEL

THE DTE TO DTE VERSION OF THE X.25 PACKET LEVEL (CURRENTLY THE SUBJECT OF STANDARDIZATION WITHIN ISO) WILL BE DEVELOPED TO PERFORM THE NETWORK LAYER FUNCTIONS OF THE ULTINET ARCHITECTURE.

ESTIMATED EFFORT FOR THIS TASK IS 12 MAN MONTHS.

2.2.3 TRANSPORT PROTOCOL

THE CLASS 2 (POSSIBLY CLASS 4) ISO TRANSPORT PROTOCOL WILL BE DEVELOPED TO PERFORM THE TRANSPORT LAYER FUNCTIONS OF THE ULTINET ARCHITECTURE.

ESTIMATED EFFORT FOR THIS TASK IS 15 MAN MONTHS

2.2.4 SESSION PROTOCOL

THE MINIMAL CONFORMANCE SUBSET OF THE ISO SESSION PROTOCOL WILL BE DEVELOPED TO PERFORM THE SESSION LAYER FUNCTIONS OF THE ULTINET ARCHITECTURE.

ESTIMATED EFFORT FOR THIS TASK IS 15 MAN MONTHS.

2.2.5 REMOTE LOGON

A REMOTE LOGON FACILITY WILL BE PROVIDED TO ENABLE A USER TO LOGON AND PERFORM TASKS ON A REMOTE COMPUTER SYSTEM. THE USER WILL BE UNAWARE THAT HE IS LOGGED TO A DIFFERENT SYSTEM OTHER THAN BY OBSERVING AN AS YET UNDEFINED PERFORMANCE DEGRADATION.

ESTIMATED EFFORT FOR THIS TASK IS 9 MAN MONTHS.

2.2.6 RESOURCE MANAGER

THE ADDITION OF THIS FACILITY WILL ALLOW THE USER TO SPECIFY A LOGICAL OUTPUT DEVICE, SUCH AS 'INVOICE PRINTER', AND HAVE THE SYSTEM ROUTE HIS OUTPUT TO THE SPECIFIED DEVICE, REGARDLESS OF ITS LOCATION WITHIN THE NETWORK. A LOGICAL EXTENSION OF THIS CAPABILITY WILL SUPPORT ELECTRONIC MAIL WITHIN THE NETWORK.

ESTIMATED EFFORT FOR THIS TASK IS 9 MAN MONTHS.

2.2.7 COMMON BUFFER QUEUE MANAGER

DUE TO THE COMPLEXITY BEING INTRODUCED DURING THIS PHASE, AND THE ANTICIPATED INTERACTION AMONG TASKS, IT IS ANTICIPATED THAT A COMMON BUFFER/QUEUE MANAGEMENT STRUCTURE WILL BE REQUIRED.

ESTIMATED EFFORT FOR THIS TASK IS 9 MAN MONTHS.

2.2.8 NETWORK DIAGNOSTICS

IT WILL BE NECESSARY TO DEVELOP A COMPREHENSIVE PACKAGE OF SYSTEM AND NETWORK DIAGNOSTICS TO ENABLE THE FIELD SUPPORT PERSONNEL TO ISOLATE AND CORRECT PROBLEMS RELATED TO THE INTERCONNECTION OF MULTIPLE PROCESSORS. IN ADDITION, TELEPHONE COMPANY RELATED PROBLEMS MUST BE EASILY IDENTIFIABLE IN ORDER TO MINIMIZE NETWORK DOWNTIME.

ESTIMATED EFFORT FOR THIS TASK IS 15 MAN MONTHS.

2.2.9 NETWORK MANAGEMENT

NETWORK MASNAGEMENT TOOLS MUST BE PROVIDED TO ALLOW THE USER TO CONFIGURE NODES INTO AND OUT OF THE NETWORK IN AN ORDERLY MANNER. THE ABILITY TO CHANGE BASIC ROUTING TABLES DYNAMICALLY MUST BE PROVIDED IN ORDER TO COMPENSATE FOR LOST COMMUNICATIONS ON CONFIGURED LINKS. STATISTICAL DATA ON THE UTILIZATION OF THE NETWORK MUST BE MADE AVAILABLE FOR ACCOUNTING AND TOPOLOGY MANAGEMENT PURPOSES. THE INITIAL EFFORT IN THIS AREA MAY BE LIMITED IN ORDER TO PROVIDE A MINIMAL SUBSET OF THIS CAPABILITY THAT IS USEFUL TO THE USER.

ESTIMATED EFFORT FOR THIS TASK IS 15 MAN MONTHS.

2.2.10 INTELLIGENT CONTROLLERS

THE NEED FOR INTELLIGENT CONTROLLERS FOR THIS PHASE OF DEVELOPMENT HAS BEEN IDENTIFIED. THE INTENT IS TO USE CONTROLLERS AVAILABLE FROM HONEYWELL AND DEC FOR THIS PURPOSE. IT MAY BE NECESSARY TO DEFINE NEW CONTROLLERS IF THE EXISTING PRODUCT WILL NOT MEET THE REQUIREMENTS OF THE NETWORK. DISCUSSIONS WITH HONEYWELL ON THIS SUBJECT NEED TO OCCUR PRIOR TO THE BEGINING OF THIS PHASE OF DEVELOPMENT.

THIS DEVELOPMENT WILL BE PERFORMED EXTERNAL TO ULTIMATE.

2.2.11 TEST AND INTEGRATION

FINAL TEST AND INTEGRATION OF THE DEVELOPMENT TASKS IN THIS PHASE WILL BE PERFORMED AT THE CONCLUSION OF THE DEVELOPMENT EFFORT.

ESTIMATED EFFORT FOR THIS TASK IS 12 MAN MONTHS.

2.2.12 TEST EQUIPMENT

A PROGRAMMABLE LINE ANALYSER WILL BE REQUIRED DURING THE DEBUG PHASE OF THIS PROJECT. TYPICAL PRODUCT COSTS ARE IN THE NEIGHBORHOOD OF $20,000. THIS DEVICE MAY HAVE BEEN PURCHASED DURING PHASE ONE DEVELOPMENT.

2.2.13 SUMMARY OF DEVELOPMENT EFFORT

THE EFFORT REQUIRED TO IMPLEMEWNT PHASE TWO IS IDENTIFIED BELOW BY ITS ASSOCIATED TASK.

TASK	EFFORT
2.2.1	12 MM
2.2.2	12 MM
2.2.3	15 MM
2.2.4	15 MM
2.2.5	9 MM
2.2.6	9 MM
2.2.7	9 MM
2.2.8	15 MM
2.2.9	15 MM
2.2.11	12 MM

TOTAL EFFORT: 123 MM

2.3 PHASE THREE IMPLEMENTATION

THE TASKS IDENTIFIED FOR PHASE THREE IMPLEMENTATION ARE IDENTIFIED BLEOW. THE LEVEL OF EFFORT TO PERFORM EACH OF THE TASK IS ALSO GIVEN.

2.3.1 X.25 LINK AND PACKET (DTE TO DCE)

THE PHASE TWO X.25 PRODUCT WILL BE MODIFIED TO SUPPORT THE DTE TO DCE INTERFACE IN ORDER TO REPLACE THE THIRD PARTY SUPPLIED INTERFACE DEVICE. THIS TASK SHOULD BE UNDERTAKEN

ONLY IF SALES OF THE PHASE ONE PRODUCT JUSTIFY ELIMINATING THE THIRD PARTY DEVICE.

ESTIMATED EFFORT TO PERFORM THIS TASK IS 12 MAN MONTHS.

2.3.2 LOCAL AREA NETWORK

THIS TASK WILL RESULT IN A NEW CONTROLLER/FIRMWARE SET TO REPLACE THE THIRD PARTY SUPPLIED DEVICE DEVELOPED IN PHASE ONE. THIS TASK SHOULD ONLY BE UNDERTAKEN IF THE SALES OF THE PHASE ONE PRODUCT JUSTIFY THE DEVELOPMENT OF A REPLACEMENT DEVICE.

ESTIMATED EFFORT TO PERFORM THIS TASK IS 24 MAN MONTHS

OUTSIDE DEVELOPMENT WILL BE REQUIRED TO PRODUCE THE NECESSARY CONTROLLERS AND INTERFACE DEVICES.

IT IS RECOMMENDED THAT THIS PRODUCT BE DEVELOPED TO THE IEEE 802 STANDARD USING CSMA/CD FIBER OPTIC TECHNOLOGY.

2.3.3 PRESENTATION LAYER PROTOCOLS

THE ISO PRESENTATION LAYER PROTOCOLS LISTED WILL BE DEVELOPED IN ORDER TO PROVIDE FULL INTEROPERABILITY WITHIN THE OPEN SYSTEMS ENVIRONMENT.

VIRTUAL FILE: MASK THE DIFFERENCES BETWEEN VARIOUS FILE STRUCTURES.

VIRTUAL TERMINAL: MASK THE DIFFERENCES BETWEEN VARIOUS TERMINAL DEVICES.

JOB TRANSFER AND MANIPULATION: PROVIDE THE PROTOCOL NECESSARY TO MOVE OBJECT CODE BETWEEN OPEN SYSTEMS AND CAUSE THEIR EXECUTION ON REMOTE PROCESSORS.

ESTIMATED EFFORT TO PERFORM THIS TASK IS 40 MAN MONTHS

2.3.4 COMMON APPLICATION LAYER PROTOCOLS

THE ISO COMMON APPLICATION PROTOCOL WILL BE DEVELOPED TO PROVIDE FOR A COMMON APPLICATION INTERFACE WITHIN THE OPEN SYSTEM INTERCONNECTION ENVIRONMENT. THIS PROTOCOL WILL ENABLE VARIOUS APPLICATIONS ON DISSIMILAR PROCESSORS TO COMMUNICATE USING A STANDARD INTERFACE PROTOCOL.

ESTIMATED EFFORT TO PERFORM THIS TASK IS 15 MAN MONTHS.

2.3.5 INTEGRATION AND TEST

THE FULL INTEGRATION AND TEST OF THE PHASE THREE PROJECT WILL BE PERFORMED UNDER THIS TASK. THE ACTUAL AMOUNT OF EFFORT REQUIRED TO PERFORM THIS TASK IS DEPENDENT UPON THE NUMBER OF THE TASKS DEFINED ABOVE THAT ARE IMPLEMENTED.

THE EFFORT REQUIRED TO PERFORM THIS TASK WILL VARY BETWEEN 4 AND 15 MAN MONTHS.

2.3.6 SUMMARY OF DEVELOPMENT EFFORT

THE PHASE THREE DEVELOPMENT EFFORT IS SUMMARIZED BELOW BY TASK IDENTIFICATION:

TASK	EFFORT
2.3.1	12 MM
2.3.2	24 MM
2.3.3	40 MM
2.3.4	15 MM
2.3.5	4 MM TO 15 MM
TOTAL EFFORT	95 MM TO 106 MM

3. MANPOWER REQUIREMENTS

THE PHASE ONE PROJECT WILL REQUIRE FOUR PROGRAMMERS ON STAFF IN ORDER TO COMPLETE DURING CALENDAR YEAR 1983. ONE ADDITIONAL PROGRAMMER SHOULD BE ADDED TO THE CURRENT STAFF AS SOON AS POSSIBLE.

THE PHASE TWO PROJECT WILL REQUIRE EIGHT PROGRAMMERS ON STAFF IN ORDER TO COMPLETE DURING CALENDAR YEAR 1984. FOUR ADDITIONAL PROGRAMMERS SHOULD BE ADDED TO THE STAFF BY JUNE, 1983.

THE PHASE THREE PROJECT CAN BE COMPLETED DURING CALENDAR YEAR 1985 WITHOUT ADDING TO THE STAFF.

3.1 SUMMARY

REQUIRED	NUMBER	EXPERIENCE
NOW	1	PREFER PICK OPERATING SYSTEM
JUNE, 1983	4	COMMUNICATIONS PROTOCOLS
TOTAL	5	

4. RECOMMENDATION

IT IS RECOMMENDED THAT PHASE ONE AND PHASE TWO DEVELOPMENT BE REVIEWED AND APPROVED AS PROPPOSED, AND THAT STAFFING BEGIN AS SOON AS POSSIBLE.

IT IS RECOMMENDED THAT THE DECISION ON THE COMPONENTS REQUIRED FOR PHASE THREE DEVELOPMENT BE IDENTIFIED AND SCHEDULED NO EARLIER THAT SEPTEMBER, 1984.

3/18/83

PROPOSED

DEALER QUESTIONAIRE

DRAFT C

THE ULTIMATE CORPORATION

DATA COMMUNICATIONS

THE ULTIMATE CORPORATION IS CURRENTLY DEVELOPING A RANGE OF SOPHISTICATED SYNCHRONOUS COMMUNICATIONS PRODUCTS. THE PURPOSE OF THIS QUESTIONAIRE IS TO DETERMINE DEALER REQUIREMENTS AS ADDITIONAL INPUT DURING THE EARLY STAGES OF THIS DEVELOPMENT.

THANK YOU FOR TAKING THE TIME TO COMPLETE THIS QUESTIONAIRE. YOUR ANSWERS WILL BE HELD IN CONFIDENCE AND WILL BE USED BY ULTIMATE TO ASSIST THE PLANNING EFFORT RELATED TO PROVIDING YOU WITH A VARIETY OF DATA COMMUNICATIONS PRODUCTS.

YOUR PROMPT RETURN OF THIS QUESTIONAIRE IN THE RETURN ENVELOPE YOU ENSURE THAT YOUR REQUIREMENTS WILL BE CONSIDERED.

DEALER NAME:_____

CONTACT:_____

PHONE:_____

PART ONE - CURRENT BUSINESS

PLEASE INDICATE QUANTITIES OF ULTIMATE SYSTEMS IN THE APPROPRIATE COLUMN.

	HONEYWELL	DEC
1. NUMBER OF STAND-ALONE MACHINES SOLD	_____/YR	_____/YR
2. NUMBER OF MULTIPLE-MACHINE INSTALLATIONS SOLD	_____/YR	_____/YR
A. MULTIPLE SYSTEMS IN ONE LOCATION	_____	_____
B. MULTIPLE SYSTEMS IN MORE THAN ONE LOCATION	_____	_____
3. CUSTOMERS YOU EXPECT TO ADD ONE OR MORE SYSTEMS IN ONE LOCATION	_____	_____
4. CUSTOMERS YOU EXPECT TO ADD ONE OR MORE SYSTEMS IN MORE THAN ONE LOCATION	_____	_____
5. CUSTOMER INQUIRIES ABOUT DATA COMMUNICATIONS	_____	_____

6. PLEASE INDICATE TYPES OF SOFTWARE PACKAGES SOLD.

_____: _____

_____: _____

_____: _____

_____: _____

7. PLEASE INDICATE SOURCE OF YOUR SOFTWARE PACKAGES.

A. IN HOUSE DEVELOPMENT

B. SEPARATE SOFTWARE COMPANY

C. OPEN MARKET

D. OTHER: _____

PART TWO - CUSTOMER PROSPECT LIST

PLEASE INDICATE QUANTITIES OF ULTIMATE SYSTEMS IN APPROPRIATE COLUMNS.

	HONEYWELL	DEC
1. PROSPECT DESIRING MULTIPLE		
A. SYSTEMS IN ONE LOCATION	_____/YR	_____/YR
B. SYSTEMS IN MORE THAN ONE LOCATION	_____/YR	_____/YR
2. PROSPECTS WHO HAVE INQUIRED ABOUT		
A. LOCAL AREA NETWORK	_____/YR	_____/YR
B. ULTIMATE TO ULTIMATE	_____/YR	_____/YR
C. X.25 TELENET	_____/YR	_____/YR
D. 3270	_____/YR	_____/YR
E. SDLC	_____/YR	_____/YR
F. SNA	_____/YR	_____/YR
3. SYSTEM SALES LOST DUE TO LACK OF		
A. LOCAL AREA NETWORK	_____/YR	_____/YR
B. ULTIMATE TO ULTIMATE	_____/YR	_____/YR
C. X.25 TELENET	_____/YR	_____/YR
D. 3270	_____/YR	_____/YR
E. SDLC	_____/YR	_____/YR

PART THREE - PROJECTIONS AND REQUIREMENTS

1. IF YOU HAD AN ON-LINE, INTERACTIVE REMOTE FILE ACCESS CAPABILITY, HOW MANY SYSTEMS COULD YOU SELL

	HONEYWELL	DEC
A. IN A LOCAL AREA NETWORK CONFIGURATION	_____	_____
B. ULTIMATE TO ULTIMATE (SWITCHED TELEPHONE)	_____	_____
C. X.25 TELENET	_____	_____

2. INDICATE YOUR RELATIVE PRIORITY FOR DEVELOPMENT OF THE FOLLOWING NETWORK CONFIGURATIONS

_____ LOCAL AREA NETWORK

_____ ULTIMATE TO ULTIMATE (SWITCHED TELEPHONE)

_____ X.25 TELENET

_____ ULTIMATE TO NON-ULTIMATE

_____ OTHER

3. INDICATE YOUR RELATIVE PRIORITY FOR DEVELOPMENT OF THE FOLLOWING NETWORK CAPABILITIES

_____ REMOTE FILE ACCESS AND TRANSFER

_____ REMOTE LOGON

_____ REMOTE PRINTER/TAPE ACCESS

_____ REMOTE JOB EXECUTION

_____ OTHER

4. DO YOU DESIRE MORE INFORMATION ON

_____ DATA COMMUNICATIONS IN GENERAL

_____ LOCAL AREA NETWORKS

_____ X.25 TELENET

_____ OTHER

5. WOULD YOU BE INTERESTED IN

_____ SEMINAR ON DATA COMMUNICATIONS

_____ MARKETING/SALES SUPPORT FOR DATA COMMUNICATIONS

6. WILL NETWORKING GIVE YOU AN EDGE OVER THE COMPETITION?

---

7. OTHER COMMENTS/SUGGESTIONS

---

8. DO YOU ANTICIPATE YOUR SALES TO INCREASE WHEN NETWORKING CAPABILITIES ARE AVAILABLE ON ULTIMATE SYSTEMS?

$_____ _____%

NUMBER OF SYSTEMS _____

THANKYOU FOR TAKING YOUR TIME TO PROVIDE US WITH THIS INFORMATION.

Read 2-17-83

ARCHITECTURAL DEFINITION

ULTINET

THE ULTIMATE CORPORATION

FEBRUARY 1983

REVISION: ORIGINAL

ARCHITECTURAL DEFINITION

TABLE OF CONTENTS

ARCHITECTURAL DEFINITION

ULTINET

0. INTRODUCTION

Ultinet is a proprietary network architecture under development by The Ultimate Corporation. At the completion of a phased implementation, this product will provide full networking for the complete line of Ultimate products. The flexibility and adaptability of the architectural design must be such that future products, not currently known or planned, must be able to interoperate within the network environment with existing products.

The intent of this document is to define an architecture which will satisfy this basic requirement. Underlying assumptions will be stated, along with a complete definition of the functionality required of each of the layered components of the architecture. Finally, a phased implementation plan will be presented which will allow for product introduction in a timely and logical manner. This will be the subject of a separate document.

It is further intended that this document be a living document. Technological advances within the data communications industry will be evaluated as they occur, therefore, this definition may be subject to change where such advances are considered appropriate for inclusion in the final product.

Finally, it is intended that this definition be given widespread visibility within The Ultimate Corporation in order to achieve a consensus on the proposed architecture, implementation plan, and final product goals. From this consensus will follow a Corporate and individual commitment to the success of the implementation of this definition.

1. Assumptions

Inherent in this definition are certain underlying assumptions which are stated below.

a) Unless unknown inefficiencies are determined to exist, Ultinet will conform to the layered architecture defined in ISO DIS7498, Basic Reference model for Open Systems Interconnection.

b) The family of standards being developed by ISO related to the Basic Reference model will be used as a basis for implementation of Ultinet.

c) Conformance to the ISO standards will be maintained at a level consistent with the performance goals of the end product. That is, a subset of the standard may be used for Ultimate communications with possible "gateway" functionality provided to the full ISO standard.

d) Necessary hardware components will be developed to support the architecture. This may include one or more intelligent communications controllers.

e) A phased approach is necessary in order to provide revenue generating product as soon as possible. Compatibility will be maintained in order to preclude the necessity of rewriting user provided applications.

f) Resources are and will continue to be made available during the lifetime of the development project.

g) The project may be stopped at the completion of any phase of development without negating any of the previous work. Each phase will result in increasing capability and functionality towards achieving full implementation of Ultinet.

2. Ultinet Goal

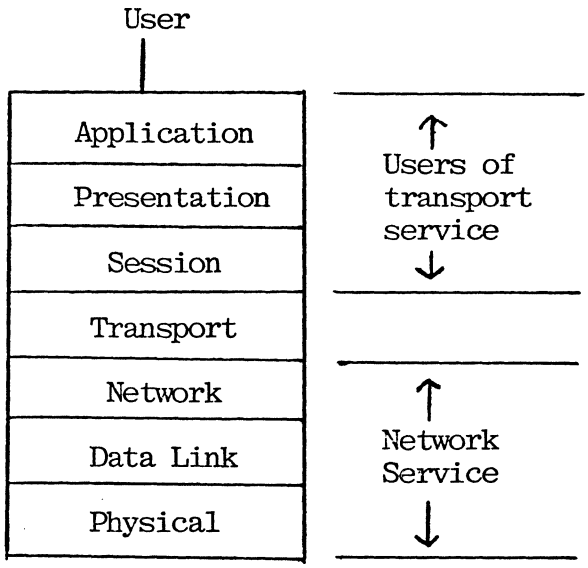
The underlying goal that is to be achieved by the Ultinet product is a cost effective means by which the user of the complete Ultimate product family may transparently access resources, including data, processes, and devices, which are made available through inclusion in the Network. This goal is to be achieved in a timely manner, and in such a way as to provide increasingly functional product to the Ultimate market.

3. The Layered Architecture

The Business Reference Model for Open Systems Interconnection, ISO DIS7498, has evolved from a perceived need by providers of data communiction product to define a framework for the design of protocols for the new generation of distributed information systems. The reference model has undergone from revisions within ISO and is expected to be approved as an International Standard by mid 1983. Close collaboration on the development of this standard has occurred between ISO and the CCITT. This collaboration is continuing in the development of the related service and protocol standards for each of the layers of the architecture. The resultant standards will be compatible between the standards bodies worldwide.

The technique of layering has been employed to provide a logical grouping of functions required to perform the communications task. The lack of such an approach would result in very complex protocols and program structures. The principle of layering will allow the implementation to evolve with little or no impact on the previous work. The initial absense of layers, and subsequent inclusion, will make no visible impact on the user of Ultinet.

The layers of the architecture and a brief description of each are provided below:



Application:

Directly serves the user by providing access to the communication environment

Presentation:

Allows the user to interpret the meaning of the data transferred.

Session:

Manages the dialog between cooperating users.

Transport:

Provides end-to-end reliable transfer of data between end users

Network:

Provides the routing and switching functions needed to support a logical connection between two end systems.

Data link:

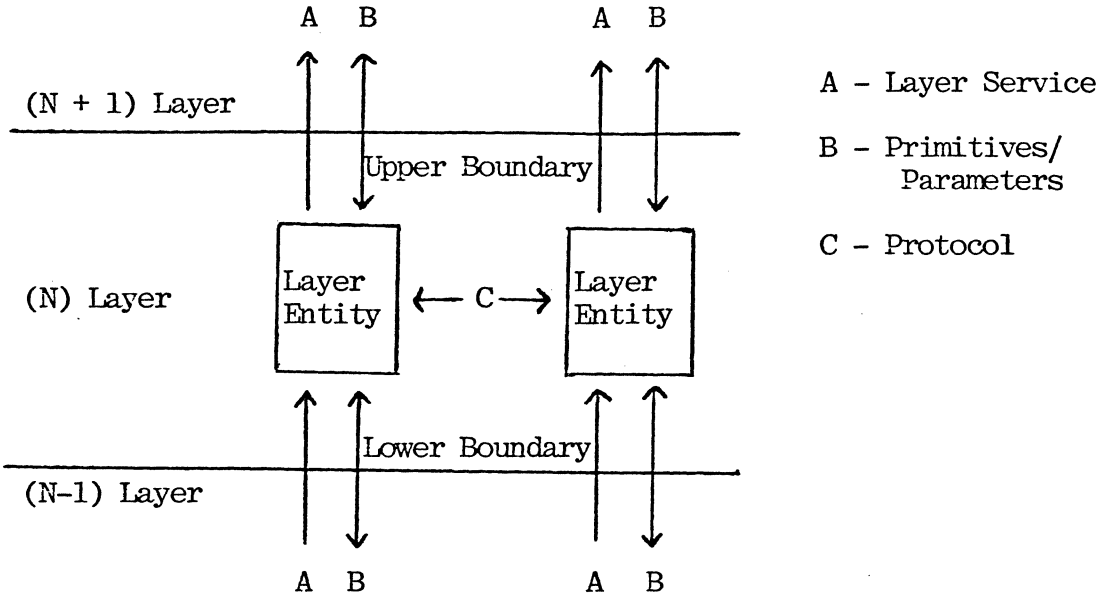
Provides for transfer of data over a physical link with the necessary error recovery and flow control.

Physical:

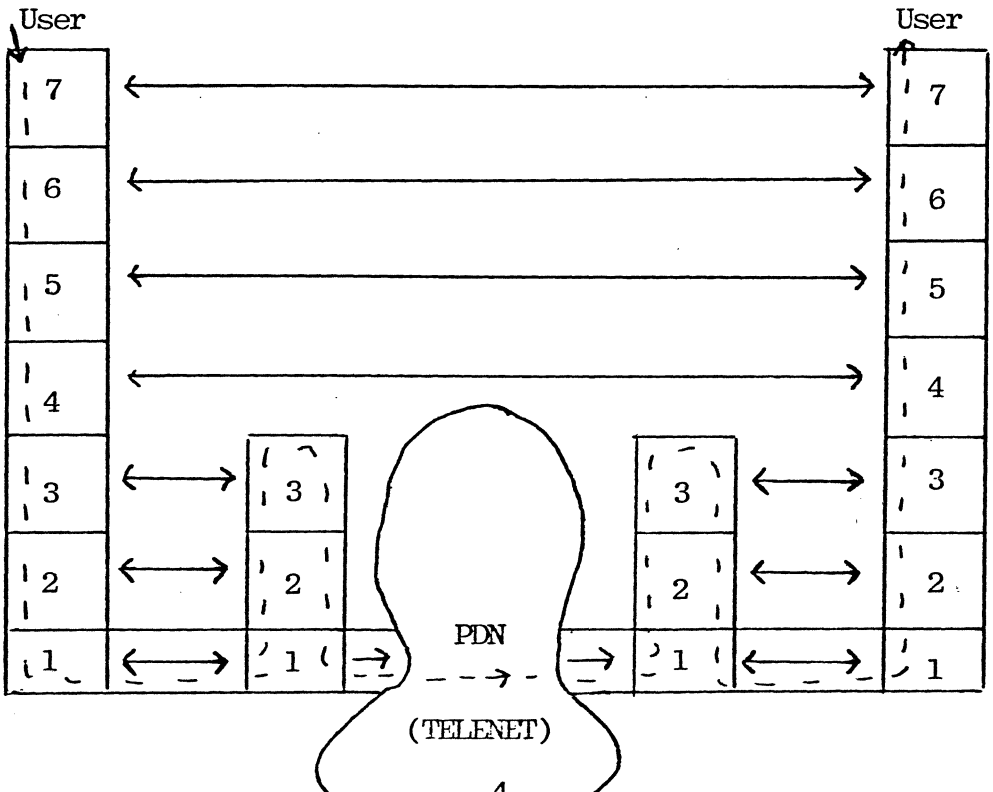
Provides functional and procedural aspects to activate, maintain, and deactivate the physical links.

The upper three layers provide direct support to the user of the communications environment. The functionality provided is the same as that required to perform inter-process communications, whether local or remote. The lower three layers provide the facilities needed to communicate, "network", between end systems, and may roughly be equated to the mailman. The middle layer, transport, provides the necessary interface between the user service and the network provider.

Services provided by a layer are visible to the next higher layer. Layer protocols are "peer" protocols and are not visible outside of the layer. Equivalent layers in cooperating systems communicate with each other using peer protocols. Adjacent layers in an end system communicate using defined service primitives which contain parameters required to perform the requested service. The following diagram graphically depicts this three way relationship between the layers.



The flow of data between end users can be graphically presented by use of the layered architecture. In the following example, two intermediate nodes are shown. In addition, the use of a Public Data Network is also shown. In this example, the nodes adjacent to the Public Data Network are referred to as "gateway" nodes.



4. Functions within the Layers

The layered architecture has been conceptually introduced in the preceeding section. This section will define the functional requirements for each of the layers as they relate to Ultinet. Support systems software will be described in Section 5.

4.1 Physical Layer

The physical layer is comprised of the transmission media and the interface to the system. Bit synchronous, two-way simultaneous protocols are supported in either a four-wire or coax cable environment. Automatic flag generation, zero-bit insertion, and cyclic redundancy check (CRC) is accomplished on the physical interface by use of commercially available chips designed for this purpose. The interface supported will conform to EIA standard RS232C. This interface will reliably support data transmission speeds up to 19.2KB. The voice switched network will support reliable data transfer up to 4800 baud. Conditioned lines are required above this data rate. Because of this, most communications do not exceed 4800 baud.

4.2 Data Link Layer

The data link layer is responsible for the reliable transmission of bits of information on the point-to-point physical media. The ISO high-level Data Link Control (HDLC) procedure, utilizing the CCITT Link Access Procedure B (LAP-B) node, will be employed at the link layer. This protocol provides the means to activate the link, control the flow of data, recover from errors caused by loss, duplication, or physical media, and to deactivate the link. Since this protocol is point-to-point, it can reliably transfer data between two points in a network only. Addressing and routing are not performed at this layer.

The link layer will be designed to support a minimum of four ports or physical connections. Since one or more ports could be active at a given time, the link layer will obtain information as to which port is to be used from the Network Layer. It will be possible to have up to seven frames of data outstanding (prior to receiving acknowledgement) on each port. It will be necessary to consider the case where satellite links are used and provide for expansion to 128 outstanding frames per port. The design and implementation will not preclude this, however, the initial implementation will only support frames outstanding. Worst case buffer requirements are 4K and 64K respectively.

4.3 Network Layer

The Network Layer is responsible for routing and relay functions within the network. This layer will supply an address in its header which corresponds to the destination node address and will select the best route to use based on cost optimization and throughput criteria. Packets of data are assembled and disassembled at this layer and flow control principles are employed to avoid congestion within the network. Each packet of data is uniquely sequenced in order to provide error recovery between nodes on the network. Virtual circuits are established, maintained, and released as required to support the transport user. The CCITT recommendation X.25 packet level will be used at this layer of the architecture. Since there are many options available to the implementor, the specification will conform to that required to interface to Telenet.

The ISO, in conjunction with the CCITT, is currently developing a standard which is a super set of X.25 to support DTE to DTE communications. The 1980 CCITT recommendation X.25 is asymetrical in that it only supports DTE to DCE communications. The correct standard, therefore, will not support direct Ultimate to Ultimate communications. Ultinet will support the new standard, which will allow both types of communications.

The combination of the network layer, link layer, and physical layer, as described in the preceding paragraphs, comprise the CCITT X.25 standard.

4.4 Transport Layer

The Transport Layer performs the end functions necessary to assure the reliable transmission of user data. This layer does not perform routing functions, and as such, is only activated in the end systems. Address translation from user provided logical address to the network address is performed by this layer. In addition, end to end flow control is performed in order to avoid congestion of buffers in the end systems. The transport layer is able to recover from network generated reset. There is a one-to-one correspondence between session connections and transport connections. Transport connections may be multiplexed on to a single network connection as a function of user specified cost and throughput requirements.

The transport layer will be implemented to conform to ISO DP8073 Transport Protocol Specification. Specifically, Class 2 protocol will be supported as a basic conformance class. In addition, it is desirable to support the ISO Class 4 protocol, which has full error detection and recovery capability. The Class 2 protocol is a subset of the Class 4 protocol.

It is intended that the transport layer will reside on the same communications controller as the X.25 protocol. This will significantly off load the host processor from the necessity to be concerned with the real-time aspects of network communications. In addition, the Class 4 transport protocol has been identified as the class of protocol to be used with local area networks. This approach has been endorsed by the European Manufacturers Association (ECMA) and the IEEE 802 Local Area Network committee. Having the transport layer coresident on a communications controller for synchronous communications will alleviate the necessity to reimplement this protocol for Local Area Network applications provided that the resident MPU on the different controllers is the same.

4.5 Session Layer

The Session Layer will provide the means for application processes to form a bounded logical connection with other application processes. Terminals, or users of terminals, will be logically bound to remote terminals, files, and application processes. The context necessary to support these logical connections is maintained by the session layer. The session layer also provides the capability to synchronize data flow between two users or application processes, and, if necessary, to resynchronize if data is lost.

While there is a one-to-one relationship between a session connection and a transport connection, several session connections may be consecutively established and terminated on a single transport connection. This facility allows the session layer to disconnect without releasing the transport connection when a user of the session service is involved in multiple file or item transfer with significant processing between transfer requests. Connection establishment need only occur at the session layer each time a transfer request is made. System resources can be reassigned upon completion of the transfer if the session is disconnected.

The basic combined subset of the ISO session layer protocol standard will form the basis for the peer-to-peer interchanges between cooperating session entities. This protocol subset makes use of a kernel set of session protocol data units, and may be expanded to encompass interactive procedures in the future should the market warrant it.

4.6 Presentation Layer

Presentation services ensure that information is transferred to the application user in a form that is understood. The meaning - semantics - of the information is fully preserved, while the format and language differences - syntax - are resolved.

Initially, it is intended that users of Ultinet comprise a homogeneous family, and, therefore, no presentation image differences will exist. As such, there is no initital requirement to provide for a presentation layer protocol in the Ultinet product. Should interconnection with non-Ultimate compatible file and data structures be required, the presentation layer protocols now under development within ISO will be implemented to the degree necessary to satisfy this need.

4.7 Application Layer

The application layer provides the interface from the user, the application program, and the operating system, to the network system. The Ultimate operating system will be modified and augmented to facilitate file transfer and remote logon capabilities.

Initially, a remote file definition item will be defined for files located on a remote machine and accessible from the local machine. System routines, such as Open, Retix, Getiton, etc. will be modified to send the appropriate command to the remote machine by interfacing to the session layer. Using the same basic principles, remote logon will be accomplished in a follow-on phase of the development.

While the definition items must be predefined, subsequent use and access is transparent to the user. In this way, files may be moved to different nodes without affecting the user application process. It will be possible to distribute files in the network in order to optimize the use of local and communications resources. Centralized control, of, for example, a file containing product pricing data can be maintained, while access to that data will be possible from anywhere in the network.

5. Support System Software

5.1 Buffer/Queue Manager

A single buffer and queue scheme will be defined that will support each of the network processes. Each task will have associated with it a queue which will contain pointers to data and commands. Queue entries will be added and released by the common queue manager.

Data buffers will be made available upon request for both input and output. Pointers to the buffer will define the start, end, and current location within the buffer. On output, user data will always start at a predefined location other than the "start". This will enable layered protocol to be added to the user data without necessitating the movement of the data in memory. Data moves occur only during transfer into or out of the host CPU to or from the communications controller.

Data buffers may be released only by the user application interface support software in the host and the transport layer in the communications controller, except when the network layer is performing a relay function.

5.2 Additional Support Software

Additional support software may be identified during the course of the design of the network software. As this occurs, this section of the definition document will be updated.

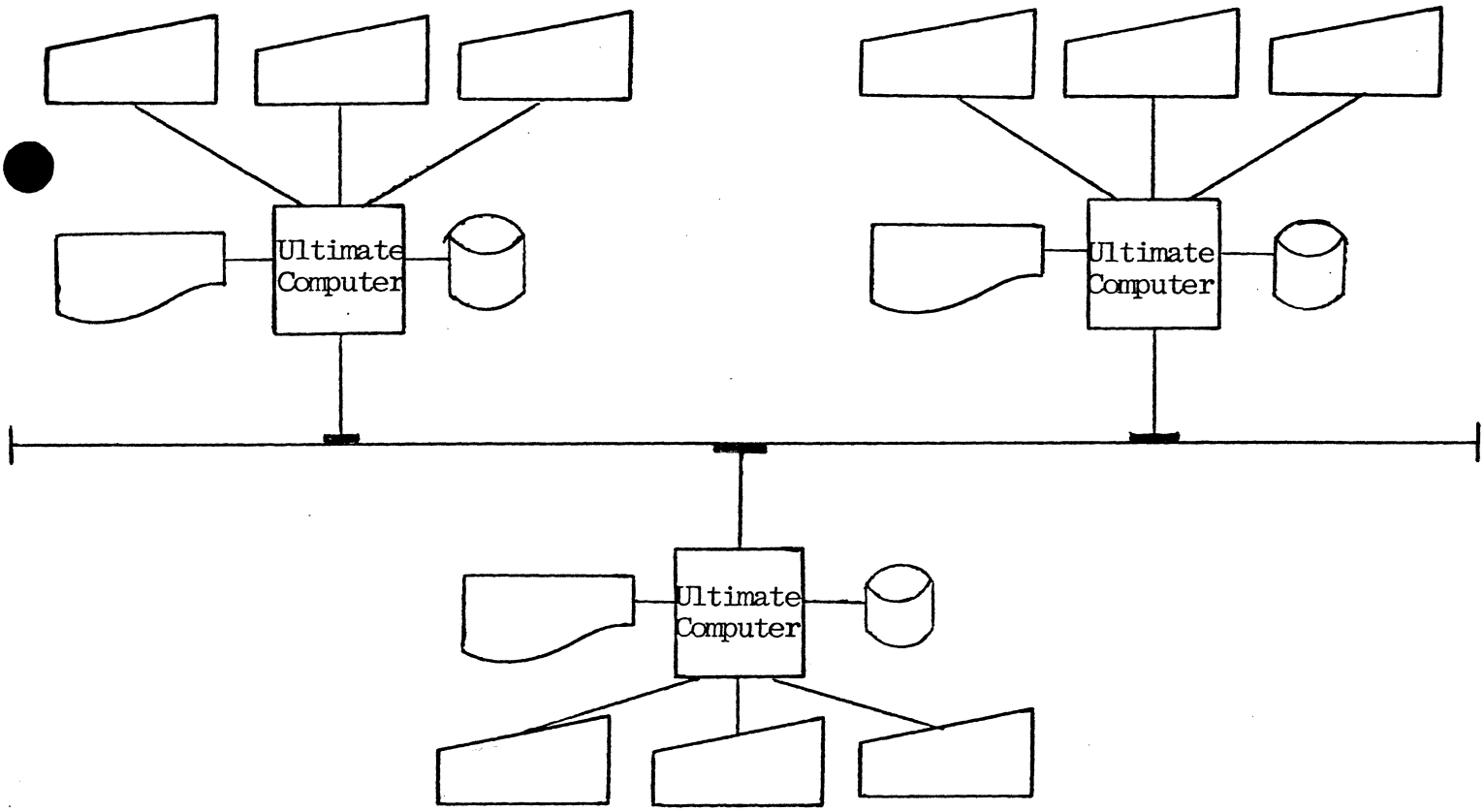
6. Implementation Plan

A phased implementation approach to the development of Ultinet will be used in order to provide an increasing network capability. Intermediate releases of software/firmware/hardware will be made to provide product that may be marketed on a timely basis. A complete implementation plan will be generated and will be the subject of a separate document.

7. Network Configurations

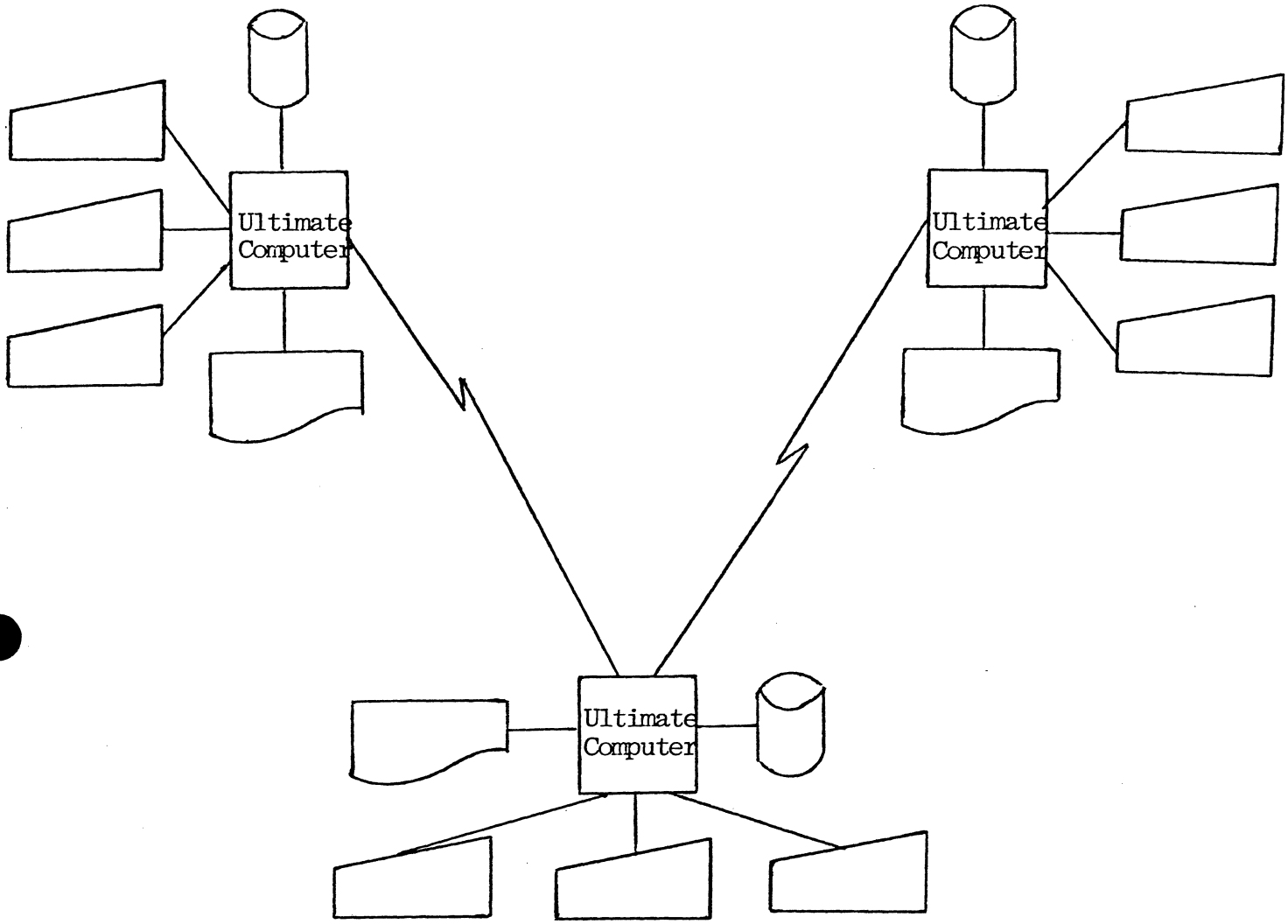
Several network topologies are supported on the network: local area networks, private line networks, and value added carrier networks.

The configuration of the local area network is:



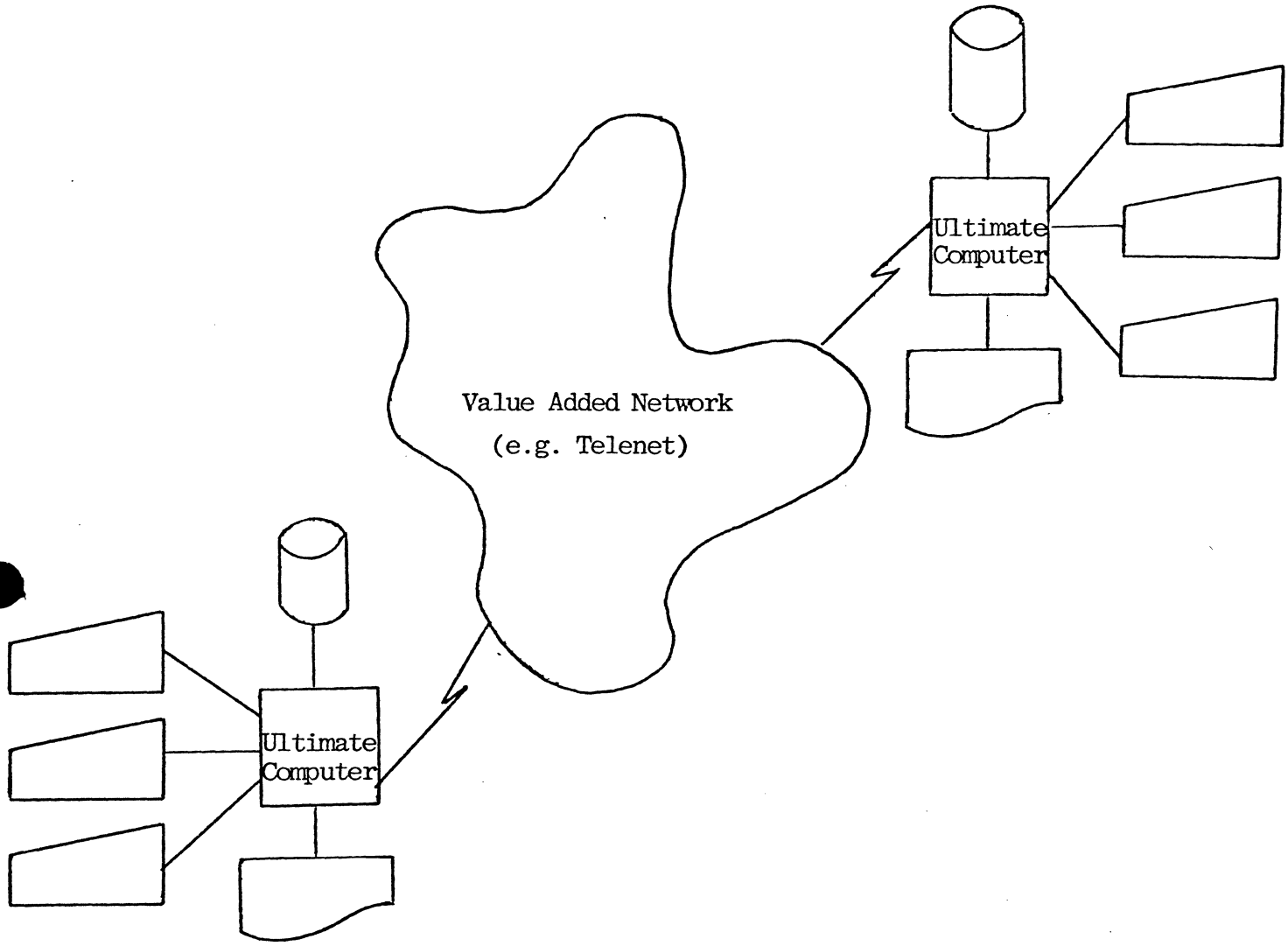
Ultimate computers are connected to a coaxial cable which runs to the necessary locations in the facilities. Access from on Ultimate computer to another is done over the high speed coaxial cable.

The configuration of the private line network is:



The connections between the Ultimate computers are made using leased telephone lines or using private telephone lines.

The attachment to a value added network is:



The connection into the value added network is made using the X.25 interface provided by the carrier.

MEMORANDUM

TO: DISTRIBUTION DATE: 3/24/83

FROM: FAROKH DEBOO REF: FD0583

SUBJECT: MINUTES OF TODAY'S MEETING TO DISCUSS OPEN ITEMS ON THE ULTINET PROJECT

ATTENDEES:
DENNIS AULER
CAROL CARMICHAEL
FAROKH DEBOO
JOHN NEUMANN

CC: SCOTT BREEDEN
CARL MARGOLIS

---

1. THE FIRST DRAFT OF THE INTERFACE SPECIFICATION IS IN THE PROCESS OF BEING COMPLETED. IT WILL BE READY NEXT WEEK.

2. THE DETAILS, SCHEDULE AND PERSONNEL ASSIGNMENT FOR THE NEW UPDATE & RETRIEVAL LOCKS IMPLEMENTATION NEEDS TO BE ADDRESSED. OPEN ITEM FOR CARL.

3. THE SCHEDULE AND PERSONNEL ASSIGNMENT FOR THE NEW GROUP & ITEM LOCKS IMPLEMENTATION NEEDS TO BE ADDRESSED. OPEN ITEM FOR CARL.

4. DETAILS ON THE COMM-SPOOLER PROCESS. THIS INFORMATION WILL BE PROVIDED TO SOME EXTENT IN THE INTERFACE SPECIFICATION.

5. NETWORK GENERATION, INCLUDING THE VARIOUS TABLES TO BE CREATED, INITIALIZED, ETC. THE INDIVIDUAL PROGRAMMERS WILL CREATE SUBROUTINES FOR THEIR OWN AREAS OF THE PROJECT, AND THESE WILL BE CONSOLIDATED INTO A "START-NETWORK" TYPE VERB/PROC ALONG WITH ANY OTHER COMMON FUNCTIONS.

6. STATUS ON A NEW ENTRY POINT IN RETIX TO PERFORM A RETIX PLUS COPY TYPE OF FUNCTION. OPEN ITEM FOR CARL.

7. IS THE CONNECT TABLE TO BE DESIGNED TO BE ASSOCIATED ONLY WITH THE FILE SERVER PROCESS OR IS IT TO BE GLOBAL TO ALL PROCESSES ON THE NODE AT WHICH IT RESIDES? FOR PHASE 1, IT MAY BE USED ONLY BY THE FILE

SERVER, BUT IT WILL BE GLOBAL FOR LATER PHASES OF THE PROJECT. A COPY OF THE CURRENT PARAMETERS IN THIS TABLE WAS PASSED OUT.

8. ERROR MESSAGES RELATED TO NETWORKING WILL BE ASSIGNED NUMBERS STARTING WITH THE LETTER "C" AND BEGINNING WITH NUMBER C000.

9. SPECIFIC NAMES WILL BE CREATED FOR EACH OF THE TABLES USED ON THIS PROJECT. THE FOLLOWING TABLES ARE CURRENTLY DEFINED:

   * THE OPEN-FILES TABLE THAT WILL BE ASSOCIATED WITH THE FILE SERVER PROCESS AND WILL HAVE AN ENTRY CREATED FOR EACH FILE THAT WAS OPENED BY A REMOTE NODE ON THIS MACHINE.

   * THE CONNECT TABLE THAT WILL KEEP TRACK OF CONNECTIONS MADE AT THE APPLICATION LEVEL (SEE ALSO POINT 7 ABOVE).

   * THE BMS TABLE THAT WILL BE MAINTAINED BY EACH USER PROCESS WHEN IT FINDS THAT IT IS TRYING TO OPEN A FILE AT A REMOTE NODE.

10. THE FORMAT FOR THE GETITM COMMAND AND RESPONSE NEEDS TO BE REDEFINED. CAROL AND FAROKH TO RESOLVE THIS POINT.

11. HANDLING OF CODE COMMON TO BOTH THE VIRTUAL FILE ACCESS ROUTINES AND TO THE FILE SERVER PROCESS. THE CURRENT SYSTEM'S FILE ACCESS ROUTINES WILL BE MODIFIED TO BE ABLE TO PROCESS BOTH LOCAL AND REMOTE FILES. WHERE NECESSARY, ADDITIONAL INTERFACES AND ENTRY POINTS WILL BE SET UP TO PROVIDE THE FUNCTIONALITY NECESSARY FOR THE FILE SERVER PROCESS.

    THE FILE SERVER IN TURN WILL HAVE ASSOCIATED WITH IT NEW CODE THAT WILL DIRECTLY USE THESE SYSTEM ROUTINES WITH THE NEW INTERFACES/ENTRY POINTS CREATED. FOR AN EXAMPLE REFER TO POINT 15 BELOW ON GBMS.

12. CONNECT REQUEST FORMAT. THIS WILL BE ADDRESSED IN THE INTERFACE SPECIFICATION.

13. OPEN-FILE TABLE. A COPY OF THE PARAMETERS IN THIS TABLE WAS PASSED OUT.

14. HANDLING OF OVERFLOW SPACE AND BUFFER MANAGEMENT IN GENERAL. THIS PROBLEM IS BEING ADDRESSED AND WILL EVENTUALLY BE INCLUDED IN THE INTERFACE SPECIFICATION.

15. THE GBMS SUBROUTINE HAS BEEN MODIFIED. IT PROVIDES ADDITIONAL INFORMATION IN H0 WHEN RTNFLG WAS SET AND THE

SUBROUTINE RETURNED WITH RMBIT = 0. IN THIS EVENT H0 WILL BE LOADED ACCORDING TO WHICH ONE OF THE FOLLOWING CONDITIONS CAUSED IT TO FAIL:

* INVALID ACCOUNT NAME
* INVALID FILE NAME
* RETRIEVALS NOT PERMITTED
* UPDATES NOT PERMITTED
* FOUND A REMOTE Q-POINTER

THIS WILL PROVIDE THE FUNCTIONALITY NECESSARY FOR THE OPEN SUBROUTINE AT THE REQUESTING USER'S SIDE TO RESPOND TO A REMOTE FILE DEFINITION ITEM AND ALSO FOR A REMOTE FILE SERVER PROCESS TO RETURN THE RELEVANT STATUS FIELD SHOULD IT FIND THAT IT CANNOT SUCCESSFULLY OPEN THE REFERENCED FILE ON ITS MACHINE.

16. ADDITIONAL DATA INTEGRITY ACROSS THE ASYNCHRONOUS INTERFACE TO THE BLACK BOX AND/OR ACROSS THE COMMUNICATION INTERFACE. IT MAY BE NECESSARY FOR THE LINE DRIVER TO INCLUDE A CHECKSUM ON THE DATA IT IS PASSING ACROSS THE ASYNC INTERFACE. A SIMPLE PROTOCOL WITH SOME FORM OF ACK AND REJECT COMMANDS IS BEING CONSIDERED.

17. FIND ANY CODE IN THE SYSTEM THAT DIRECTLY PERFORMS FILE ACCESSES WITHOUT THE STANDARD FILE ACCESS ROUTINES (EG. CLEARFILE IN BASIC) AND MODIFY IT TO BE ABLE TO HANDLE REMOTE FILES IF NECESSARY.

18. THE FORMAT OF THE CONTENTS OF THE BASE, MODULO & SEPAR FIELDS FOR IDENTIFYING A REMOTE FILE IS BEING FINALISED.

M E M O R A N D U M

TO: DAVE GOULD

FROM: JOHN NEUMANN

DATE: MARCH 28, 1983

SUBJECT: NETWORK ARCHITECTURE

CC: CARL MARGOLIS
FRANK KACERAK

IT WOULD APPEAR ON FIRST EXAMINATION THAT THE IMPLEMENTOR AND DESIGNER OF COMMUNICATIONS PRODUCT CAPABILITY IS FACED WITH SEVERAL ALTERNATIVE APPROACHES TO SOLVING THE COMMUNICATIONS PROBLEM. A KEY ELEMENT IN THE DECISION PROCESS IS A CLEAR UNDERSTANDING OF THE PROBLEM TO BE SOLVED AND THE OBJECTIVE TO BE ACHIEVED BY THE DEVELOPMENT OF A PRODUCT. ONCE ALL THE PARTIES TO THE DECISION ARE IN AGREEMENT ON THESE TWO ISSUES THE SPECTRUM OF CHOICES WILL NARROW.

WITH THIS IN MIND THEN, IT IS IMPORTANT TO IDENTIFY WITH A SINGLE RECOGNIZABLE SOURCE FOR A CLEAR CONCISE DEFINITION OF THE OBJECTIVE AND GOALS OF THE DEVELOPMENT PROJECT RELATED TO PROVIDING COMMUNICATIONS ON THE ULTIMATE FAMILY OF COMPUTER PRODUCTS. I HAVE ASSUMED THAT THIS SOURCE IS THE ORIGINAL PRIVATE PLACEMENT MEMORANDUM WHICH DEFINES THE OBJECTIVE OF THE ULTINET SUBSYSTEM AS "THE CREATION OF HARDWARE AND SOFTWARE TO PROVIDE A COMMUNICATIONS CAPABILITY WHICH MAY BE INTEGRATED WITH EXISTING COMPUTER SYSTEMS IN THE ULTIMATE PRODUCT LINE TO PERMIT A USERT OF AN ULTIMATE COMPUTER SYSTEM TO ACCESS AND UPDATE FILES IN ANOTHER SYSTEM ON-LINE OR TO LOG ON TO A REMOTE SYSTEM TO OPERATE THAT SYSTEM", AND "PERMIT ACCESS TO ON-LINE REMOTE ULTIMATE COMPUTER SYSTEMS AND TO TRANSMIT DATA OVER COMMUNICATIONS NETWORKS IN ORDER TO AFFORD USERS GREATER FLEXIBILITY IN THE USE AND DISTRIBUTION OF THEIR COMPUTING RESOURCES". IF THIS IS INDEED THE DIFINITION OF THE PROJECT AND OBJECTIVE, WHAT ARE THE ALTERNATIVES?

SYSTEMS NETWORK ARCHITECTURE:

IBM, OVER THE PAST DECADE, HAS DEVELOPED AND EVOLVED A COMMUNICATIONS SYSTEM KNOWN AS 'SNA'. SINCE ITS INTRODUCTION IN THE EARLY 1970'S A LARGE NUMBER OF COMPANIES HAVE DEVELOPED AND OFFERED PRODUCT TO CO-EXIST WITHIN THIS ARCHITECTURE. THIS INTERCONNECTION TO SNA HAS BEEN ALMOST EXCLUSIVELY AS WHAT IS KNOWN AS A 'TYPE 2 PHYSICAL UNIT (PU)', OR MORE COMMONLY AS AN EMULATION OF THE 3270 AND, MORE RECENTLY, 3770 AND 3790. THE FUNCTIONALITY PROVIDED IS THAT OF EITHER INTERACTIVE (3270) OR BATCH (3770/3790) SNA COMMUNICATIONS. THE ONLY FIRM I AM AWARE OF WHICH OFFERS A PU TYPE 3 (HOST) PRODUCT OFFERING IS AMDAHL CORPORATION WITH ITS 470/4705 SUPER COMPUTER/FRONT END PROCESSOR OFFERINGS. IN ORDER TO ACHIEVE THE OBJECTIVE AS STATED ABOVE, A TYPE 3 PU WOULD HAVE TO BE IMPLEMENTED WHICH WOULD ALLOW ULTIMATE TO ULTIMATE COMMUNICATIONS ACCROSS SNA DOMAINS. SUPPORT OF SNA TYPE 2 PU

ON ULTIMATE WOULD ALLOW ULTIMATE SYSTEMS TO LOOK LIKE IBM 3270/3770/3790 SYSTEMS TO EITHER REPLACE EXISTING IBM EQUIPMENT OR ADD-ON TO AN EXISTING SNA NETWORK.

WHILE SNA IS AN INDUSTRY PROVEN AND ACCEPTED ARCHITECTURE, IT IS VERY COMPLEX. FEW OUTSIDE OF IBM UNDERSTAND THE FULL RAMIFICATION OF SUCCESSFULLY SELLING IN THIS MARKET. FOR EXAMPLE, IF AN ADD-ON PRODUCT IS SOLD TO AN EXISTING SNA USER WHICH IS NOT IBM, IBM WILL NOT RECONFIGURE SNA TO ACCOMODATE THIS ADD-ON PRODUCT. THIS MEANS THAT EITHER THE USER HAS IN-HOUSE STAFF TO ACCOMPLISH THIS, OR ELSE THE VENDOR MUST BE IN A POSITION TO ACCOMPLISH THIS TASK. GIVEN THAT THERE ARE SOME 800 NETWORK VARIABLES, ALL OF WHICH MUST BE DEFINED CORRECTLY FOR SNA TO WORK, THIS IS A VERY FORMIDABLE TASK. ULTIMATE MUST BE PREPARED TO PROVIDE THIS TYPE OF SUPPORT IF IT IS GOING TO GO INTO THE SNA WORLD, SINCE THE USERS DEMAND IT (THEY GET IT FROM IBM). I DON'T WISH TO NEGATE THE DESIREABILITY OF ENTERING THIS MARKET, I MERELY POINT OUT THAT IT IS NOT AS EASY AS IT MIGHT APPEAR ON THE SURFACE. THOSE WHO HAVE DONE IT RIGHT, HAVE SEEN FINANCIAL SUCCESS WITHIN THEIR PRODUCT OFFERING. THOSE WHO HAVEN'T, HAVE LOST MONEY. THE LEVEL OF COMMITMENT REQUIRED TO SUPPORT PRODUCT IN THE SNA WORLD IS CONSIDERABLE. SNA CONTINUES TO EVOLVE SUCH THAT THERE IS NO GUARANTEE THAT PRODUCT DEVELOPED 'TO SPEC' TODAY WILL ACTUALLY WORK WHEN THE DEVELOPMENT CYCLE IS COMPLETED WITHOUT FURTHER MODIFICATIONS.

IN ADDITION TO THE SUPPORT ISSUES THERE IS THE GENERAL PROBLEM OF SELLING INTO THIS MARKET. THE USER WHO REQUIRES SNA PRODUCT IS QUITE SOPHISTICATED AND IS USUALLY ON THE "FORTUNE" LIST. THE SALES FORCE NEEDED TO ATTACK THIS MARKET MUST BE EQUALLY SOPHISTICATED IN ITS KNOWLEDGE OF LARGE SYSTEM COMMUNICATIONS PROBLEMS. MOST IMPORTANTLY, HOWEVER, THE VENDOR MUST BE CREDIBLE. HE MUST HAVE A SUCCESSFUL TRACK RECORD IN COMMUNICATIONS. THE TYPICAL MIS MANAGER WILL NOT GENERALLY TAKE RISKS IN PURCHASING EQUIPMENT. THAT TRANSLATES QUITE SIMPLY INTO THE FACT THAT ADDED FUNCTIONALITY, COST SAVINGS, AND PERFORMANCE IMPROVEMENTS ARE NOT INDUCEMENTS TO PURCHASE FROM AN UNKNOWN.

FINALLY, THE SNA WORLD IS ALSO A COBOL WORLD. THOUGH NOT REQUIRED, IT IS HIGHLY DESIREABLE. IF NOT COBOL, THEN EFFICIENT MEANS TO CONVERT CUSTOMER APPLICATIONS TO RUN ON THE ULTIMATE SYSTEM ARE REQUIRED.

OPEN SYSTEMS INTERCONNECTION:

THIS EMERGING ARCHITECTURE, THOUGH NOT 'PROVEN' IS CERTAINLY ACCEPTED IN THE INDUSTRY. EVERY VENDOR OF COMMUNICATIONS SYSTEMS, FOR EXAMPLE IBM AND HONEYWELL, CLAIM TO HAVE A LAYERED ARCHITECTURE WHICH IS COMPATIBLE WITH THE REFERENCE MODEL FOR OPEN SYSTEMS INTERCONNECTION. PHILOSOPHICALLY THEY ARE RIGHT! THEIR PROTOCOLS ARE NOT COMPATIBLE, FOR THE PROTOCOLS OF OSI ARE NOW BEING FORMALIZED BY ISO AND CCITT, AS WELL AS THE NATIONAL STANDARDS BODIES OF THE WORLD. AS OF THIS TIME THE FIRST FIVE LAYERS SEEM TO BE QUITE STABLE, WHILE THE UPPER TWO LAYERS AND MANAGEMENT ASPECTS ARE STILL UNDERGOING CONSIDERABLE DEBATE WITHIN THE STANDARDS WORLD.

OSI AS A CONCEPT EMBODIES THE REQUIREMENT TO DEFINE AN ARCHITECTURE WHICH WILL ALLOW MULTIPLE VENDORS TO IMPLEMENT COMMUNICATION SYSTEMS THAT WILL INTERCONNECT. THE USER DEMANDS THIS CAPABILITY IN ORDER TO AVOID BEING 'ROPED INTO' A SINGLE VENDOR OFFERING. AN ORGANIZATION IN THE UNITED STATES, WHOSE MEMBERS ARE DRAWN FROM THE "FORTUNE" COMPANIES, SPRUNG UP SEVERAL YEARS AGO IN AN ATTEMPT TO INFLUENCE VENDORS TO QUICKLY ADOPT THE OSI PRINCIPLE. THIS GROUP, THE NETWORK USERS ASSOCIATION, IS STILL ACTIVE TODAY AND HAS LIAISON WITH THE ANSI COMMITTEES DEVELOPING THE STANDARDS. THEY HAVE BEEN MOST CRITICAL OF ANSI WHEN NEGATIVE US BALLOTS HAVE BEEN SUBMITTED TO ISO ON BOTH THE REFERENCE MODEL AS WELL AS THE TRANSPORT STANDARDS. THIS WAS PARTLY DUE TO A MISUNDERSTANDING OF THE REASON FOR THE NEGATIVE BALLOTS, BUT ALSO OUT OF FRUSTRATION AND CONCERN THAT THE STANDARDS PROCESS WAS TAKING TOO LONG. THIS IS A CLEAR INDICATION OF THE EAGERNESS WITH WHICH THE LARGE USER COMMUNITY IS AWAITING THE COMPLETION OF THE WORK ON OSI PROTOCOLS.

OSI DEFINES A PEER PROTOCOL WHICH WILL ALLOW IMPLEMENTATIONS TO COMMUNICATE AS EQUAL PARTNERS IN A NETWORK ENVIRONMENT. THIS MEANS THAT THERE IS NO MASTER/SLAVE RELATIONSHIP BETWEEN COOPERATING SYSTEMS AS THERE IS IN 'HIERARCHICAL' NETWORKS SUCH AS SNA. THIS MEANS THAT ULTIMATE SYSTEMS WILL BE ABLE TO COMMUNICATE WITH OTHER ULTIMATE SYSTEMS IN A USER TRANSPARENT ENVIRONMENT, SATISFYING THE PROJECT OBJECTIVES AS STATED ABOVE. SHOULD ULTIMATE DESIRE TO BE ABLE TO INTERCONNECT WITH OTHER VENDORS (IBM WILL PROVIDE A GATEWAY FROM SNA, FOR EXAMPLE) IN THE FUTURE, THE ONLY REQUIREMENT WOULD BE TO ADD THE PROTOCOLS OF LAYERS 6 AND 7, WHICH ARE NOT A PART OF THE CURRENT PHASE ONE/TWO IMPLEMENTATION PLAN.

MY CONVERSATIONS WITH HONEYWELL AND DEC INDICATE THEIR STRONG SUPPORT FOR OSI AND THE EMERGING PROTOCOL STANDARDS. DECNET, FOR EXAMPLE, WILL SOON BE OFFERING FULL COMPATIBILITY WITH ISO PROTOCOLS THROUGH LAYER 4 (TRANSPORT). HONEYWELL IS ALSO EVOLVING DSA TOWARDS THE END GOAL OF BEING COMPATIBLE. MR. DIETER FISCHER OF HIS (PHOENIX) PRODUCT PLANNING HAS INDICATED TO ME THAT MY DESIRE TO DEVELOPE A COMMUNICATIONS CONTROLLER TO SUPPORT THE FIRST FOUR LAYERS OF THE ARCHITECTURE WAS CONSISTENT WITH HONEYWELL'S NEEDS AS WELL. I BELIEVE THAT COOPERATIVE DEVELOPMENT EFFORTS ARE CERTAINLY POSSIBLE WITH HONEYWELL IN THIS AREA, AND I THINK IT MAY BE POSSIBLE TO EXPLORE THIS WITH DEC AS WELL.

IN SUMMARY, IT IS MY BELIEF THAT OSI PROVIDES A FRAMEWORK WHITHIN WHICH ULTIMATE CAN SUCCESSFULLY COMPETE IN THE EMERGING DATA COMMUNICATIONS MARKETPLACE WITHOUT THE ENCUMBERANCES THAT THE SNA ENVIRONMENT PLACES ON ITS USERS.

ULTIMATE NETWORK ARCHITECTURE:

A THIRD POSSIBILITY WHICH I WILL BRIEFLY MENTION IS AN ULTIMATE ARCHITECTURE WHICH IS NEITHER SNA NOR OSI. THIS APPROACH HAS LITTLE DEVELOPMENT OR MARKETING BENEFIT, SINCE THERE ALREADY EXISTS AN INDUSTRY ACCEPTED ARCHITECTURE. IF ADOPTED, DEVELOPMENT TIME WOULD CERTAINLY INCREASE AND MARKETABILITY WOULD DECREASE. THERE IS NO BENEFIT THAT I CAN SEE IN REINVENTING THE WHEEL AT THE EXPENSE OF LOOSING MARKET

SHARE. THIS APPROACH IS BEING USED BY MICRODATA, BY THE WAY, WHICH WILL DELAY SIGNIFICANTLY THEIR ABILITY TO ENTER THE MARKET WITH A VIABLE PRODUCT (END OF 1986) FOR THE REALITY AND SEQUEL PRODUCT LINES. THEIR APPROACH IS BASED ON THE BELIEF THAT SOVEREIGN IS THE ONLY DISTRIBUTED SYSTEM WITHIN THEIR PRODUCT LINE (THIS IS A RESULT OF STRONG LOBBYING BY CMC).

NO MATTER WHAT DIRECTION ULTIMATE FINALLY DECIDES ON (SNA OR OSI), THIS APPROACH SHOULD NOT BE CONSIDERED.

SUMMARY:

GIVEN THE STATED OBJECTIVES THAT THIS PRODUCT DEVELOPMENT IS INTENDED TO SATISFY I RECOMMEND THAT WE CONTINUE ON THE DEVELOPMENT PATH WE HAVE EMBARKED UPON. SNA CAN AND SHOULD BE CONSIDERED AS A VIABLE PRODUCT OFFERING IN THE FUTURE, BUT DOES NOT MEET THE OBJECTIVES AS STATED. OSI WILL EVENTUALLY DOMINATE THE NON SNA DISTRIBUTED PROCESSING ENVIRONMENT, AND SNA (IBM) WILL HAVE TO ACCOMODATE THIS WITHIN THEIR ARCHITECTURE. IN FACT, MY CONTACTS WITHIN THE IBM NETWORK ARCHITECTURE DIVISION IN RALEIGH ASSURE ME THAT THIS IS TRUE. KNOWING THIS, I DON'T BELIEVE THAT ULTIMATE CAN MAKE A MISTAKE IN FOLLOWING THE CURRENT APPROACH TO DISTRIBUTED SYSTEMS AND DATA COMMUNICATIONS.

M E M O R A N D U M

TO: DENNIS AULER DATE: 3/29/83

FROM: FAROKH DEBOO REF : FD0683

SUBJECT: INFORMATION FOR YOUR MARCH 83 STATUS REPORT

2/28 - 3/04 THE PROJECT WAS SPLIT INTO THREE SECTIONS BETWEEN THE THREE PROGRAMMERS. THE USER SIDE OF THE VIRTUAL FILE ACCESS ROUTINES WAS ASSIGNED TO ME. STARTED DESIGN OF THE INDIVIDUAL MODULES.

3/07 - 3/11 CONTINUE WITH DESIGN. STARTED WRITING A DESIGN SPECIFICATION BEGINNING WITH THE OPEN SUBROUTINE.

3/14 - 3/25 TEMPORARILY STOPPED WRITING DESIGN SPECIFICATION AFTER CONSULTATION WITH YOU. CONTINUED SETTING UP MODIFICATIONS FOR THE OPEN SUBROUTINE.

THERE WAS SOME CONFUSION ABOUT WHICH PROGRAMMER SHOULD ACTUALLY BE MODIFYING THE SYSTEM ROUTINES. THIS POINT AND SEVERAL OTHER OPEN ITEMS WERE DISCUSSED IN THE 3/24 MEETING.

3/28 - PRESENT HAD THE 3/28 MEETING WITH CARL TO SET UP AMONGST OTHER THINGS DETAILED ASSIGNMENTS AS TO WHO MODIFIES WHICH SYSTEM ROUTINE, TO DECIDE ON WHAT TYPE OF INTERNAL SPECIFICATIONS NEED TO BE WRITTEN AND FOR THE SETTING UP OF A PROJECT SCHEDULE.

MEMORANDUM

TO: COMMUNICATIONS GROUP

FROM: CAROL CARMICHAEL

SUBJECT: NOTES ON DECISIONS OF THE INFORMAL MEETING, MARCH 30, 1983.

Dennis, Carol
J.N, FD

A) IF THE FILE-SERVER PROCESSOR ENCOUNTERS AN ABORT CONDITION WHEN EXECUTING CODE FOR A REMOTE REQUEST THEN THE FSP WILL RETURN FROM THE DEBUGGER AND ISSUE A DISCONNECT TO THE REQUESTING PARTY.
(REQUEST)

B) NO DISCONNECT RESPONSE WILL BE SENT BY THE FSP AFTER A DISCONNECT-INDICATION IS RECEIVED AND PROCESSED.

C) A PSEUDO-SESSION LAYER MUST KEEP TRACK OF THE LOGICAL CONNECT NUMBERS. THIS PSEUDO-SESSION LAYER EXISTS ABOVE THE LINE-DRIVER AND BELOW THE COMMUNICATIONS PROCESSOR.

D) ONLY THE VIRTUAL PROCESSOR (INTERFACE WITH THE USER) HAS A TIMER ASSOCIATED WITH IT. IT WILL TIME-OUT ON CONNECT REQUESTS.

TO: CARL MARGOLIS

FM: JOHN NEUMANN

SUBJ: TRIP REPORT TO ZBK/HONEYWELL

DATE: APRIL 25, 1983

THE PURPOSE OF THIS TRIP WAS TO DISCUSS DEALER DATA COMM REQUIREMENTS WITH AL PRICE IN TORONTO, DISCUSS PRODUCT FUNCTIONALITY FOR AN X.25 INTERFACE WITH ZBK TELECOMPUTERS IN MONTREAL, AND DISCUSS ULTIMATE DATA COMM REQUIREMENTS WITH HONEYWELL IN BILLERICA.

AL PRICE DISCUSSION:

I MET WITH AL PRICE OF CALNEK-PRICE IN MONTREAL ON APRIL 11, 1983, TO DISCUSS HIS APPLICATION IN DATA COMMUNICATIONS. THE TYPICAL CONFIGURATION HE SELLS TO ONE OF HIS LARGE CUSTOMERS IS THE DPS HONEYWELL SYSTEM, USING A DEVICE MANUFACTURED BY MEMOTEC TO INTERFACE X.25 TO DATAPAC IN CANADA. THIS CUSTOMER HAS A NUMBER OF SYSTEMS THROUGHOUT CANADA WHICH ARE CONFIGURED IN THIS MANNER. HE HAS DEVELOPED THE SOFTWARE NECESSARY TO INTERFACE TO THE MEMOTEC DEVICE. THIS DEVICE IS VERY EXPENSIVE COMPARED TO THE ZBK BOX. AN 8-PORT VERSION COSTS HIM $4000.00 CANADIAN. THIS IS ABOUT TWICE THE COST OF AN EQUIVALENT ZBK DEVICE. THE SOFTWARE NECESSARY TO DO REMOTE LOG ON AND FILE TRANSFER HAS ALSO BEEN DEVELOPED. I GOT THE IMPRESSION THAT THIS SOFTWARE IS VERY SPECIFIC TO HIS REQUIREMENTS AND PROBABLY WOULD NOT BE OF MUCH USE TO US, HOWEVER THE APPROACH HE USED SHOULD BE INVESTIGATED. I HAVE ASKED DENNIS AULER TO CALL AL AND POSSIBLY GET INTO A MORE DETAILED DISCUSSION OF HIS IMPLEMENTATION.

HE ALSO HAS A LOCAL AREA NETWORK PROBLEM WHICH REQUIRES THE DATA RATE WE ARE PLANNING TO OFFER. HE IS CONCERNED THAT SINCE WE ARE POSSIBLY GOING TO PROVIDE AN INTERIM PRODUCT WHICH MAKES USE OF THE ASYNC PORTS THAT IT WILL NOT MEET HIS OVERALL THROUGHPUT REQUIREMENTS. THE LONG TERM SOLUTION IS WHAT HE NEEDS IN THE SHORT TERM, BUT HE UNDERSTANDS THE TIME IT TAKES TO PROVIDE THIS TYPE OF CAPABILITY.

ADDITIONAL REQUIREMENTS HE MENTIONED WERE THE ABILITY TO GATEWAY TO SUCH SERVICES AS TELEX AND OTHER NETWORK SERVICES. IN ADDITION, HE STRESSED A NEED FOR FULL MODEM CONTROL AND XON/XOFF FOR THE DEC VERSION OF THE SYSTEM. THIS IS ALSO NEEDED TO CONNECT TO THE X.25 BOX AND LAN BOX.

ZBK TELECOMPUTERS:

ON APRIL 12TH I WENT TO MONTREAL TO LOOK INTO THE ZBK X.,25 INTERFACE PRODUCT. WE SPENT SEVERAL HOURS DISCUSSING THE TECHNICAL DETAILS OF THE PRODUCT AND THEN WENT TO A CUSTOMER INSTALLATION TO SEE A DEMO OF THE PRODUCT CONNECTED TO DATAPAC. THE PRODUCT WORKS AS ADVERTISED. IN ADDITION, IT

IS POSSIBLE TO RUN THE MULTIPLEXER BACK TO BACK TO ACHIEVE ULTIMATE TO ULTIMATE COMMUNICATIONS. SO THE FIRST PHASE PRODUCT CAN INCLUDE X.25 TELENET, ULTIMATE TO ULTIMATE SWITCHED AND ULTIMATE TO ULTIMATE LAN.

THE ONLY REAL CONCERN I HAVE IS THE SIZE AND APPARENT STABILITY OF THE COMPANY. IT IS SMALL, THE DO PRODUCTION RUNS WHEN THE NEED TO (ONCE A MONTH), AND MAY HAVE A CASH FLOW PROBLEM. IT IS A SUBSIDIARY OF A LARGER COMPANY (HOW BIG, I DON'T KNOW) SO THERE MAY NOT BE ANY PROBLEM. IN ANY EVENT, THE SOFTWARE INTERFACE TO THE DEVICE IS INDUSTRY STANDARD X.28. WHILE THE PRODUCT IS CERTIFIED ON DATAPAC IT MUST BE CERTIFIED ON TELENET. I HAVE BEEN ASSURED THAT THIS WOULD BE DONE IF WE REQUIRE IT.

I HAVE A COMPLETE SET OF THE FUNCTIONAL DESCRIPTION WHICH WE ARE CURRENTLY EVALUATING.

THE BOTTOM LINE..... I THINK AND RECOMMEND WE TAKE THE NEXT STEP TO SECURE AN OEM RELATIONSHIP IN ORDER TO BEGIN TAKING DELIVERY OF THE PRODUCT. HE WILL SHIP US TWO/THREE UNITS AS SOON AS I WANT THEM. WE WILL NEED THEM FAIRLY SOON IN ORDER TO BEGIN DRIVER (X.28) DESIGN AND IMPLEMENTATION. THEY WILL BE REQUIRED TO CHECKOUT PHASE ONE PRODUCT SOON AS WELL.

HONEYWELL DISCUSSIONS:

ON APRIL 13TH AND 14TH I MET WITH DICK LEMAY AND VARIOUS DATA COMM EXPERTS FROM HONEYWELL TO DISCUSS ULTIMATE COMMUNICATIONS REQUYIREMENTS, AND HOW THEY MIGHT BEST BE SERVED BY EXISTING PRODUCT WITHIN HONEYWELL (MLCP/NMLC). AFTER LENGTHY DISCUSSION WITH THE NMLC DESIGN/IMPLEMENTATION PEOPLE IT BECAME OBVIOUS THAT THE CURRENT FIRMWARE SET ON THE NMLC WOULD NOT MEET OUR COMMUNICATIONS REQUIREMENTS. THAT LEAVES AN OPTION TO DESIGN OUR OWN FIRMWARE FOR THE BOARD. NO PROBLEM, IT CAN BE DONE. THERE IS A LARGE RISK FACTOR WHICH WAS CONSIDERED. THE NMLC CANNOT BE EXPANDED TO ACCOMODATE MORE MEMORY. THE MEMORY ON BOARD IS LIMITED TO 4K FOR THE Z80 AND 64K FOR THE 2901. GIVEN THE ON-BOARD BUFFER REQUIREMENTS AND PROGRAM SPACE, THE RISK IS THAT YOU RUN OUT OF MEMORY, OR THAT SYSTEM THROUGHPUT WILL NECESSARILY DEGRADE DUE TO LACK OF BUFFER SPACE.

THE SECOND OPTION THAT WAS DISCUSSED RELATED TO THE CURRENT DEVELOPMENT WITHIN HONEYWELL FOR AN INTELLIGENT CONTROLLER THAT WILL MEET THEIR LOCAL AREA NETWORK DESIGN OBJECTIVES. THIS CONTROLLER IS A MOTOROLA 68000 BASED CONTROLLER WITH MEMORY EXPANSION CAPABILITY TO 512K, EQUALLY DIVIDED BETWEEN PROGRAM AND DATA SPACE. THE CONCEPT EMPLOYS A MOTHER/DAUGHTER BOARD APPROACH. THE DAUGHTER BOARD WILL PERFORM THE MEDIA ACCES FUNCTION. THEIR FIRST LOCAL AREA NETWORK WILL USE THE IEEE TOKEN BUS MEDIA ACCESS CONTROL. FOR ULTIMATE TO USE THIS BOARD FOR ITS WIDE AREA NETWORK SOLUTION A NEW DAUGHTER BOARD WILL HAVE TO BE DESIGNED AND DEVELOPED. DICK LEMAY WILL RESPOND WITH A PROPOSAL TO DO THIS.

BOTTOM LINE.... LETS WAIT FOR THE PROPOSAL, BUT I AM INCLINED TO RECOMMEND THIS APPROACH EVEN WITHOUT SEEING THE COST FOR DEVELOPMENT SINCE IT CLEARLY WILL SATISFY OUR REQUIREMENTS.

IN ADDITION WE WILL BE ABLE TO USE THE LAN BOARDS THEY DEVELOP, THEREBY MINIMIZING THE NUMBER OF DIFFERENT BOARDS WE WOULD HAVE TO SUPPORT, AND ALSO OBTAIN LEVERAGE FROM THE DEVELOPMENT THAT HONEYWELL WILL DO TO ACCOMODATE THE LAN PROBLEM. IN ADDITION UNIT COST WILL DECREASE BY USE OF A BOARD THAT WILL SEE LARGE VOLUMN WITHIN HONEYWELL.

UPON RECEIPT OF THEIR PROPOSAL I AM PREPARED TO GO BACK TO HONEYWELL, NEW JERSEY, OR WHERE EVER ELSE IT TAKES TO GET THIS PROJECT AGREED TO AND UNDER WAY. I BELIEVE THAT WE NEED A DECISION ON THIS BY THE END OF MAY SO THAT WE MAY SOLICIT A QUOTE TO DO A SIMILAR BOARD FOR THE DEC PRODUCT LINE

H O N E Y W E L L Interoffice Correspondence RL:83:28

Date: April 20, 1983

Subject: THIRD ULTINET MEETING

To: F. Gilfeather

From: R.Lemay
Organization: C&SP
HED: MA32
MS: 999
HVN: 275-7580

cc: J. Conway
R. Farrell
A. Hirtle
J. Neumann
L. Niessen
E. Spiewak
B. Tymann
J. Vernon

This report describes the results of the third joint Ultimate/Honeywell meeting which occurred on April 13-14, 1983. The purpose of the meeting was to explore the NMLC as a proper hardware foundation for Ultimate's networking product.

Using standard hardware, Ultimate is developing a product suitable for interfacing Ultimate systems to Value Added networks more or less as described in the minutes of the Second Ultinet Meeting (RL:82:96).

Ultimate intends to market a product for private networks which has the following characteristics:

- First delivery 4Q84
- ISO compliant
- Physical, link, network, & transport fully implemented outside CPU.

During the meeting the NMLC hardware was explored to determine if it has the horsepower and private storage to implement the first four levels. The conclusion reached was that it had enough horsepower but not enough storage.

A presentation of the LAN controller followed. This mother board has sufficient horsepower and memory to do the job and has fewer restrictions regarding the use of its resources than does the NMLC. It further would permit Ultimate to avoid any LAN hardware development work.

The conclusion reached is that Ultimate will fund a daughter board design for the LAN controller housing four RS-232 synchronous ports and a maximum RS-232 transfer rate of 19.2K baud. John Neumann returns to Ultimate with the intent of selling Ultimate management on such a proposal.

If the proposal is accepted, the following rough schedule should provide a callibration as to the intent of the proposal:

FOUR PORT DAUGHTER BOARD SCHEDULE

• Start specification	Jun '83
• Complete specification	Aug '83
• Start hardware checkout	Jan '84
• Ship prototype to Ultimate for s/w checkout	Jun '84

ISSUES

- Do we have in Honeywell somewhere a 68000 Real-Time Operating System?
- If we do, can we sublicense it to Ultimate?
- What shipping criteria do we use to determine when Honeywell can ship the prototype to Santa Ana for software checkout?
- Who codes what ISO levels?

First proposal:

PHYSICAL	HONEYWELL
LINK	HONEYWELL
NETWORK	ULTIMATE
TRANSPORT	ULTIMATE

Richard A. Lemay
Manager, Special OEM Products
Custom & Special Products

/cg

Rcvd 5-4-83

LIST OF ITEMS TO DISCUSS WITH DAVE GOULD ON MAY 5TH.

1. MARKETING REACTION TO ARCHITECTURE DEFINITION

2. MARKETING REACTION TO IMPLEMENTATION PLAN

3. MARKETING REQUIREMENTS FOR PHASE 1 OF PLAN
   - A. X.25 CONNECTIVITY (ZBK VS. MICOM)
   - B. LAN (MULTIPLE VS. DOUBLE, SPEED)

4. PHASE TWO HARDWARE
   - A. HONEYWELL
   - B. DEC
   - C. OTHER (IE: IBM PC)

5. SALES ISSUES
   - A. FORTUNE COMPANIES
   - B. SUPPORT SOFTWARE (IE. COBOL)
   - C. PRODUCT SUPPORT
   - D. DIRECT VS. DEALER SALES
   - E. MARKETING EDUCATION
   - F. SALES SUPPORT

To: Carl Margolis

From: John Neumann John

Subject: Dealer Survey - Results?

Date: August 17, 1983

CC: Frank Kacerak
Dave Gould

I have just completed a summary of the dealer questionaires received to date, of which there are 19. I will not attempt to interpret the results, but rather provide the data for interpretation by others more qualified than myself.

Part I - Current Business

The purpose of Part I was to obtain a perspective of the current business our dealers are involved in. The questions dealt with the current customer base, types of applications, and how those applications are developed/procurred. In this part, as in the the others, an attempt has been made to differentiate between the Honeywell based systems vs. the DEC based systems. The results will be detailed with this objective in mind.

1. Number of Stand-alone systems sold?

   Honeywell - 107 (41%)

   DEC - 154 (59%)

2. Number of Multi-machine Installations sold?

   Honeywell - 22 (21% of #1 above)

   DEC - 21 (14% of #1 above)

2a. Single Location installation (LAN)

   Honeywell - 19 (18% of #1 above)

   DEC - 9 (6% of #1 above)

2b. Multiple Location installation (Remote)

   Honeywell - 13 (12% of #1 above)

   DEC - 13 (8% of #1 above)

3. Customers expected to add systems in one location

   Honeywell - 16 (36%)

   DEC - 28 (64%)

4. Customers expected to add systems in remote location

Honeywell - 17 (35%)

DEC - 31 (65%)

5. Customer inquiries about Data Communications

Honeywell - 90 (61%)

DEC - 57 (39%)

6. Types of Software Packages sold?

Distribution
Manufacturing
General Business
Custom
Inventory Control
Point of sale
Records Management
Schedule and control
Financial Institutions
Accounting
Municipal Government
Library Systems
Medical/Dental
Small Business Applications
Insurance
Utilities
Collection Agency
Medical Laboratory
Hotel
MDS
Office Products
Brokerage
Social Services

7. Source of Software Packages

In-house Development - 19 (100%)

Separate Software Company - 10 (43%)

Open Market - 9 (39%)

Part II - Customer Prospect List

The purpose of this part of the survey was to obtain information from the dealers about their prospective customers' Data Communications requirements. This information could provide Ultimate with a picture of the short term needs as perceived by the dealers in relation to their active new sales efforts.

1. Prospects desiring multiple systems

1a. in one location (LAN)

Honeywell - 60 (41%)

DEC - 87 (59%)

1b. in more than one location (REMOTE)

Honeywell - 46 (37%)

DEC - 77 (63%)

This next question provides some interesting data. The intent of the question was to focus on specific network configurations that prospective customers have asked about. I will try to provide summary data after presenting the raw data that I think might be meaningful.

2. Prospects who have inquired about

2a. Local Area Network

Honeywell - 88 (55%)

DEC - 71 (45%)

2b. Ultimate to Ultimate

Honeywell - 99 (47%)

DEC - 114 (53%)

2c. X.25 TELENET

Honeywell - 95 (56%)

DEC - 75 (44%)

2d. 3270

Honeywell - 87 (55%)

DEC - 70 (45%)

2e. SDLC

Honeywell - 58 (50%)

DEC - 57 (50%)

2f. SNA

Honeywell - 70 (57%)

DEC - 53 (43%)

Of the total inquiries, 937, they were divided as follows:

Honeywell - 497 (53%)

DEC - 440 (47%)

and when Ultimate to Ultimate configurations (2a,-2b, and 2c)

are compared to IBM Connectivity (2d, 2e, and 2f) the following statistics are found:

Ultimate to Ultimate - 542 (58%)

Honeywell - 282 (52%)

DEC - 260 (48%)

IBM Connectivity - 395 (42%)

Honeywell - 215 (54%)

DEC - 180 (46%)

The following question should produce relative priority of our development thrust.

3. System sales lost due to lack of

3a. Local Area Network - 18 (20%)

Honeywell - 11 (61%)

DEC - 7 (39%)

3b. Ultimate to Ultimate - 23 (25%)

Honeywell - 9 (39%)

DEC - 14 (61%)

3c. X.25 TELENET - 20 (22%)

Honeywell - 14 (70%)

DEC - 6 (30%)

3d. 3270 - 21 (23%)

Honeywell - 17 (81%)

DEC - 4 (19%)

3e. SDLC (SNA?) - 9 (10%)

Honeywell - 6 (67%)

DEC - 3 (33%)

Of the lost sales, they are distributed between

Honeywell - 57 (63%)

DEC - 34 (37%)

and may additionally be compared as follows:

Ultimate to Ultimate - 61 (67%)

Honeywell - 34 (56%)

DEC - 27 (44%)

IBM Connectivity - 30 (33%)

Honeywell - 23 (77%)

DEC - 7 (23%)

Part III - Projections and Requirements

This Part of the survey was intended to give the dealer an opportunity to project his needs in terms of types of configurations needed, relative priority of development, and support requested from Ultimate in the new marketplace.

1. If you had on-line interactive remote file access capability, how many systems could you sell:

1a. Local Area Network - 95 (31%)

Honeywell - 38 (40%)

DEC - 57 (60%)

1b. Ultimate to Ultimate - 98 (32%)

Honeywell - 34 (35%)

DEC - 64 (65%)

1c. X.25 - 113 (37%)

Honeywell - 37 (33%)

DEC - 76 (67%)

Of the projected sales they were

Honeywell - 109 (36%)

DEC - 197 (64%)

and of the Honeywell systems they were configured

LAN - 38 (35%)

REMOTE - 34 (31%)

X.25 - 37 (34%)

and of the DEC systems they were configured

LAN - 57 (29%)

REMOTE - 64 (32%)

X.25 - 76 (39%)

2. Relative priority for development

LAN - 2.33

REMOTE - 2.16

X.25 - 2.27

Non-Ultimate- 2.77

This would indicate the priority of communications needed by the dealers as:

Ultimate to Ultimate

X.25

LAN

Ultimate to Non-Ultimate

3. Relative priority of Functional Capabilities

Remote File Access - 1.4

Remote Logon - 2.0

Remote Printer/Tape Access- 3.6

Remote Job Execution - 2.6

This would indicate the priority of functional capability needed by the dealers as:

Remote File Access and Transfer

Remote Logon

Remote Job Execution

Remote Printer/Tape Access

4. More Information desired on communications

Basic Data Communications - 13

Local Area Networks - 15

X.25 TELENET - 14

other - 1

5. Dealer Interest in

Seminar on Data Communications - 11

Marketing/Sales support - 16

6. Will Networking give you an edge over the competition?

This question invoked mixed reactions from the dealers. The majority of dealers felt that Networking would give them an edge over the competition, while some thought that it would only provide parity.

Edge - 14

Parity - 4

No Response - 1

7. Other Comments and suggestions:

"The requirements I see are for a number of micro's (PC's) in user departments executing one function with ability to log onto host, access files, and transfer batch data" Anacomp, Inc.

"High speed LAN is very important. Easy interface to PC like system is important" Calnek Price

"We have been forced to sell Prime solutions in place of Ultimate because of Lack of Communications. Also would like to see ASAP office Automation product similar to Honeywell (Which I consider excellent)" General Computing Services

"3270 capability is critically important, also transaction processing 3780 interface" Ultradata Corp.

"I am not knowledgeable enough to respond well to this request. Suggest any textbook or other source that would help" Ultimate Intermountain.

"The use of a Background Processor may be useful. It could initiate a job, free up the terminal for other work, and continue processing the job in the background mode. Stacking such jobs would be very helpful for large installations" Area Computer Services, Inc.

"I can't use the Ultimate product line for my main market thrust until I have 1) Remote File Access, 2) Remote Logon, and 3) Ultimate to non-Ultimate" Lab Force, Inc.

"Will all components be supported through Honeywell field service? Can image campaign for large customers for whom networking is a must be considered?" Systems House, Inc.

"Networking is particularly important in fortune 1000 type sales. Will become a large benefit when combined with the 32-bit machines" Ultimate-Southern California.

8. Anticipated Sales Increase when Networking available

$8,3000,000.00

122 Systems

10% to 75% (Depending on Dealer)

Three Dealers did not respond, while Two Dealers could not

estimate the increase.

To: Dave Gould

From: John Neumann

Date: November 9, 1983

Subj: Visit to Calcomp

CC: Carl Margolis
Frank Kacerak
Karl Gates
Rich Lauer
Dennis Auler

Attachments: Calcomp Configurations

On November 2, 1983, I visited California Computers, Inc. (Calcomp), in Anaheim, California. Calcomp is a large user of the Prime Information System, and in particular, Primenet. The System configuration used by Calcomp is depicted in Figure 1 attached. Figure 2 shows the system when their expansion plans are complete. As a guide for my discussion with Calcomp, I used the Checklist previously generated for discussions with Primenet users.

I want to thank Rich Lauer for arranging the meeting between myself and Roy Young, Director of Information Systems and Rex Tompkins, Manager of Data Processing Administration. Both gentlemen were most cooperative, and gave me a tour of their operations at the end of the discussions. The time I spent at Calcomp was approximately 1 hour. The information I obtained during the discussion is presented below.

Motivation for using Primenet:

Calcomp was looking for an application solution when buying their Management Information System. The major problem they were solving was oriented towards data entry. After looking at several possibilities, they settled in on the Pick Operating System. This led them to Microdata, where they discovered a lack of Communications on the Reality System. Because of their configuration requirements, discussed next, they turned to Prime's Information System, and Primenet.

Calcomp Configuration:

Figure 1 graphically depicts the current, and original, system configuration employed by Calcomp to solve their data processing problem. Figure 2 depicts the system configuration after expansion plans are complete.

Operational Use of System:

As shown in the configuration, all 250-300 terminals are

connected to a Front End Processor Switch. The Rixon 850. When the user wishes to 'Logon' to a system, he must specify which system he is logging on to. Only a handful of the terminals are direct connected to the Prime system, the remaining terminals may use any one of the three systems in Anaheim. Users are not allowed to logon to the New Hampshire system. A user may be logged on to one system at a time.

The Primenet link to New Hampshire is 9600 Baud and is used only for Administrative control and nightly File Transfers. No interactive traffic (except admin) is on the Primenet link. The terminals run at 2400 Baud to the Rixon Switch. It is felt that the Primenet link is too slow for interactive traffic.

They are using the Rixon Switch as a form of Local Area Network. In New Hampshire, they have an Ungerman-Bass Net One installed (Ethernet compatible) for the Local Area Network.

Files are duplicated on the three Prime systems in Anaheim as they have no file transfer capability. Files are also duplicated between New hampshire and Anaheim in order to avoid using Primenet as much as possible, which they consider too slow.

There is no Security within the Prime Information System other than normally associated with Pick. User security is controlled by their application. The application is menu driven for each user, and the user is limited in what he can do or access by the menu.

Network Management:

Primenet has a Network Control Facility which can be used to externally control the network. The control function is distributed in the sense that any node can operate as the Net Manager. They use this capability only for back-up, since the actual control is centralized in Anaheim. The Network provides no statistical data for use by the manager. They are not even able to obtain raw data from the network pertaining to utilization and load. Right now all they can do is guess as to how the network is functioning to meet their requirements.

Configuration changes require parameter changes in the Network configuration tables. They state that this operation is relatively simple and straight forward.

Desired Capabilities:

There were two things that Calcomp said that they wanted, but currently did not have, in their system. The first thing mentioned was the need to be able to get useful statistical information from the network. As mentioned earlier, they must currently experiment with various configuration parameters to optimize the system. Even at that, they do not know whether they have found the proper configuration yet. The second capability they expressed a desire for was a dedicated file server accessable from multiple CPU's to avoid the

current mode of duplicating files on the systems. They expressed concern that access to the file server must be at channel speed, or they would see little benefit.

One final observation, they believed that the Prime implementation of Primenet was slow and need not be.

Follow-on:

Roy Young indicated a desire to maintain contact with Ultimate regarding our implementation and to be able to contact me regarding the specification of their future expansion plans. I indicated a willingness to do this on behalf of Ultimate Corporation.

Figure 1: Current Calcomp Network Configuration

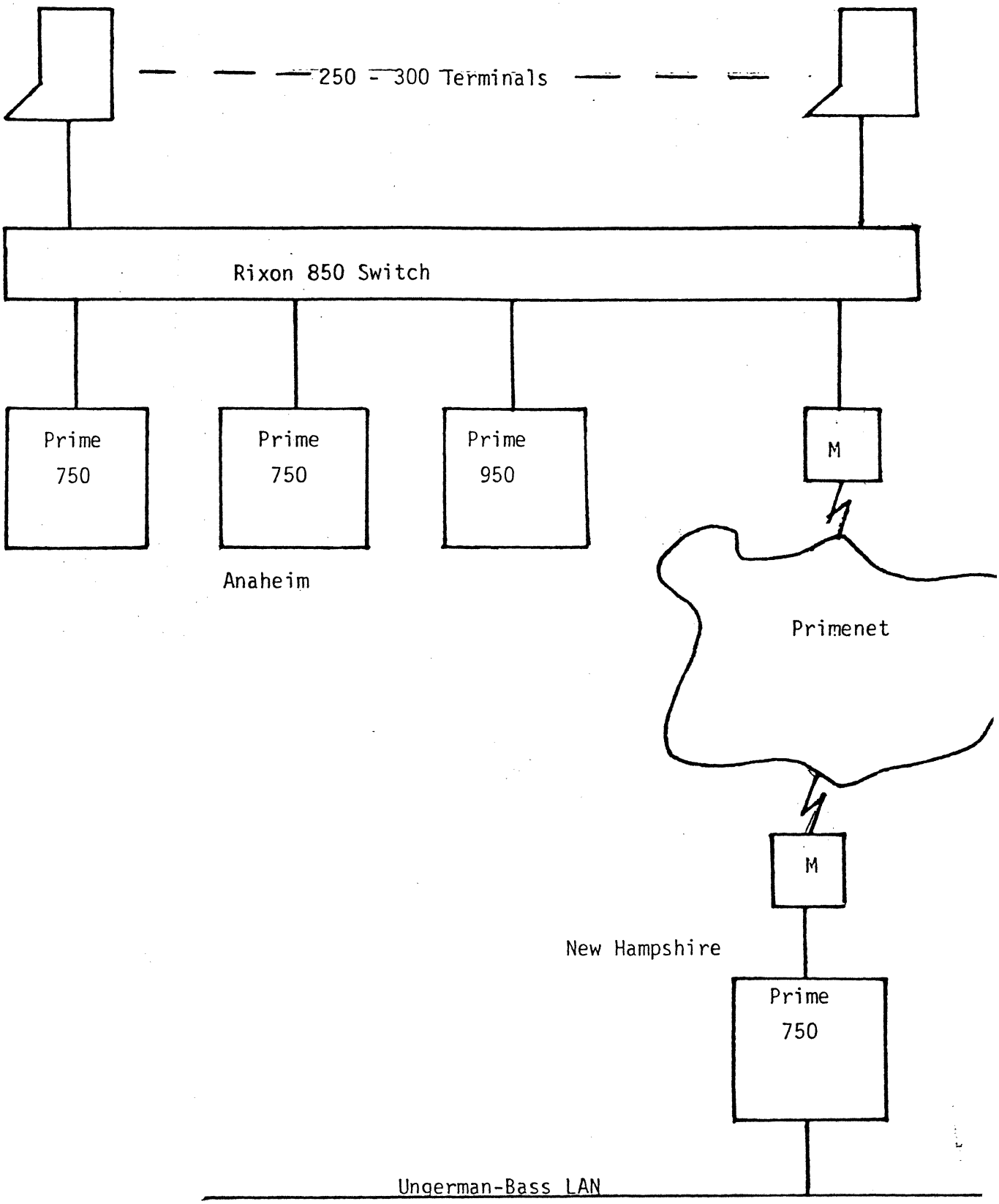
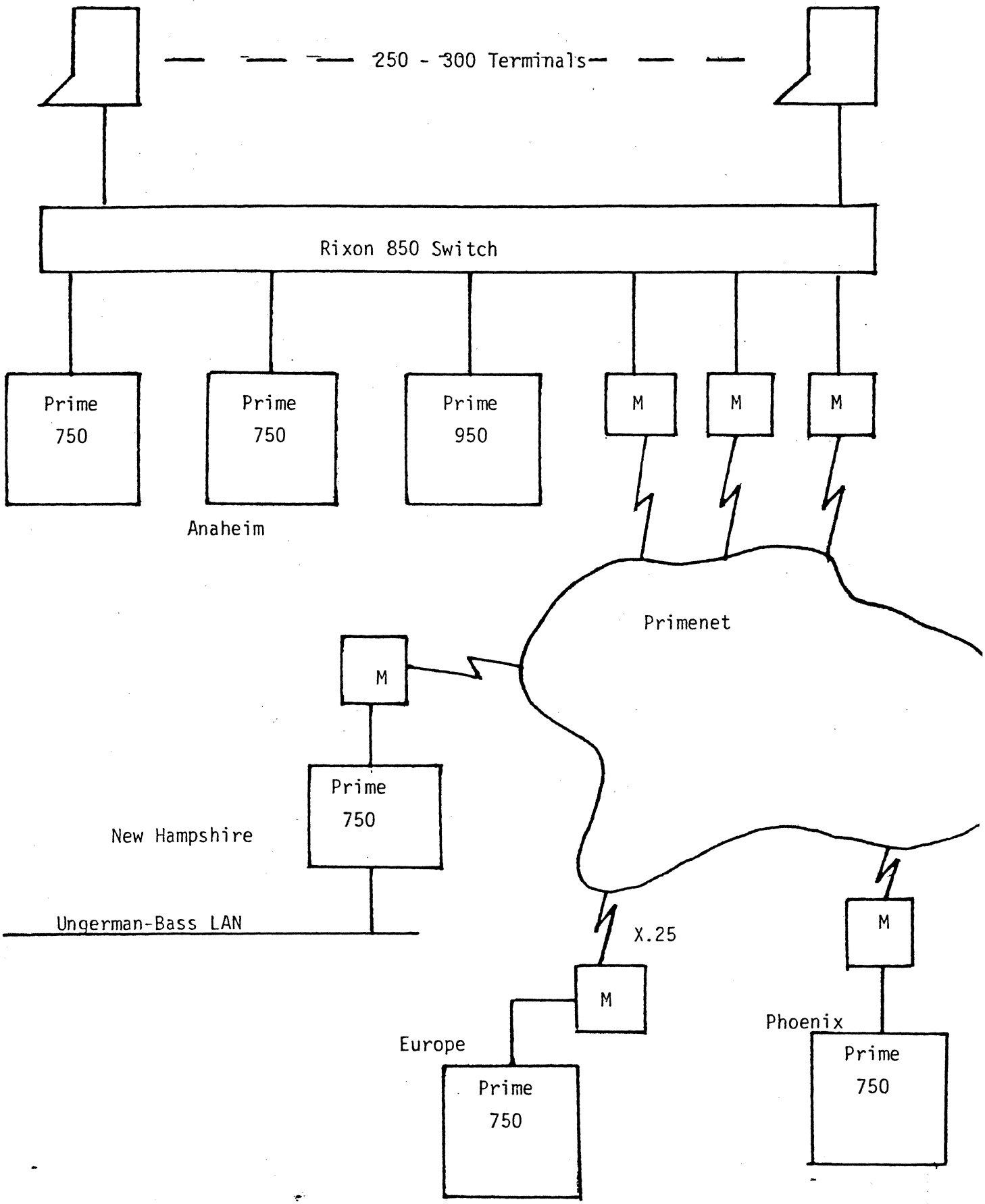


Figure 2: Planned Calcomp Network Configuration



Rcvd 2/22/83

PHASE TWO SCHEDULE

THIS DOCUMENT REPRESENTS A PROPOSED SCHEDULE FOR IMPLEMENTATION OF THE PHASE TWO DEVELOPMENT TASKS. WHERE INTERDEPENDENCIES EXIST THEY ARE IDENTIFIED.

TASK NUMBER	TITLE	START	COMPLETE
2.2.1	SYMETRICAL HDLC	7/83	6/84
2.2.2	X.25 PAKET LEVEL	7/83	6/84
2.2.3	TRANSPORT PROTOCOL	7/83	9/84
2.2.4	SESSION PROTOCOL	7/83	9/84
2.2.5	REMOTE LOGON	1/84	10/84
2.2.6	RESOURCE MANAGER	1/84	10/84
2.2.7	COMMON BUFFER/QUEUE	1/84	10/84
2.2.8	NETWORK DIAGNOSTICS	7/83	9/84
2.2.9	NETWORK MANAGEMENT	9/83	11/84
2.2.10	INTELLIGENT CONTROLLERS	6/83	4/84
2.2.11	TEST AND INTEGRATION	10/84	1/85

TASKS 2.2.1, 2.2.2, AND 2.2.3 REQUIRE TASK 2.2.10 TO BE COMPLETED ON SCHEDULE IN ORDER TO COMPLETE.

IT IS POSSIBLE TO BEGIN TASKS 2.2.1 AND 2.2.2 CONCURRENT WITH 2.2.10 IF THE WORK IS PERFORMED BY CONSULTANTS. TASK 2.2.4 COULD ALSO BE DEVELOPED BY OUTSIDE CONSULTING, AS CAN TASK 2.2.8.

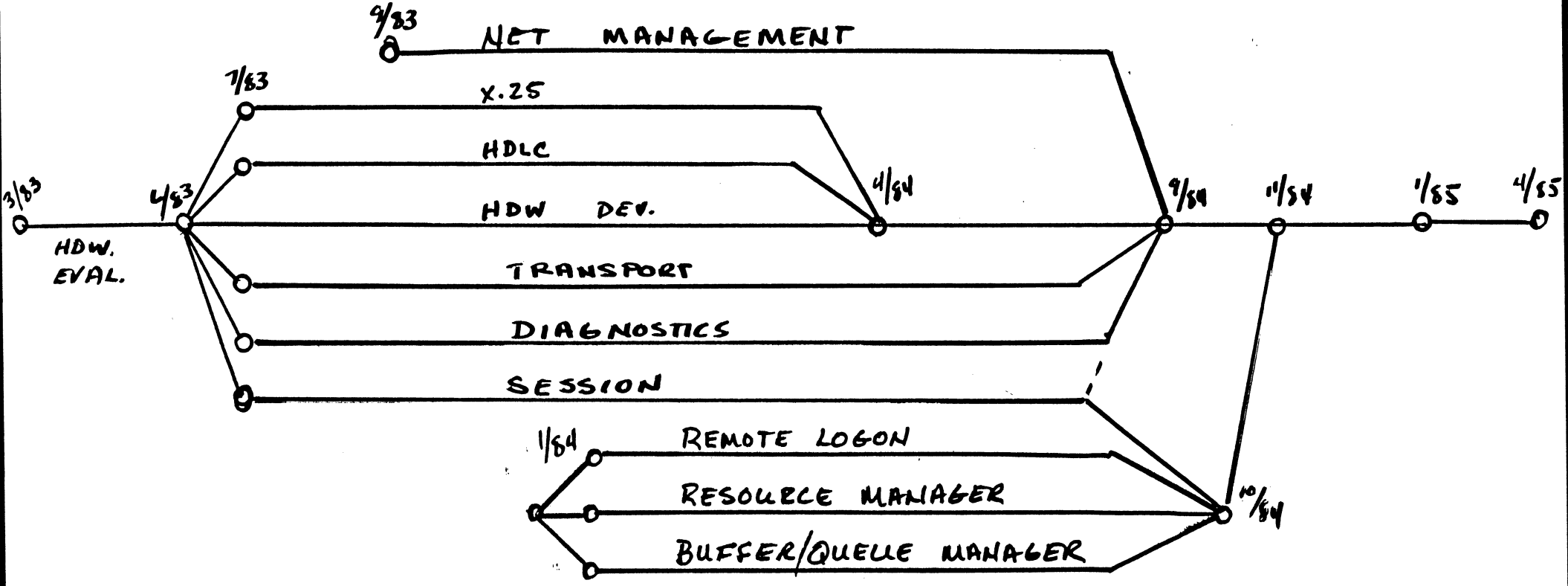
TASKS SCHEDULED TO BEGIN ON 1/84 ARE TIMED SUCH THAT THE PHASE ONE STAFF CAN MOVE INTO PHASE TWO.

THE MAJOR CONSTRAINT ON THE PROJECT IS RESOLUTION OF THE INTELLIGENT CONTROLLER ISSUE WITH HONEYWELL AS SOON AS POSSIBLE.

BETA TESTING SHOULD BE ABLE TO BEGIN DURING 1/85 WITH CUSTOMER SHIPMENTS BEGINING 4/85. BETA TESTING WITH PHASE ONE PRODUCT CAN BE COMPLETED DURING THE SAME TIME PERIOD.

PHASE THREE COULD BEGIN DURING 5/85 AND AN ESTIMATED COMPLETION OF 12/86.

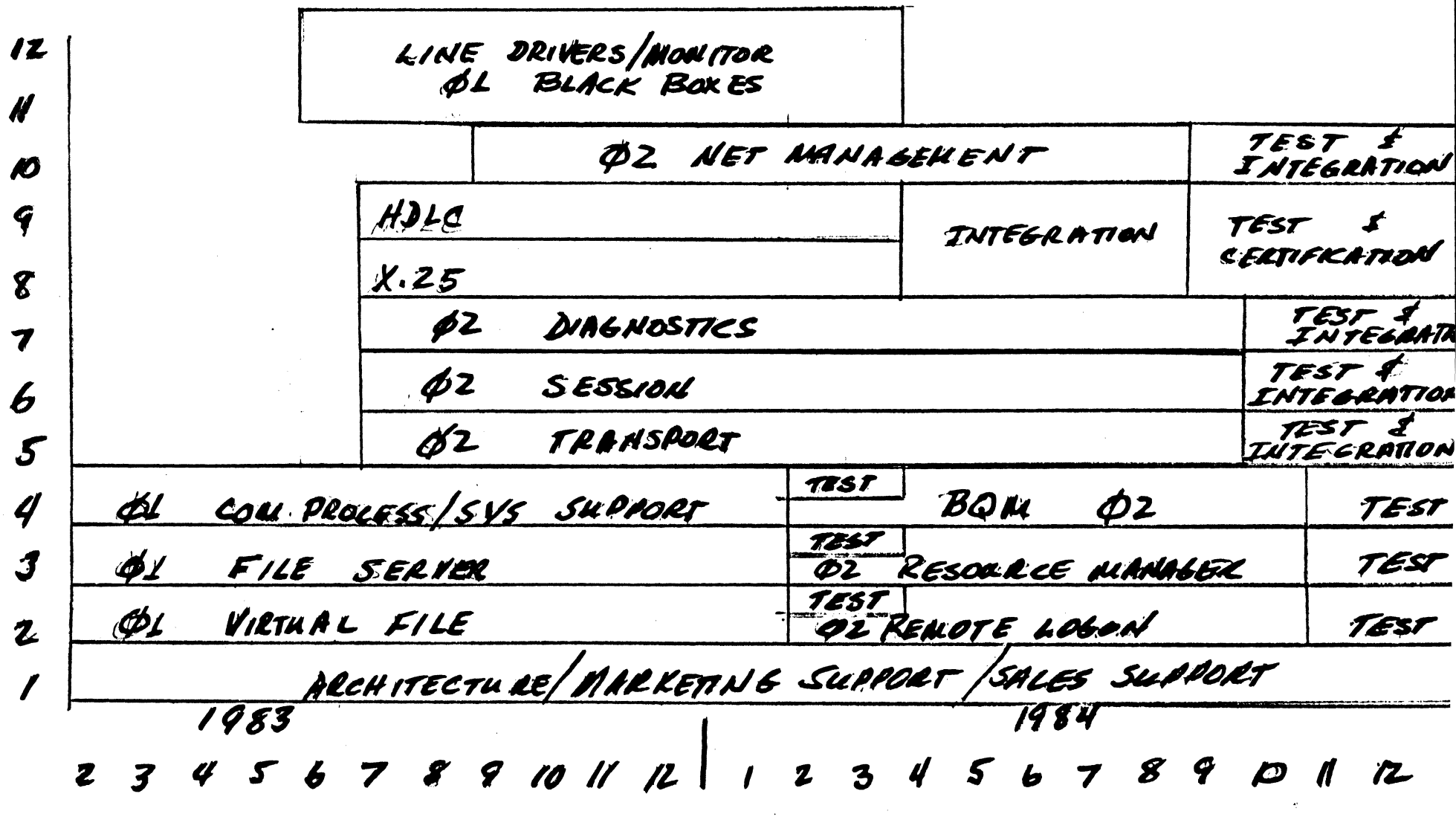
THIS SCHEDULE IS PRELIMINARY AND REPRESENTS CURRENT UNDERSTANDING OF THE EFFORT INVOLVED TO DEVELOP PHASE TWO.

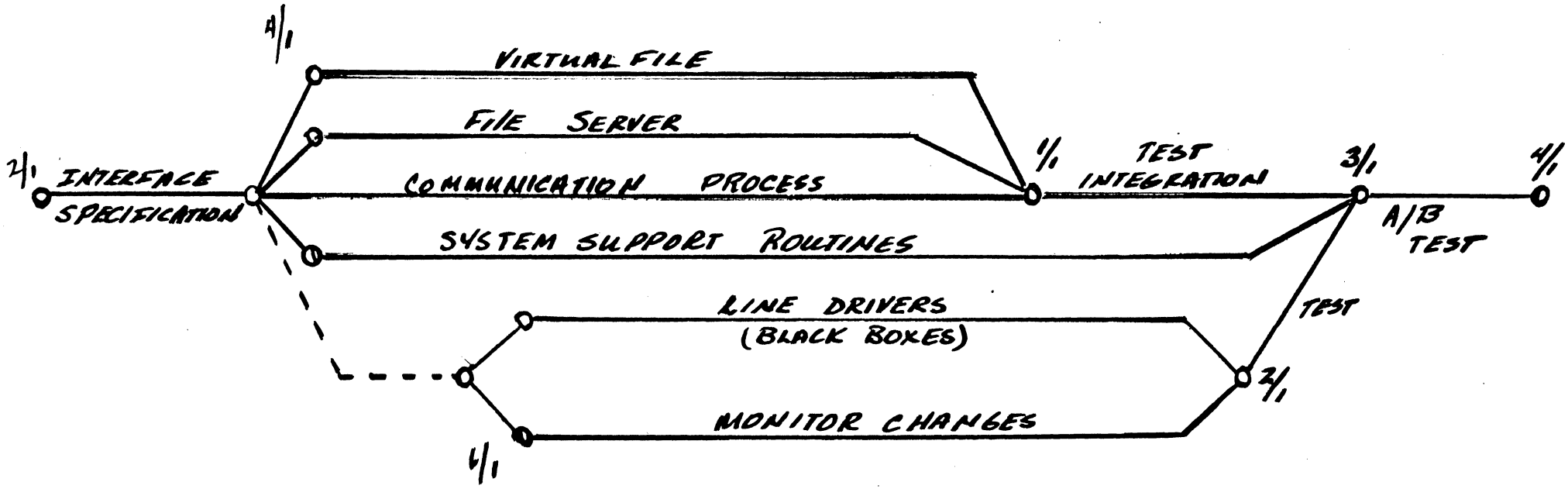


IMPLEMENTATION PHASES:

NETWORK	DESIGN	IMPLEMENTATION	INTEGRATION	A TEST	B TEST

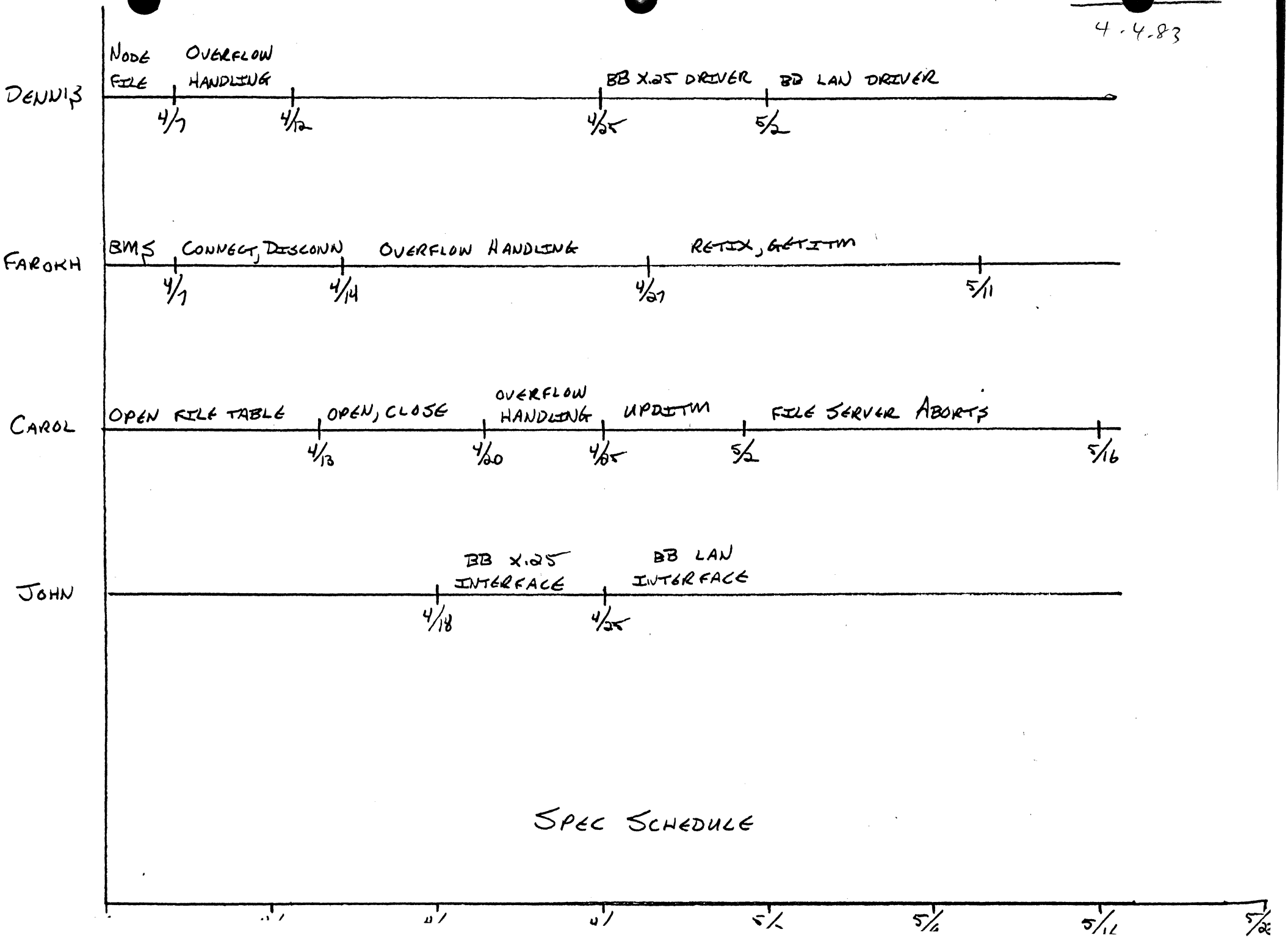
2/24/83


12
11
10
9
8
7
6
5
4
3
2
1
LINE DRIVERS/MONITOR
ØL BLACK BOXES
Ø2 NET MANAGEMENT
TEST & INTEGRATION
HDLC
X.25
INTEGRATION
TEST & CERTIFICATION
Ø2 DIAGNOSTICS
TEST & INTEGRATI
Ø2 SESSION
TEST & INTEGRATION
Ø2 TRANSPORT
TEST & INTEGRATION
ØL COM. PROCESS/SYS SUPPORT
TEST
BQM Ø2
TEST
ØL FILE SERVER
TEST
Ø2 RESOURCE MANAGER
TEST
ØL VIRTUAL FILE
TEST
Ø2 REMOTE LOGON
TEST
ARCHITECTURE/MARKETING SUPPORT/SALES SUPPORT
1983
1984
2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12


A/1
VIRTUAL FILE
FILE SERVER
2/1
INTERFACE
SPECIFICATION
COMMUNICATION PROCESS
1/1
TEST
INTEGRATION
3/1
4/1
A/B
TEST
SYSTEM SUPPORT ROUTINES
LINE DRIVERS
(BLACK BOXES)
TEST
2/1
MONITOR CHANGES
4/1

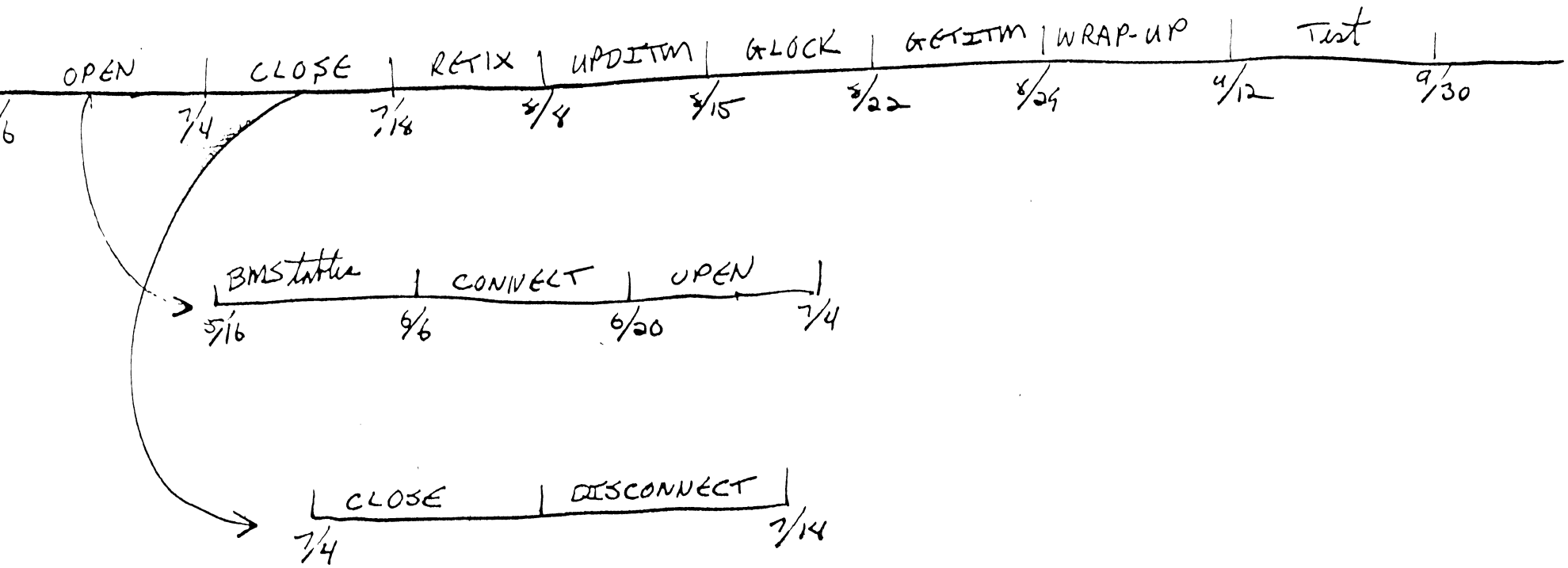
Rev 4/4/83

4.4.83

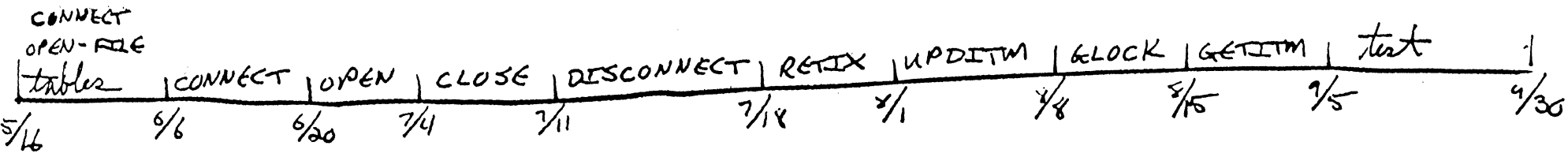


Virtual Pr ss

5-11-83



File Server



5-11-85

2-7-83

USER LEVEL DOCUMENTATION FOR THE ULTIMATE NETWORK

The Ultimate communications network capability allows users on one computer to access files on another computer as if the files were on his own computer. Remote file access (that is, access to files on another computer) can be done using four file interface languages: Recall, BASIC, TCL-II verbs, and assembler subroutines. Item locks have been added to facilitate network file access. Several methods of physical connection between the computers are possible: local area network, private line network, and value added carrier network.

File access on a local machine is done with "D" pointers in the master dictionary for the account or with "Q" pointers. A file can be defined on a remote machine using an extension of the "Q" pointer which, in addition to giving the account name and file name, gives the computer name on which the file resides. The general format of the remote file definition is:

file-name

1. Q
2. account-name
3. file-name
4. node-name

The node-name can be any arbitrary string of alphanumeric characters, up to fifty characters long. Examples could be "DALLAS-STORE1" or "HEADQUARTERS-ACCOUNTING". Upon file open, the system takes the node-name from the file's Q definition and converts it to the network address assigned at network generation.

At the command level, the network file transfer capabilities of the system can be invoked using TCL-II verbs and Recall commands. When invoked, verbs at this level will transfer the number of items specified, in their entirety, from one computer to another.

Two examples of TCL-II verbs are EDIT and COPY. To edit an item in a remote file, the user would enter

```
>ED REMOTE-FILE-NAME ITEM-LIST
```

Entering this command causes a logical connection to be made to the remote computer and the first item in the item-list to be transmitted to the local computer. After the item has been transferred to the user's workspace, it can be edited in the usual way. After editing, the item can be exited, in which case it is discarded, or it can be filed, in which case it is transmitted back to the original computer and updates the old copy of the item. Processing then continues with the next item in the list, continuing until the item list is exhausted.

The copy command with remote files can have several different formats depending on where the source and destination files are.

```
>COPY REMOTE-FILE-NAME ITEM-LIST
TO: LOCAL-FILE-NAME
```

Execution of this command causes an logical connection to the remote computer to be established. The items in the list are then transferred from REMOTE-FILE-NAME to the local computer and stored in LOCAL-FILE-NAME.

```
>COPY LOCAL-FILE-NAME ITEM-LIST
TO: REMOTE-FILE-NAME
```

Execution of this command causes a logical connection to the remote computer to be formed. The items in the list are selected from the LOCAL-FILE-NAME and transferred from the local computer to the remote computer and stored in REMOTE-FILE-NAME.

```
>COPY REMOTE-FILE-NAME1 ITEM-LIST
TO: REMOTE-FILE-NAME2
```

Assuming that REMOTE-FILE-NAME1 and REMOTE-FILE-NAME2 exist on different computers, execution of this command causes logical connections to be formed between the local computer and remote computer 1 and between the local computer and remote computer 2. The item list is selected from REMOTE-FILE-NAME1 and is transferred to the local computer. As each item is received by the local computer, it is transferred to remote computer 2 and stored in REMOTE-FILE-NAME2.

As can be inferred from the above example, the file items will always be on the local computer sometime during the transfer phase. This fact must be considered when operating on a file, since there is the possibility of using the net-

work very inefficiently, especially in the use of Recall commands. Some examples of Recall commands and their peculiarities while working on remote files follow.

The LIST command can have the following format:

```
>LIST REMOTE-FILE-NAME 'ITEM-NAME1''ITEM-NAME2'
```

Execution of this command causes a logical connection to the remote computer to be formed. The items ITEM-NAME1 and ITEM-NAME2 are retrieved from REMOTE-FILE-NAME and transmitted to the local computer, where the item-names will be displayed on the local terminal if the items have been found.

Recall commands can also use listing or selection criteria when operating on a remote file. However, since the remote file's dictionary is not retrieved, the attribute definitions must be on the local computer.

```
>LIST REMOTE-FILE-NAME 'ITEM-LIST' NAME ADDRESS WITH COST >
"500"
```

After the logical connection is made, the items in the item-list are transferred to the local computer, and those that meet the selection criteria will have the item-name along with the NAME and ADDRESS fields displayed to the terminal. The NAME, ADDRESS, and COST attribute definitions must be contained in the local account's master dictionary.

Another variant of the LIST command is

>LIST REMOTE-FILE-NAME 'ITEM-LIST' USING DICT LOCAL-FILE-NAME

After the logical connection is made, the items in the item-list are transferred to the local computer and displayed on the terminal using the attribute definitions in the dictionary of LOCAL-FILE-NAME.

An example of inefficient use of Recall commands on the network is

```
>SSELECT REMOTE-FILE-NAME
1487 ITEMS SELECTED
>LIST REMOTE-FILE-NAME
```

After the logical connection to the remote computer is made, the entire REMOTE-FILE-NAME file is transferred to the local computer, a sort is done on it, and a SELECT list is created. The LIST command is then executed using the SELECT list just formed. Every item in REMOTE-FILE-NAME is again transferred from the remote computer to the local computer in the order of the SELECT list and displayed on the terminal. This example is inefficient in two ways. First, even though the sort in the selection phase is acting only on the item-IDs of the remote file, the items are retrieved from the remote file in their entirety. Secondly, after the SELECT list is formed, the entire item is again retrieved from the remote computer for the LIST command.

BASIC allows the user to open and access files on a remote computer as if they resided on his local computer. The added networking capability is transparent to the BASIC programmer with two minor exceptions: the function of the READU instruction changes slightly and a CLOSE file instruction has been added.

The READU instruction no longer locks the entire file group that contains the item, but instead locks only the item itself. Item-locks allow other users to access other items that lie in the same group. Previously, if the user tried to READU an item in a group locked by someone else, he was put to sleep until the other user unlocked the group, even though the two users were concerned with two different items. Now one user will not prevent another user from simultaneously reading a different item from the same file group that he is reading. If, however, the two users are attempting to read the same item, the second user is put to sleep until the user who locked the item unlocks it with a WRITE statement.

OPENing a file that lies on a remote computer will make a logical connection between the local computer and the remote computer that contains the remote file. This logical connection remains open until the file is closed by ending the BASIC program or by executing a CLOSE statement on the file. If the CLOSE statement is executed on a local file it is treated as a NOP. If the CLOSE statement is executed on a remote file, the remote computer is informed that the file is no longer needed by the local computer and the logical

connection between the two computers is broken. Since only a limited number of logical connections can be opened on any one computer, files should be closed when they are no longer needed.

Accessing files in the network environment using assembly language, with the exception of the addition of a few assembly language instructions and some side effects that should be transparent to the user, is no different than the normal access of local files. The network access mechanisms are in system subroutines such as RETIX and UPDITM, and if these routines are used in the normal manner the assembly language programmer need not be aware that he is accesssing remote files.

The system subroutines that access files are OPEN, RETIX (and its variations), GETITM, UPDITM, and GUNLOCK. New system subroutines that have been added are RETIXI and IUNLOCK.

The OPEN command, when operating on a remote file, detects from the file definition item, the "Q" pointer, that the file is remote and determines the network address that was assigned to the remote computer at network generation time. OPEN then sends a request to the remote computer to open the required file. If the file can be opened, the remote sends a confirmation message back to the local computer. After opening the file, OPEN returns in the BMS area the network address of the file, a code used by the remote computer that it uses to identify the opened file, and an indicator that the file is remote. The fact that BMS no longer contains the actual base, modulo, and separation of a file is one of the side effects of accesssing a remote file. User code can no longer manipulate the BMS and expect to get valid results. The system subroutines that use BMS, such as RETIX, react to

the remote file indication in the BMS and access the remote computer addressed in the BMS for file retrieval.

RETIX, GETITM, and UPDITM have the same input and output interfaces as previously with the exception of the BMS being in a special format as described above. RETIX has the side effect of having the retrieved remote item copied to overflow space and having the output parameters pointing to the item in overflow space instead of pointing into the actual file space, which, of course, would be pointless. The first time in, GETITM retrieves a whole group from the remote file and copies it into overflow space on the local computer. Each execution of GETITM returns pointers to an item in the copied group. When all the items in the group have been retrieved, execution of GETITM causes another group in the file to be retrieved from the remote computer and copied into the local computer's overflow space, overlaying the previously retrieved group. UPDITM works as before, with the item that was built in the user's workspace being transmitted to the remote computer and then being updated to the remote file.

Group locks are handled the same as previously for local files. If the locked group is in a remote file, an entry in the group lock table is created in both the local and remote computers. After the item is retrieved from the remote computer, it is copied into overflow space on the local computer and the first frame of the linked overflow space that contains the item is put in RECORD and is entered in the local group lock table. When the item is retrieved from the remote computer, the group that the item lies in on the

remote computer is locked and the base FID of that group is passed back to the local computer. This group base FID from the remote computer is placed in the same group lock table entry as the overflow frame FID, as described above, that is the beginning of the copied item. All of this is transparent to the user who called RETIXU. The only real difference is that RECORD does not really contain the base FID of the group, but instead has the FID of a frame of overflow space. When GUNLOCK is called using the same FID that was returned in RECORD from the RETIXU call, that frame is taken out of the group lock table and remote group FID that was stored in the table entry is sent back to the remote computer with a GUNLOCK command, causing that group to be unlocked on the remote computer. Again this mechanism is transparent to the user who calls GUNLOCK or UPDITM (which also unlocks the group).

A new type of RETIX, RETIXI, has been added which only locks the requested item and not the whole group that contains the item. It is intended that the retrieved item be copied to overflow space and not be left in the file space when being inspected since other processes can update items in the same group, invalidating any pointers to the retrieved and locked item. UPDITM will release the item lock for the item being updated. If the user wishes to unlock an item that he is not going to update, he uses IUNLOCK. The input to IUNLOCK is the value of RECORD that was returned from RETIXI and the item-ID in the BMS work area in the same format as when calling RETIX. The RETIXI subroutine is intended to be used primarily for the implementation of the

new function of the READU statement in BASIC.

There is a "virtual terminal" capability which allows a user whose terminal is physically connected to a local computer to form a logical connection to a remote computer, logon to the remote computer, and then use the remote computer as though his terminal were physically connected to the remote computer.

The format for establishing the virtual terminal connection is:

```
>REMOTE-LOGON REMOTE-COMPUTER-NAME ACCOUNT-NAME, PASSWORD
```

Execution of this command establishes a logical connection to REMOTE-COMPUTER-NAME, spawns a process on the remote computer and logs the user onto the desired account. All output from the terminal is now automatically directed to the process on the remote computer, and all terminal output from the remote computer is directed back to the user's terminal on the local computer.

The user can now use any verbs, commands, and files on the remote computer as though he were directly attached to it. Recall commands can now be executed much more efficiently than in the previous examples where the remote files that were being acted on had to be retrieved from the remote computer to the local computer. Using the virtual terminal facility, the files are manipulated on the remote computer where they reside, and only the output of the command is transmitted back to the local computer.

An example of Recall using the virtual terminal facility is

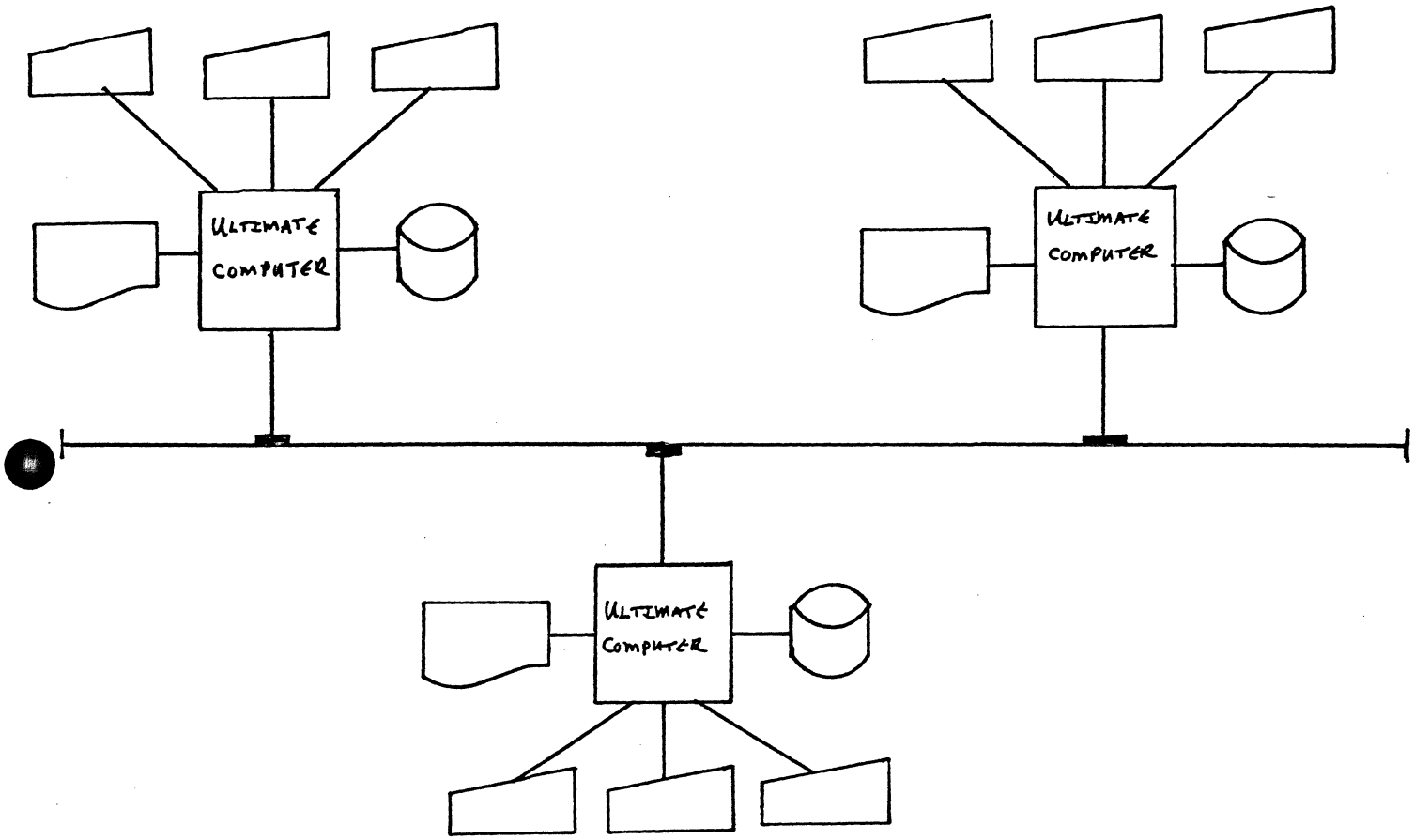
```
>SSELECT REMOTE-FILE-NAME
1487 ITEMS SELECTED
>LIST REMOTE-FILE-NAME
```

Using the virtual terminal facility, the SSELECT command and file name are sent to the remote computer where the sort and selection are performed on the file. The "ITEMS SELECTED" message is returned to the terminal on the local computer while the select-list remains on the remote computer. The LIST command and file name are then sent to the remote computer where the LIST function is performed using the previously formed select-list. The output of the LIST command is then sent to the local computer and is displayed on the user's terminal.

To break the virtual terminal connection, the user enters OFF to the TCL prompt. This ends the session on the remote computer, returns the remote process to the pool of available processes, and breaks the logical connection to the remote computer. The user is then given another TCL prompt which comes from his own local process; he is now once again logically and physically connected to his local computer.

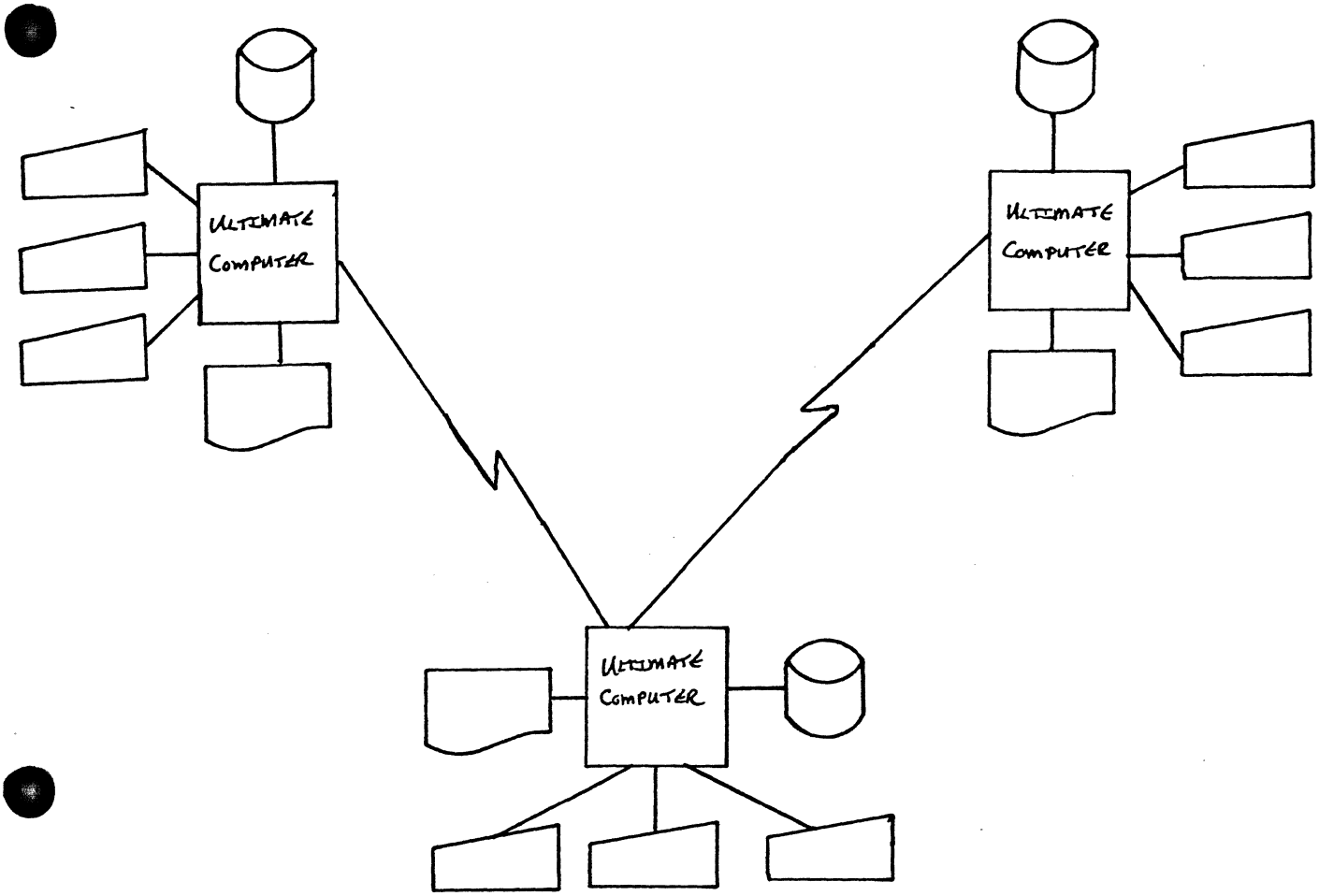
Several network topologies are supported on the network: local area networks, private line networks, and value added carrier networks.

The configuration of the local area network is



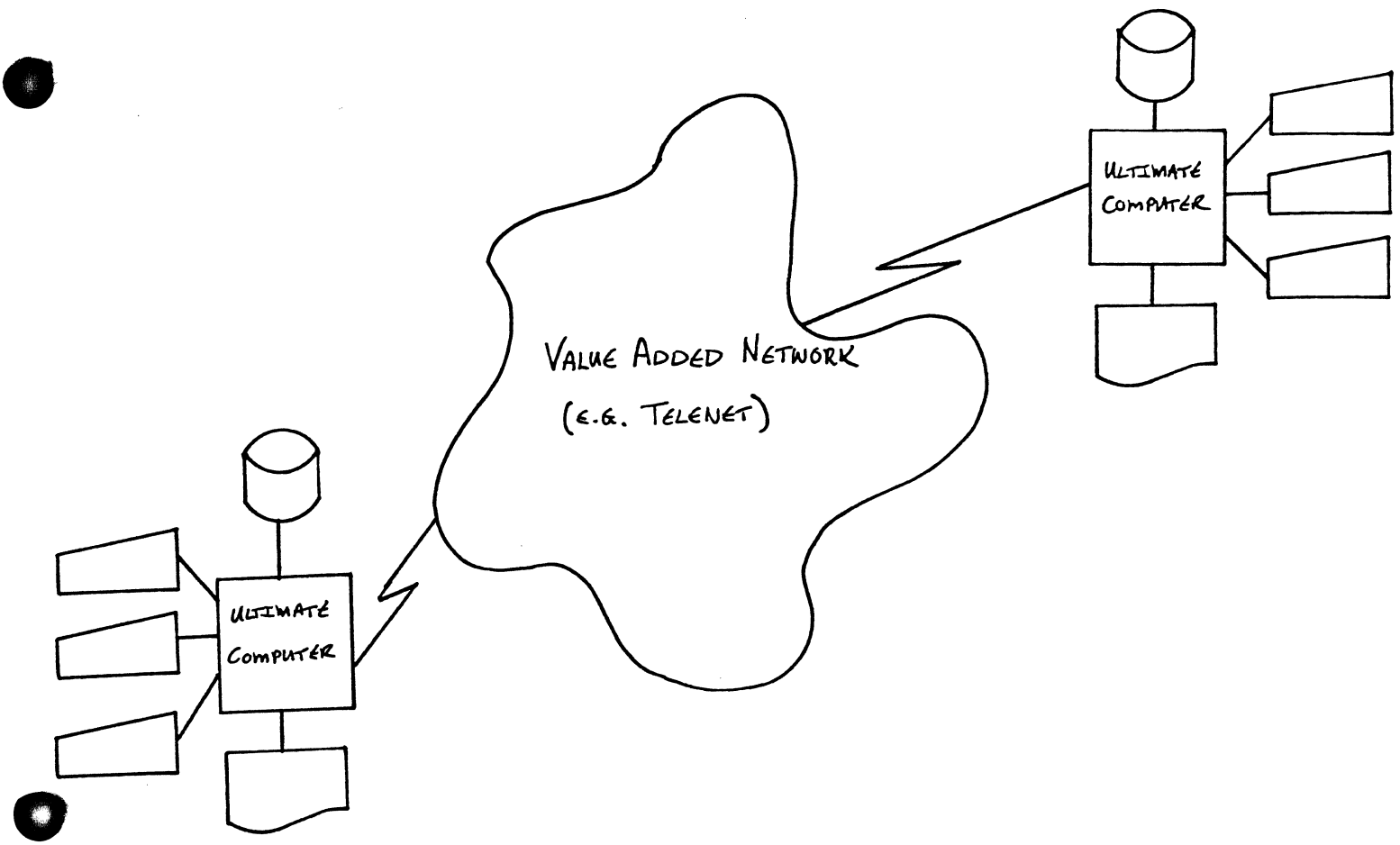
Ultimate computers are connected to a coaxial cable which runs to the necessary locations in the facilities. Access from on Ultimate computer to another is done over the high speed coaxial cable.

The configuration of the private line network could be



The connections between the Ultimate computers are made using leased telephone lines or using private telephone lines.

The attachment to a value added network could be



The connection into the value added network is made using the X.25 interface provided by the carrier.

Farokh Deboo

PRELIMINARY

INTERNAL SPECIFICATION

FOR

REMOTE FILE ACCESS

[Revised October 26, 1983]

TABLE OF CONTENTS

1.0 INTRODUCTION

File access on the system is currently done via several different processes : TCL-II verbs, Recall verbs, SELECT verbs and BASIC. All these methods use the same system subroutines: OPEN, RETIX, GETITM, UPDITM, GUNLOCK and variations of the same.

A 'Remote File Definition Item' for files located on a remote machine is defined and the above subroutines are extended to perform differently when such a remote file is being accessed

When a remote file is encountered by one of these subroutines, it passes a command to the remote machine referenced by the file definition item. The remote machine in turn returns a response message back to the requesting machine with the appropriate status information.

This document covers the basic Application Layer Protocol for the Ultimate network. It provides the format for a remote file definition item, defines commands and responses for remote file access and talks about some of the tables and processes that need to be set up to perfom these accesses.

The protocol and format for the individual commands and responses corresponding to the above-mentioned subroutines (OPEN, RETIX, etc.) are defined in separate documents.

1.1 RELATED DOCUMENTATION

1. Architectural Definition, ULTINET.
2. Ultimate Corporation ULTINET Implementation Plan.
3. User Level Documentation for the Ultimate Network.
4. Specifications on the Open-File Table.
5. Design Specification on Base, Modulo & Separation and Related Tables for Remote File Access.

6. Network Node File User Specification.

7. Interprocess Communication Queue Programmer Specification.

8. Design Specification for Making and Breaking Connections with a Remote Node.

9. Specifications on the Remote Open/Close Command.

10. Design Specification for Retrieving Items from a Remote File.

11. Internal Specifications on "UPDITM" System Routine.

12. Specifications for Group/Item Lock Tables.

1.2 TERMINOLOGY

In the rest of this document, each computer in the Ultimate network is referred to as a node. Each node is assigned a user-visible node name (eg. BLUE) and each node name has a corresponding node number for use at the system-level. The Network Node File contains the necessary information for translating the node name to a node number and vice versa.

2.0 FILE DEFINITION ITEMS FOR FILES AT A REMOTE NODE

Given below is the format of a remote file definition item. It is identical to the current file synonym definition item except for the node name in attribute 4.

0	FILE NAME
1	Q
2	Account name on the Remote Node
3	File name on the Remote Node
4	Node name of the Remote Node
5	<Reserved>
6	<Reserved>
7	<Reserved>
8	<Reserved>
9	Justification on Type Code
10	Maximum Field Length
11	<Reserved>
12	<Reserved>
13	<Reserved>

Unlike the local Q-definition item, on a remote Q-definition item, the Account Name parameter cannot be a null field. The file name in attribute 3, as in the current file synonym definition item, may be in one of the following forms:

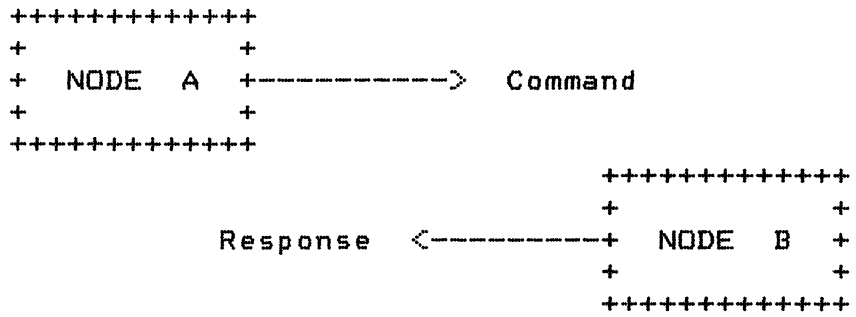
* "FILENAME".

* "FILENAME1,FILENAME2", which implies that the remote access is to be performed on the data file "FILENAME2" associated with the dictionary file "FILENAME1".

* Null, indicating the Master Dictionary of the referenced account on the remote node.

Attributes marked as reserved are currently null attributes.

3.0 COMMANDS & RESPONSES FOR REMOTE FILE ACCESS



The requesting node transmits a command for the required type of file access to the remote node. The remote node in turn processes the command it has received and returns a response. For the first phase of the project, it is assumed that a new command will not be issued until a response is received to the previous command, or unless the previous command was aborted for any reason.

Commands and responses are defined in separate specifications for the following operations

* OPEN, to open a file for remote file access.
* RETIX and variations of RETIX to support group locks, item locks, etc., to retrieve an item from a remote file.
* GETITM, to sequentially retrieve items from a remote file.
* UPDITM, to update or delete an item in a remote file.
* GROUP and ITEM UNLOCK, to unlock a group or item in a remote file.
* CLOSE, to close a remote file.

The system currently does not contain a CLOSE subroutine

call. This subroutine will be called at the termination of the access of a given file. When the file is not remote to the given node, this call will essentially perform a NOP. When closing a remote file, a CLOSE command will be transmitted to the remote node.

In addition to the above commands and responses, a COMMAND REJECT response is used and is described in later sections of this specification.

The following abbreviations will be used in this specification:

C1	:	Primary Command field to identify the type of command being transmitted to a remote node.
C2	:	Secondary Command field to identify the variation of the primary command to be performed.
CAUSE	:	Error Message number in a command reject response.
IND	:	Indicator field sent in a command message.
STAT	:	Status field returned in the response from a remote node.
E#	:	The entry number in the OPEN-FILE Table at a remote node.
,	:	Indicates that the fields on either side of the comma have been concatenated without any delimiter.
[ ]	:	Indicates Optional Parameters within parentheses.

The following fields have a fixed length of 1 byte:

C1 and C2.

The following fields have a fixed length of two bytes:

IND, STAT, E# and CAUSE.

3.1 COMMAND/RESPONSE HEADER

Each command and response message has a header with the following fields:

1. The indicator (IND) field for a command or the status (STAT) field if it is response.

2. The primary command (C1) field.

3. The secondary command (C2) field.

4. The "TO-ADDRESS".

5. The "FROM-ADDRESS".

[6. OPEN-FILE Entry number (E#) corresponding to the file being accessed at the remote node.]

The most significant bit of the first byte is used to distinguish a command from a response message. The C1 and C2 fields returned in a response should be identical to the C1 and C2 fields sent in the corresponding command. The value of zero for the primary command field (C1) is reserved and is currently not used. The TO- and FROM-ADDRESS specify the destination and source of the referenced command/response.

The OPEN-FILE entry number (E#) is included in all commands except for the OPEN command.

3.1.1 THE INDICATOR FIELD

The indicator (IND) field is two bytes long and has the following bit assignments (bit 0 is most significant bit) :

Bit #	Bit Name	Description
0	TYPE	Always set to 0 for commands.
1 - 15	<Reserved>	Currently always 0.

3.1.2 THE STATUS FIELD

The status (STAT) field is two bytes long and has the following bit assignments (bit 0 is most significant bit) :

Bit #	Bit Name	Description
0	TYPE	Always set to 1 for responses.
1	CMNDREJ	Set to 1 if in a command reject response.
2	FAILED	Set to 1 if execution of the command resulted in failure.
3 - 7	<Reserved>	Currently always set to 0.
8 - 15	--	Qualifiers for the different status messages in each of the categories described by the most significant bits.

3.1.3 THE TO-ADDRESS & FROM-ADDRESS FIELDS

The TO- and FROM-ADDRESS specify the destination and source of the referenced command/response at the application level. Each of these fields are 4 bytes in length and have the following two byte sub-fields:

1. Node #

2. Line #

A line number of X'FFFE' is used to identify the File Server Process on each node of the network.

3.2 The COMMAND REJECT Response

If a command received at a remote node could not be executed (eg. receiving a command with an illegal parameter in the command string) it is rejected by the remote node and a Command Reject Response is returned to the requesting node.

This response can be sent to any of the commands listed above, and will have the following format :

STAT , C1 , C2 , TO-ADDR , FROM-ADDR , CAUSE [, Parameter]

where STAT = X'COXX'

An optional parameter may be returned as necessary at the tail of the command reject response.

3.2.1 RECEIVING AN INVALID COMMAND

A command reject response is returned if a remote node received a command with an invalid indicator field or invalid primary or secondary command field, etc. If a command reject is received in response to an invalid command, the initiating process enters an abort condition and breaks the session connection with the remote node related to the given file access.

3.3 RECEIVING AN INVALID RESPONSE

If an illegal response is received by the initiating process, it enters an abort condition and breaks the session connection with the remote node related to the given application-level file access.

3.4 ABORTING A GIVEN REMOTE FILE ACCESS WITH A CLOSE COMMAND

A given file access at a remote node may be aborted before receiving the corresponding response by issuing a CLOSE command for the referenced entry number at that node.

4.0 THE OPEN-FILE TABLE AT THE REMOTE NODE

When receiving an OPEN command the remote node attempts to open the referenced file, and if successful adds it to its list of open files in this table. Details on this table can be found in the OPEN-FILE Table Specification.

5.0 INTERACTION WITH THE VARIOUS LEVELS OF THE NETWORK PROTOCOL

This document covers the basic Application Layer Protocol for the Ultimate Network. The functionality and related services provided by the other layers of the network protocol are discussed in separate documents.

6.0 PROCESSES CREATED AT EACH NODE TO SUPPORT REMOTE FILE ACCESS

The following sub-sections describe some of the processes that are created at network generation time to support remote file access.

6.1 THE COMM SPOOLER

The Comm Spooler is the process associated with the communication interface. It is responsible for maintaining the interface between the virtual processes at each node and the communications onto the network.

6.2 THE FILE SERVER

Each node also has associated with it a File Server Process. When a user sends a command to a remote node to perform a remote file access, the communication interface at this remote node directs this command to the File Server Process on this node.

The File Server carries out the processing dicatated by the received command and returns a response to the initiating

user via the communication interface.

7.0 EXCEPTION CONDITIONS

7.1 ERROR MESSAGES PASSED IN A COMMAND REJECT RESPONSE

The following is a list of the error conditions returned in the CAUSE field of a command reject response. The CAUSE numbers are specified in decimal format.

Remote Access Denied because of :

[2031] Invalid Entry Number.

[2032] Requested Entry Number assigned to another user.

[2019] Received Command with Illegal TO/FROM-ADDRESS at Remote Node.

[2035] Received Illegal Primary Command at Remote Node.

[2036] Received Illegal Secondary Command at Remote Node.

[2038] Received Illegal Indicator Field at Remote Node.

Except for the case of illegal command fields, the command reject response will have as its trailing parameter the value of the illegal parameter field(s) in question.

Error messages will also be generated for some of the conditions listed in the specifications for each of the individual responses.

Farokh Deboo

PRELIMINARY DESIGN SPECIFICATION ON
BASE, MODULO & SEPARATION AND RELATED TABLES
FOR REMOTE FILE ACCESS

[Revised October 24, 1983]

C O N T E N T S

1.0 BASE, MODULO & SEPARATION VALUES FOR REMOTE FILE ACCESS

When a file at a remote node has been opened, the Base, Modulo and Separation fields in the originating user's PCB are updated as follows (bit 0 is the most significant bit) :

* Bit 2 of BASE is set to distinguish it from the BASE field of a local file. Bits 3 to 15 are loaded with the line number of the user and the remaining two lower order bytes of BASE are loaded with the entry number in the BMS table for this particular line.

  In this way the BASE field obtained after an OPEN on any file, local or remote, is unique on any node of the network.

* The MODULO field is don't care.

* The lower byte of the SEPAR field is don't care and the upper byte has the same flag bits as for a local file as follows (bit 0 is the most significant bit):

  Bit 0 = 1 if updates are permitted on this file
  Bit 1 = 1 if referencing a pointer file
  Bits 4- 7 contain the level # of the file

2.0 BMS RELATED TABLES & THE COMMUNICATIONS STORAGE BLOCK (CSB)

At network generation time, a BMS Pointer Table is set up. The starting address of this table is stored in element BMSPTR of the COMM-CB control block. It consists of three linked frames from overflow broken up into ten byte entries, one for each line on the node. Each entry in the BMS Pointer Table has the following fields :

* STATUS [1 Byte]
  Bit 7 = SETUP flag.
  Bits 0 to 6 are reserved and currently set to 0.

* CSB-LOC
  A pointer to the Communication Storage Block for this line if the SETUP bit is set to 1.
  [6 Bytes]

Initially, SETUP = 0. When a call is made to perform a remote OPEN, the user first checks the SETUP bit for its line. If it is zero, it means that the BMS Table has not been created for this line, and results in a block of frames being obtained from overflow.

The first of each of this set of frames is the start of the BMS Table for this line. It is in linked format and can be linked to additional frames as necessary.

The second frame is the Communications Storage Block (CSB) and is used to store communications oriented elements and to save system elements when a remote file is being accessed. One of the entries in the CSB will be BMS-LOC, a pointer to the BMS Table for this line. The CSB is in unlinked format.

The third and last frame in the block is a frame used for scratch purposes by the user.

The CSB-LOC pointer for this line is made to point at the CSB just created, and the SETUP bit is set to 1, to indicate that the BMS Table and CSB can now be accessed via the pointers just set up.

2.1 ENTRIES IN THE BMS TABLE

Each entry of the BMS table is made up of the following fields:

* A status field with the following bits

  > Connection Established
  > OPEN Established
  > This entry currently in use

  The remaining bits of this status field are reserved for internal use.
  [1 Byte]

* ND#, the number of the referenced remote node.
[2 Bytes]

* E#, the entry number of the opened file on the remote node's Open-File Table.
[2 Bytes]

* OVFSTRT. A pointer to the first frame of linked overflow space accumulated, to keep track of the frames being passed on to the process that owns this BMS table for the referenced file that has been OPENED. The frames referenced by OVFSTRT will be released to overflow when the corresponding file is CLOSED.
[4 Bytes]

Each entry of the BMS table will be 25 bytes in length and is referenced by entry numbers starting with the number 1. A 25 byte header at the start of the BMS Table is used to store information common to the line on which this process is operating. The following fields are assigned in this space:

* A status field with the following bits

  > NOTEMPTY, the BMS Table for this line is not null.
  [1 Byte]

* LASTENT, the number corresponding to the last entry in use in the BMS Table for this line.
[2 Bytes]

3.0 RELATED DOCUMENTATION

* Internal Specification for Remote File Access.

Farokh Deboo

PRELIMINARY DESIGN SPECIFICATION
FOR MAKING & BREAKING CONNECTIONS
WITH A REMOTE NODE

[Revised June 13, 1983]

C O N T E N T S

1.0 ESTABLISHING & RELEASING A CONNECTION --- THE CONNECT AND DISCONNECT PRIMITIVES

Before any data can be exchanged across the communication interface, a logical connection needs to be set up. A connection is considered to have been established if there exists an entry in the BMS table with the CONNECT bit set in its status field and with the ND# equal to the number of the referenced node.

For the first phase of the project, the following two primitives are being used for creating a connection and for breaking a given connection:

* CONNECT
* DISCONNECT

Each of these primitives is discussed in detail in the following sections. These primitives provide the functionality of the 'pseudo-session' services, until the formal session services are implemented in later phases.

The CONNECT and DISCONNECT primitives are broken down into the following events:

CONNECT-REQUEST
CONNECT-INDICATION
CONNECT-RESPONSE
CONNECT-CONFIRM

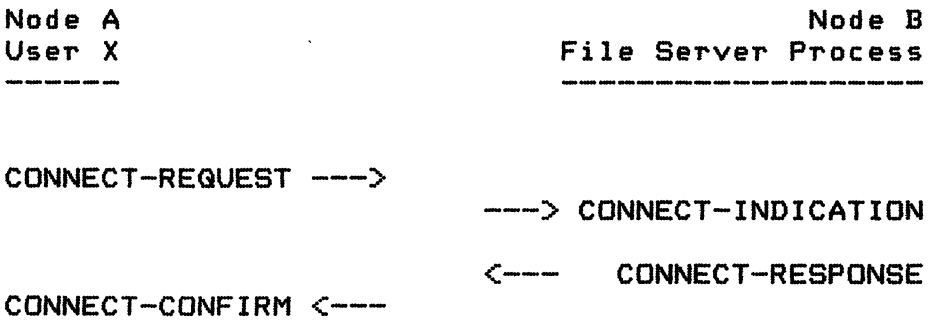
DISCONNECT-REQUEST
DISCONNECT-INDICATION

1.1 SETTING UP A CONNECTION

When a call is made by the user to the OPEN subroutine and it is determined that a remote file is being accessed, a search is made through the BMS Table to find out if a connection exists to the referenced remote node. If it is determined that a connection has already been made, data transfer phase will be entered immediately.

If a connection has not been set up, a CONNECT-REQUEST is transmitted to the remote node. The initiator at this stage can expect one of the following two responses from the remote node:

(a) CONNECT-CONFIRM, indicating that the requested connection was successfully established.



(b) DISCONNECT-INDICATION, indicating that the connection could not be made. The source of the rejection can be either the File Server Process at the remote node or the communication process between the two nodes of the network.

The following examples show some of the sequences possible for such conditions.

```
Node A                                  Node B
User X                          File Server Process
-----                           -------------------


CONNECT-REQUEST          --->

                        (Connection rejected by the
                        Communication Process due to
                        network failure)

DISCONNECT-INDICATION <---
```

```
Node A                                  Node B
User X                          File Server Process
------                          -------------------

CONNECT-REQUEST          --->
                            ---> CONNECT-INDICATION

                        (Connection rejected by the
                        File Server Process since it
                        is out of resources)

                            <--- DISCONNECT-REQUEST
DISCONNECT-INDICATION <---
```

1.1.1 THE CONNECT-REQUEST TIME-OUT

The CONNECT-REQUEST has a time-out associated with it. If after transmitting the CONNECT-REQUEST, a CONNECT-CONFIRM is not received within this time interval, it will be assumed that the connection was not successful. The user that was attempting to initiate the connection will then transmit a DISCONNECT-REQUEST to the remote node.

1.1.2 FILE SERVER RECEIVING ILLEGAL ENTRIES BEFORE A CONNECTION HAS BEEN ESTABLISHED

If, before a connection has been established, the File Server receives an illegal entry from the receive queue, it will simply discard it.

1.2 BREAKING AN EXISTING CONNECTION

Receiving a DISCONNECT-INDICATION at any time results in the connection being broken between the two referenced nodes. Under normal conditions, the user that initiated the OPEN may request a disconnection when it is through with its current remote file access.

Node A User X	Node B File Server Process
DISCONNECT-REQUEST --->	---> DISCONNECT-INDICATION

(Normal termination of pseudo-session)

Under other conditions, receipt of a DISCONNECT-INDICATION is treated as an abort and appropriate error recovery will have to be initiated. The following example shows one possible sequence for a disconnect under such conditions.

Node A User X	Node B File Server Process
DISCONNECT-INDICATION <---	---> DISCONNECT-INDICATION

(Abort due to network failure)

The user process may also request a disconnect when it receives an illegal entry from the receive queue. The File Server generates a DISCONNECT-REQUEST if it receives a duplicate CONNECT-INDICATION or if it receives an illegal entry from the receive queue.

2.0 DATA TRANSFER PHASE

After a connection has been established, the data transfer phase is entered. The user intitiating the OPEN can start sending commands to and receiving responses from the remote node.

2.1 EXITING THE DATA TRANSFER PHASE

The data transfer phase can be exited in two ways:

* The normal case is when the initiating user directly or indirectly makes a call to CLOSE with the disconnect option invoked. This results in an orderly shutdown of the connection. (It may also result in the abortion of the previous command if this command was pending).

* The user requests a disconnect because of a protocol error.

* The user receives a DISCONNECT-INDICATION for one

of several reasons. This can happen, for instance, when a network failure occurs while the two nodes were in data transfer phase or if an illegal entry or a duplicate CONNECT-INDICATION was received by the File Server. Appropriate error recovery will be initiated and error messages will be sent to the initiating user to notify it of the abort.

3.0 THE CAUSE FIELD AS A MEANS OF QUALIFYING THE CAUSE OF A GIVEN EVENT

The receipt of a CONNECT-RESPONSE, CONNECT-CONFIRM, DISCONNECT-REQUEST and a DISCONNECT-INDICATION may be due to several reasons and is indicated by a CAUSE field in the entry dequeued from the receive queue.

3.1 CAUSES FOR A CONNECT

Currently there is only one cause, and the CAUSE field of the control entries related to the CONNECT primitive will not be relevant.

3.2 CAUSES OF A DISCONNECT

A DISCONNECT-REQUEST may be issued for any of the following reasons (values of the CAUSE field are in decimal format) :

* Orderly release of the connection (issued by the initiating user).

  [CAUSE = 0000]

* After a time-out to a CONNECT-REQUEST. The DISCONNECT-REQUEST is issued by the initiating user in this case.

  [CAUSE = 2005]

* On the user receiving an illegal response to a CONNECT-REQUEST.

  [CAUSE = 2003]

* On the user Receiving an illegal response during data transfer phase.
[CAUSE = 2008]

* On the File Server Receiving an illegal response during data transfer phase.
[CAUSE = 2010]

* In response to the File Server having received a CONNECT-INDICATION after a connection has already been successfully established.
[CAUSE = 2014]

* The File Server may also issue a DISCONNECT-REQUEST if it has run out of resources on receiving a new CONNECT-INDICATION and is not in a position to support any more logical connections.
[CAUSE = 2013]

Each one of the above will result in a DISCONNECT-INDICATION being sent across the network with the given CAUSE field.

A DISCONNECT-INDICATION may also be sent by the communication process for one of the following reasons :

* Network Failure.
[CAUSE = 2004]

* Network Access Denied to Requesting User.
[CAUSE = 2039]

3.3 FORMAT OF CONNECT/DISCONNECT CONTROL ENTRIES

The format for each of the control entries to be passed across the network is described in the Interprocess Queue specification. In the case of DISCONNECT type of control entries, the value of the CAUSE field will be determined by which of the reasons listed in the previous section caused the DISCONNECT to occur.

3.3.1 IDENTIFICATION OF THE CONNECTION-RELATED PRIMITIVE EVENTS VIA THE QUALIFIER FIELD

The 16 bit QUALIFIER field in a queue entry will identify the particular event primitive. The following bit assignments will be used (bit 0 is most significant bit):

Bit 1	CONNECT primitive.
Bit 2	DISCONNECT primitive.
Bit 8	REQUEST or INDICATION event.
Bit 9	RESPONSE or CONFIRM event.

For further details on bit assignments in the QUALIFIER field, refer to the Interprocess Communication Queue Specification.

4.0 THE CONNECT TABLE

A Connect Table is maintained at each node of the network. For the initial phases of the project, it only reflects connections made with the file server on each node. The Connect Table is split in two parts and each part consists of a set of linked frames from overflow. The pointers "CONN-TBL1" and "CONN-TBL2" in the COMM-CB control block reference the start of the two parts of this table.

Each entry in the first part of the Connect Table is 4 bytes long and consists of the following fields:

* REM-NODE, the node number of the remote node corresponding to this connection.
  [2 Bytes]

* REM-LINE, the line number on the remote node corresponding to this connection.
  [2 Bytes]

Each entry in the second part of the Connect Table is 100 bytes in length and has the following fields:

* LOC-LINE, the local line number corresponding to this connection. Currently this is always the line number of the file server process. As the Connect Table becomes more global, this field will identify the kind of network application this connection represents.

[2 Bytes]

* DATE, the date on which this connection was established.

[2 Bytes]

* TIME, the time of day when this connection was established.

[4 Bytes]

* CONN-UNITS, the number of charge-units expended by the process on the line defined by LOCLIN to support this connection.

[4 Bytes]

* LAST-C1, the last primary command field received for this connection.

[1 Byte]

* LAST-C2, the last secondary command field received for this connection.

[1 Byte]

* INIT-ACC, the name of the account that initiated this connection. The last character of the account name is followed by an attribute mark.

[51 Bytes]

An entry in the Connect Table is considered as available if it has a value of zero in the corresponding 4 byte field in the first part of the table. Additional fields will be added as necessary to entries in the second part of the table. For instance, it is anticipated that a 1 or 2 byte STATUS field may be required for each connection.

Whereas the earlier sections of this document were related to the "pseudo-session" layer and its associated services, the Connect Table described in this section is related to the application layer of the network protocol.

5.0 RELATED DOCUMENTATION

1. Design Specification on Base, Modulo & Separation and Related Tables for Remote File

Access.

2. Interprocess Communication Queue Programmer Specification.

3. Specification on the Open-File Table.

4. Internal Specification for Remote File Access.

3/28/83

Interprocess Communication Queue

Programmer Specification

The interprocess communication queue is a mechanism whereby a virtual process can pass commands and data to the communications process to transmitted or acted upon and where-by the communications process can pass data and responses to virtual processes.

The queue consists of two linked lists, one for items to be passed to the communications process (the XMT-Q) and one for items to be passed to the virtual processes (the RCV-Q). The entries in the queues are fifty bytes long and are the same format for both the RCV-Q and the XMT-Q. The elements in a queue entry are not all used, depending upon whether the entry is in the RCV-Q or the XMT-Q.

The format of queue entry is:

```
  0   -- QUALIFIER1
  1   -- QUALIFIER2
 2-3  -- TO-NODE
 4-5  -- TO-LINE#
 6-7  -- FROM-NODE
 8-9  -- FROM-LINE#
10-15 -- START-HEADER
16-21 -- START-DATA
22-27 -- END-DATA
28-47 -- available   28-29 CAUSE
48-49 -- Q-LINK
```

The QUALIFIER is two bytes which describe the function of the queue entry. Bits in the QUALIFIER are set to flag the function desired.

	QUALIFIER1	QUALIFIER2
	0 - DATA	0 - REQUEST
	1 - CONNECT	1 - INDICATION
	2 - DISCONNECT	2 - RESPONSE
	3 -	3 - CONFIRMATION
	4 -	4 -
	5 -	5 -
	6 -	6 - REL - HEADER
	7 -	7 - REL - DATA

The TO-NODE and TO-LINE# fields contain the destination address of the request or data unit described by the queue entry. This is a necessary entry for the XMT-Q.

The FROM-NODE and FROM-LINE# fields contain the source address of the indication or data unit described by the queue entry. This is a necessary entry in the RCV-Q, but not needed by the XMT-Q.

The START-HEADER storage register points to the first byte of the header. For a XMT-Q entry, it is assumed that the header is from the position of the storage register to the end of that frame. For a RCV-Q entry, it is assumed that the storage register points to the start of the received data

The START-DATA storage register points to the first byte of the data to be sent in a XMT-Q entry. The header specified by the START-HEADER storage register is concatenated with the data when the data is transmitted. START-DATA is not used in a RCV-Q entry since the communications process does not know where the header ends and the data begins in a received data unit.

The END-DATA storage register points to the last byte of data to be sent or that has been received.

The Q-LINK field is used to keep track of the available entries in the RCV-Q and XMT-Q, to order the entries in the RCV-Q in a first in-first out fashion, and to order the entries in the XMT-Q. The Q-LINK entry is a number between 0 and 499 pointing to the next entry to be processed after the current entry. If no further entries are linked to the current entry, the Q-LINK field has the value x'FFFF'.

Frames 645 and 646 are used for a list of heads and tails for the RCV-Q and XMT-Q respectively. There is one entry in each the RCV-Q and the XMT-Q for each line in the system. For instance, the first entry belongs to line 0 and the tenth entry belongs to line 9. Each entry is four bytes long, two bytes for the head of the list and two bytes for the tail of the list. The value of the head or tail can be between 0 and 499, indicating from the first queue entry to the five hundredth entry in the respective queue. If the line has no entries in its queue, the head and tail will both be x'FFFF'.

There are several subroutines available to manipulate queue entries.

$CONSTRUCT-QS DEFM 0,620

The CONSTRUCT-QS subroutine will initialize all the parameters of the RCV-Q and XMT-Q. This routine should only be used at system start-up and restart. The subroutine can be called using the TCL verb CONSTRUCT-Q.

Input interface: none

Elements used: R13, R14, R15, D0, NNCF

Subroutines used: GETBLK, LINK

Output interface: none

$GET-AVAIL-XMT-ENTRY DEFM 0,621

The GET-AVAIL-XMT-ENTRY subroutine returns to the user an available entry in the XMT-Q.

Input interface: none

Elements used: R13, R14, R15, D0

how about

Output interface: R14 points to the first byte of an available transmit queue entry

$RETURN-AVAIL-ENTRY DEFM 1,621

The RETURN-AVAIL-ENTRY takes a RCV-Q entry and returns it to the pool of available RCV-Q entries.

Input interface: R14 points to the first byte of a RCV-Q entry

Elements used: R13, R14, DO, TEMPTLY

Subroutines used: DECINHIB

Output interface: none

$GET-RCV-Q-ENTRY DEFM 2,621

The GET-RCV-Q-ENTRY subroutine returns an entry from the line's list of received entries in the RCV-Q if there is one available.

Input interface: none

Elements used: R13, R14, R15, RMBIT, DO

Subroutines used: LINESUB, DECINHIB

Output interface: RMBIT = 0 if no entries in this line's RCV-Q
RMBIT = 1 if an entry was found in the RCV-Q
R14 points to the first byte of a RCV-Q entry if one was found

$INSERT-XMT-Q-ENTRY DEFM 3,621

The INSERT-XMT-Q-ENTRY subroutine takes a XMT-Q entry and inserts it in the line's XMT-Q for transmission.

Input interface: R14 points to the first byte of the entry that is to be inserted into the XMT-Q

Elements used: R13, R14, R15, DO, TEMPTLY

Subroutines used: LINESUB, DECINHIB

Output interface: none

THE ULTIMATE CORPORATION
REVISION 1, FOR YOUR COMMENTS
BY C.CARMICHAEL

Farokh

SPECIFICATIONS ON THE OPEN-FILE TABLE

THE OPEN-FILE TABLE IS DESIGNED TO PROVIDE A MEANS FOR THE ULTINET SYSTEM TO USE A COMMON REFERENCE NUMBER WHEN ACCESSING A REMOTE FILE. THIS NUMBER IS REFERRED TO AS THE "E#" THROUGHOUT THE ULTINET SPECS.

THE FILE-SERVER PROCESSOR OF THE REMOTE NODE IS RESPONSIBLE FOR CREATING AND MAINTAINING THE TABLE ENTRIES WHEN A REMOTE OPEN COMMAND IS SUCCESSFULLY COMPLETED. THE E# IS THEN SENT BACK TO THE NODE:LINE# THAT REQUESTED THE FILE BE OPENED. THEREAFTER, ANY ACCESS OF THE FILE WILL BE MADE THROUGH REFERENCING THE E#. IF BOTH THE DICT AND DATA SECTIONS ARE TO BE OPENED THEN TWO E#'S WILL BE SENT AS PROVIDED FOR WITHIN THE OPEN RESPONSE (SEE OPEN/CLOSE SPECS FOR DETAILS).

THE NUMBER OF REMOTELY OPENED FILES WILL BE LIMITED. THIS NUMBER WILL BE DETERMINED AT A LATER DATE BASED ON THE ABILITY OF THE FILE-SERVER PROCESSOR TO HANDLE THE WORK-LOAD.

AT NETWORK GENERATION TIME, LINKED FRAMES WILL BE RESERVED TO ACCOMODATE THE NUMBER OF OPEN-FILE TABLE ENTRIES ALLOWED. A STORAGE REGISTER WILL BE RESERVED IN THE COMM-CB AREA WHICH WILL POINT AT THE BEGINNING OF THE OPEN-FILE TABLE.

THE OPEN-FILE TABLE IS DIVIDED INTO TWO PARTS. THE FIRST PART IS COMPRISED OF AVAILABLE STATUS BYTES, WHICH FLAG AN OPEN ENTRY IN THE TABLE. THE POSITION OF THE STATUS BYTE IS USED AS THE E# AND AS THE OFFSET INTO THE SECOND PART OF THE TABLE. THE SECOND PART OF THE TABLE CONTAINS:

1. FLAGS BYTE.

   BIT 0 = DAF1, FLAGS "UPDATE" OPTION IN EFFECT.
   BIT 1 = DAF7, FLAGS GETITM INITIALIZATION DONE.

2. CONNECT TABLE ENTRY #. (1 BYTES)

3. POINTER INTO CONNECT TABLE ENTRY. (6 BYTES)

4. BASE OF THE FILE. (4 BYTES)

5. MODSEP OF THE FILE. (4 BYTES)

6. DATE OF THE FILE OPEN. (2 BYTES)

7. TIME OF THE FILE OPEN. (4 BYTES)

8. SBASE, USED BY GETITM, LAST GROUP RETRIEVED. (4 BYTES)

9. SMODSEP, USED BY GETITM, NUMBER OF GROUPS LEFT. (4 BYTES)

10. OVRFLCTR, USED BY GETITM, FID OF THE ITEM-ID TABLE. (4 BYTES)

11. NXTITM, USED BY GETITM, POINTER INTO ITEM-ID TABLE. (6 BYTES)

THE CONNECT TABLE SR ADDS THE ABILITY TO CROSS-REFERENCE THAT TABLE'S DATA. THIS ALLOWS DATA SUCH AS THE NODE, LINE, AND REQUESTING ACCOUNT NAME TO BE STORED ONLY ONCE PER CONNECTION.

NETWORK NODE FILE

USER SPECIFICATION

APRIL 6, 1983

THERE IS A FILE, NODES, ON THE NETWORK ACCOUNT WHICH DEFINES THE NODES IN THE NETWORK. THE NODES FILE IS USED TO TRANSLATE THE USER LEVEL NODE NAME TO NETWORK RECOGNIZABLE PARAMETERS.

IN THE NETWORK SYSTEM, THE VARIOUS MEMBER COMPUTERS OF THE NETWORK ARE GIVEN DESCRIPTIVE NAMES SUCH AS CHICAGO OR PAYROLL-SYSTEM. WHEN A DESCRIPTIVE NAME IS USED WHEN REFERENCING A FILE, THE SYSTEM WILL AUTOMATICALLY MAP THE DESCRIPTIVE NAME INTO A NETWORK ADDRESS RECOGNIZABLE BY THE NETWORK.

THERE ARE TWO TYPES OF ITEMS IN THE NODES FILE. THE FIRST TYPE OF ITEM HAS AS ITS ITEM-ID THE DESCRIPTIVE NAME OF THE NODE AND CONTAINS THE NUMBER OF THE NODE ASSIGNED AT NETWORK GENERATION. THE SECOND TYPE OF ITEM HAS AS ITS ITEM-ID THE NODE NUMBER AND CONTAINS RELEVANT DATA ABOUT THE NODE, SUCH AS ITS PUBLIC DATA NETWORK ADDRESS.

WHEN A FILE IS DEFINED AS REMOTE, THE MASTER DICTIONARY DESCRIPTION OF THE FILE IS A "Q" POINTER WITH THE THIRD ATTRIBUTE OF THE "Q" POINTER BEING THE DESCRIPTIVE NAME OF THE NODE, SUCH AS, "CHICAGO". WHEN A REMOTE FILE IS OPENED, THE SYSTEM TAKES THE THIRD ATTRIBUTE OF THE FILE DEFINITION, THE DESCRIPTIVE NODE NAME, AND USES IT TO RETRIEVE AN ITEM WHOSE ITEM-ID IS THE DESCRIPTIVE NODE NAME FROM THE NODES FILE ON THE NETWORK ACCOUNT. THIS ITEM IS THE NODE-NAME-ITEM. THE FIRST AND ONLY ATTRIBUTE OF THE NODE-NAME-ITEM CONTAINS AN ASCII DECIMAL NUMBER THAT HAS BEEN UNIQUELY ASSIGNED TO THE REMOTE NODE AT NETWORK GENERATION TIME. IT IS THE NUMBER WHICH THE REMOTE NODE RECOGNIZES AS ITS NAME. WHEN PASSING DATA AND COMMANDS TO THE COMMUNICATIONS PROCESSOR, THE USER CONVERTS THE NODE NUMBER TO BINARY AND USES IT AS THE DESTINATION NODE IN THE PARAMETERS IT PASSES TO THE COMMUNICATION PROCESSOR. THE GENERAL FORMAT OF THE NODE-NAME-ITEM IS:

DESCRIPTIVE NODE NAME
1. NODE NUMBER

THE SECOND TYPE OF ITEM IN THE NODES FILE IS THE NODE-NUMBER-ITEM. THE NODE-NUMBER-ITEM CONTAINS INFORMATION ON HOW TO REACH THE REMOTE NODE USING THE NETWORK. THE NUMBER BY WHICH THE REMOTE NODE RECOGNIZES ITSELF IS NOT NECESSARILY THE SAME AS THAT WHICH IS USED BY THE NETWORK ITSELF TO ROUTE INFORMATION. THE NODE-NUMBER-ITEM HAS ONE OF SEVERAL ATTRIBUTES WHICH DEFINE BY THEIR POSITION THE TYPE OF NETWORK CONNECTION BEING USED AND THE ADDRESS THAT THE NETWORK ITSELF USES TO ROUTE INFORMATION BETWEEN NODES CONNECTED TO IT. THE FIRST ATTRIBUTE IS USED AS THE NODE ADDRESS IN A LOCAL AREA NETWORK OR A PRIVATE LINE NETWORK, AND IT IS GENERALLY THE SAME AS THE ITEM-ID OF THE NODE-NAME-ITEM. FOR EXAMPLE, IF THE NODES ARE CONNECTED TO A LOCAL AREA NETWORK AND THE NODE NUMBER OF THE REMOTE COMPUTER AS RETRIEVED FROM THE NODE-NAME-ITEM IS 21, THE FIRST ATTRIBUTE OF THE NODE-NUMBER-ITEM IS THE ASCII NUMBER 21. THE SECOND ATTRIBUTE OF THE NODE-NUMBER-ITEM IS USED FOR THE ADDRESS OF A REMOTE NODE IN A PUBLIC DATA NETWORK SUCH AS TELENET. THE REMOTE NODE RECOGNIZES ITSELF BY THE SAME NUMBER AS THE NODE-NUMBER-ITEM-

ATTRIBUTE TO ROUTE THE DATA TO THE REMOTE NODE. THE THIRD ATTRIBUTE OF THE NODE-NUMBER-ITEM IS USED IF THE REMOTE NODE MUST BE REACHED USING A DIAL-UP TELEPHONE LINE FOR A POINT-TO-POINT CONNECTION TO THE REMOTE NODE. THE THIRD ATTRIBUTE CONTAINS THE TELEPHONE NUMBER OF THE REMOTE NODE. AGAIN THE REMOTE NODE RECOGNIZES ITSELF AS THE SAME NUMBER AS THE NODE-NUMBER-ITEM-ID. ONLY ONE ATTRIBUTE IN THE NODE-NUMBER-ITEM MAY HAVE A VALUE. IF MORE THAN ONE ATTRIBUTE IN THE NODE-NUMBER-ITEM IS NON-NULL, THE FIRST ONE ENCOUNTERED DEFINES THE TYPE OF NETWORK CONNECTION BEING USED. THE GENERAL FORMAT OF THE NODE-NUMBER-ITEM IS:

NODE-NUMBER
1. LOCAL AREA OR PRIVATE NETWORK ADDRESS
2. PUBLIC DATA NETWORK ADDRESS
3. TELEPHONE NUMBER OF REMOTE NODE

THERE IS ONE ITEM IN THE NODES FILE WHICH DEFINES THE NAME OF THE LOCAL NODE. THE ITEM-ID OF THIS ITEM IS "I-AM". THE FIRST ATTRIBUTE OF THE I-AM ITEM GIVES THE NETWORK NODE NUMBER ASSIGNED TO THE LOCAL NODE DURING NETWORK GENERATION. THE LOCAL NODE NUMBER IS AN ASCII DECIMAL NUMBER. THE GENERAL FORMAT OF THE I-AM ITEM IS:

I-AM
1. LOCAL NODE NUMBER

Farokh

TABLE OF CONTENTS

THE ULTIMATE CORPORATION
Revision 0, for your comments
by C. Carmichael

INTERNAL SPECIFICATIONS ON "UPDITM" SYSTEM ROUTINE

THE ULTIMATE CORPORATION
Revision 0, for your comments
by C. Carmichael

1 REMOTE "UPDITM" OVERVIEW

All commands and responses used in the ULTINET SYSTEM have a uniform format. For details of this and standard abreviations used throughout these specs refer to the INTERNAL SPECIFICATIONS FOR REMOTE FILE ACCESS document.

Execution of a remote UPDITM command is used to add, change, or delete an item from a file on a remote node. This command must be preceded by an OPEN command which references the file to be updated. If this is not done, or if there is a spurious E# the command will be rejected. For details on OPEN and the E# see SPECIFICATIONS ON THE REMOTE OPEN/CLOSE COMMANDS and SPECIFICATIONS ON THE OPEN-FILE TABLE.

The virtual process on the requesting side of the ULTINET Network is responsible for building the UPDITM command. The command is delivered to the File-Server Processor (called the FSP) on the remote node.

The FSP retrieves the file information from the Open-file table parameters based on the E# that is sent as part of the command. It then proceeds to update the item specified and send a normal response or reports back to the requestor the reason for the failure.

2 UPDITM COMMAND FORMAT

The format for the UPDITM command is:

1. Indicator1 field = x'00' for commands (1 byte).

2. Indicator2 field = x'00' for commands (1 byte).

3. Primary command = x'04' specifying UPDITM command. (1 Byte)

4. Secondary command (1 byte).

   x'00' = Update item flag.
   x'01' = Delete item flag.
   x'02' = Update, leave group locked flag.

5. E#, Open-file table entry number (2 bytes).

6. Item-id (variable length).

7. Item body, for adds and changes (variable length).

The variable length data elements within the command string are separated by attribute marks, x'FE's. The command string itself is terminated by a segment mark, a x'FF'.

THE ULTIMATE CORPORATION
Revision 0, for your comments
by C. Carmichael

2.1 SUCCESSFUL UPDITM-RESPONSE FORMAT

The format for a successful UPDITM response command is:

1. Status1 = x'80', a normal command response (1 byte).
2. Status2, unused (1 byte).
3. Primary Command echoed back to requestor.
4. Secondary Command echoed back to requestor.
5. Group FID, if secondary command is x'02' (4 bytes).

2.2 FAILED UPDITM-RESPONSE FORMAT

The format for an UPDITM-RESPONSE which is unsuccessful is:

1. Status1 = x'A0', a failed command response (1 byte).
2. Status2 is unused (1 byte)
3. Primary Command echoed back to requestor (1 byte).
4. Secondary Command echoed back to requestor (1 byte).
5. Error message # (2 bytes), such as:

   2001 Aborted to debugger at the remote node.
   2009 GROUP FORMAT ERROR ENCOUNTERED ON REMOTE NODE.
   2031 Remote Access Denied -- Invalid Entry #.
   2032 Remote Access Denied -- CI assigned to another user.
   2033 Remote Access Denied -- Invalid Group FID Specified.
   2034 Remote Node out of Disk Space.

6. Parameter list for GFE's, [illegible] debugger responses

THE ULTIMATE CORPORATION
Revision 2, for your comments
by C. Carmichael

SPECIFICATIONS ON THE REMOTE OPEN/CLOSE COMMANDS

3 OPEN/CLOSE COMMANDS OVERVIEW

The OPEN and CLOSE commands are the initiating and terminating commands in the remote file access procedures. The receipt of an OPEN command assumes that a connection between a local and remote node has been established. If either an OPEN or CLOSE command is received without being preceded by a CONNECT command the remote will ignore the command.

Furthermore, if a CLOSE command is received without being preceded by an OPEN command the remote will send an ABORT/DISCONNECT. For further details about CONNECT/DISCONNECT'S see the PRELIMINARY DESIGN FOR MAKING & BREAKING CONNECTIONS WITH A REMOTE NODE.

The Virtual Process on the requesting side of the Ultinet Network is responsible for building the OPEN/CLOSE commands. A special remote-open routine is entered when, during the processing of an OPEN command, a Remote-file-definition-item (called Remote-Q) is encountered (for details of the the Remote-Q refer to the INTERNAL SPECIFICATIONS FOR REMOTE FILE ACCESS document). This routine builds the OPEN command string which is described in the next section. The command is delivered to the File-Server Processor (called the FSP) on the remote node.

The FSP is responsible for opening the file, adding the information about the file to its Open-File table (see SPECIFICATIONS ON THE OPEN-FILE TABLE for details), and sending back to the requesting node a successful command response which includes an Open-file table entry number (called E#).

The Virtual process at the requesting node then creates an entry in the users BMS table which is used to keep track of the E# and overflow space associated with that file.

The CLOSE command can be explicitly requested by the virtual user or it can be part of the wrapup procedure that terminates a specific job. Besides creating the CLOSE command string the Virtual process will reinitialize the BMS table entry and release the overflow used for that file. Upon receipt of a CLOSE command the FSP will delete the entry in the Open-file table, unlock any items or groups associated with the file, and send a command response indicating the CLOSE was successful.

All commands and responses used in the ULTINET SYSTEM have a uniform format. For details of this and standard abreviations used throughout these specs refer to the INTERNAL SPECIFICATIONS FOR REMOTE FILE ACCESS document. All commands and responses associated with the OPEN file routine are terminated with a segment mark.

THE ULTIMATE CORPORATION
Revision 2, for your comments
by C. Carmichael

4 OPEN COMMAND

The OPEN command is issued by a user wishing to access a file on a remote machine. The successful execution of this command causes the retrieval of the B/M/S of a file and the creation of an entry in the remote machine's Open-file table. The position of the entry within this table is known as the "E#" and is used to reference the file.

4.1 OPEN COMMAND FORMAT

The format for the OPEN command is:

1. Indicator1 field = x'00' for commands (1 byte).
2. Indicator2 field = x'00' for commands (1 byte).
3. Primary command = x'01' specifying OPEN command (1 byte).
4. Secondary command (1 byte).

   Bits 0-4 unused.
   Bit 5 open dict and data sections, SAVE bit.
   Bit 6 open dict section, DAF8 bit.
   Bit 7 update priviledges requested, DAF1 bit.

5. Requesting Node# (2 bytes).
6. Requesting Line# (2 bytes).
7. Requesting Account name (variable length).
8. Remote Account name (variable length).
9. File name as in attr.3 of Remote-Q (variable length).
10. Data name if DICT,DATA format is used at TCL (variable length or null).

The variable length data elements within the command string are separated by attribute marks, x'FE's. The command string itself is terminated by a segment mark, a x'FF'.

THE ULTIMATE CORPORATION
Revision 2, for your comments
by C. Carmichael

4.2 SUCCESSFUL OPEN-RESPONSE FORMAT

The format for a successful OPEN-RESPONSE command is:

1. Status1 = x'80', a normal command response (1 byte).

2. Status2 (1 byte).

Bits	0-3	unused
bit	3	represents DAF8 bit, a single ~~level~~ file.
Bit	5	represents a pointer-file. ~~(DICT is of TYPE 1)~~
Bit	6	represents updating OK.
Bit	7	represents updating the dictionary is OK where applicable.

4 = Pointer File

3. Primary Command echoed back to requestor.

4. Secondary Command echoed back to requestor. 4

5. Open-File table "E#" for data section (2 bytes). 6

6. Open-file table "E#" for dict section where applicable. 8
   Note: If both the DICT and DATA section are requested and the file is DICT only the E# is repeated. (2 Bytes).

4.3 FAILED OPEN-RESPONSE FORMAT

The format for an OPEN-RESPONSE which is unsuccessful is:

1. Status1 = x'A0', a failed command response (1 byte).

2. Status2 is unused (1 byte).

3. Primary Command echoed back to requestor (1 byte).

4. Secondary Command echoed back to requestor (1 byte).

5. Error message # (2 bytes), such as:

2200 ~~REMOTE ACCOUNT NAME?~~ IS NOT AN ACCOUNT NAME ON REMOTE NODE
2201 IS NOT A FILE NAME ON REMOTE ~~MACHINE~~ NODE
2202 ILLEGAL SECOND LEVEL OF REMOTE-Q FOUND.
2210 IS ACCESS PROTECTED ON REMOTE MACHINE. UPD (2211) RETR (2210)
2213 REMOTE FILE DATA LEVEL DESCRIPTOR IS MISSING.
2033 ~~OFF~~
2034 Out of disk space at Remote Node
2001 Buffered debugger

5 CLOSE COMMAND

The CLOSE command is the complement to the OPEN command. Just as OPEN causes the retrieval of the B/M/S of a file and creates an Open-file table entry, the CLOSE causes the item in the Open-file table to be deleted. The closing of file(s) can be initiated by the user or can be the natural results of WRAPUP. Furthermore, the user can optionally request the closing of an individual file or the closing of all files that were opened by the user. If all files are closed then the CLOSE routine additionally searches the Group-lock table and unlocks any group/item(s) associated with the user.

5.1 CLOSE COMMAND FORMAT

The format for the CLOSE command is:

1. Indicator1 field = x'00' for commands (1 byte).
2. Indicator2 field = x'00' for commands (1 byte).
3. Primary command = x'06' specifying CLOSE command (1 byte)
4. Secondary command (1 byte).

   X'00' = a specific file is to be closed.
   ~~X'01' = all files opened by user are to be closed~~
5. Open-file table E# (2 bytes).
   (Used only if the secondary command is x'00')

5.2 SUCCESSFUL CLOSE-RESPONSE FORMAT

1. Status1 = x'80', a normal command response (1 byte).
2. Status2 = x'00', not used (1 byte).
3. Primary Command echoed back to requestor.
4. Secondary Command echoed back to requestor.
5. Error message # (2 bytes)

   2015 REMOTE FILE xxx CLOSED.
   2016 ALL YOU REMOTE FILES CLOSED ON NODE xxxx.

5 Parameter list of E#'s closed

5.3 FAILED CLOSE-RESPONSE FORMAT

The only circumstance that would prevent a successful file closing is assumed to be an error in the command string. Therefore a command-reject response is returned to the user. The format for a command-reject is covered in the INTERNAL SPECIFICATION FOR REMOTE FILE ACCESS.

Also abort x'A0'00, 2001, ---
Out of Disk space x'A0'00, 2034

Farokh Deboo

PRELIMINARY DESIGN SPECIFICATION FOR RETRIEVING ITEMS FROM A REMOTE FILE

[Revised September 25, 1983]

C O N T E N T S

1.0 OVERVIEW

After having opened a file via the OPEN subroutine, a user can retrieve items from this file by making a call to either the RETIX (or one of its variants) or the GETITM subroutine.

This document describes the protocol to be observed at the application level and some of the changes that need to be made on the operating system to support the retrieval of items from a remote file.

1.1 RELATED DOCUMENTATION

1. User Level Documentation for the Ultimate Network.
2. Internal Specification for Remote File Access.
3. Specifications on the Open-File Table.
4. Design Specification on Base, Modulo & Separation and Related Tables for Remote File Access.
5. Specifications on the Remote Open/Close Commands.
6. Design Specification for Making and Breaking Connections with a Remote Node.
7. Specifications for Group/Item Lock Tables.

1.2 TERMINOLOGY

Commas are used to indicate concatenated fields in the command and response formats. When referencing bits in a field, bit 0 will indicate the most significant bit.

2.0 RETRIEVING ITEMS WITH RETIX

The following variants of RETIX are supported for performing remote file accesses:

* RETIX
* RETI
* RETIXU : RETIX with a group lock.
* RETIXUX : RETIXU with the option to quit if the group was found locked.
* RETIXI : RETIX with an item lock.
* RETIXIX : REITIXI with the option to quit if the item/group was found locked.

The current RETIXX entry point will not perform remote file accesses.

2.1 THE RETIX COMMAND

When a call is made to one of the above RETIX subroutines, the BASE field is checked to ascertain whether or not a remote file is being accessed. If the file is local, normal processing continues.

If it is found to be a remote file, the user process prepares to transmit a RETIX command to the referenced remote node. This command has the following format :

IND , C1 , C2 , TO-ADDR , FROM-ADDR , E# , ITEM-ID ^

where,

* IND is the indicator field. Always set to zero. [2 Bytes]
* C1 the primary command field. C1 is set equal to X'02' for RETIX type commands. [1 Byte]

* C2 the secondary command field qualifies the variation of RETIX that is to be executed and has the following bit assignment (bit 0 is the most significant bit):

  Bits 0 - 4 Reserved and currently set to 0.

  Bit 5 Set if the X option is in effect (ie. RETIXUX, RETIXIX). Valid only if bit 7 = 1.

  Bit 6 0 for Group Lock, 1 for Item Lock. Valid only if bit 7 = 1.

  Bit 7 0 if no locks requested, 1 if locks are requested.

  [1 Byte]

* TO-ADDR Is the application-level destination address (node #, line #) of the File Server at the Remote Node.

  [4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the user requesting the RETIX.

  [4 Bytes]

* E# the entry number in the remote node's Open-File Table used to reference the given remote file.

  [2 Bytes]

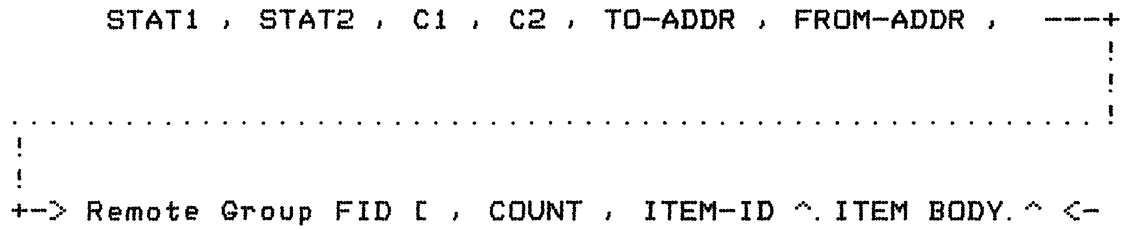
* ITEM-ID the variable length Item-Id of the item to be retrieved. The Item-Id is terminated by an attribute mark.

2.2 THE RETIX RESPONSE

After having processed the received RETIX command, the remote node returns a RETIX response. The following subsections describe the types of responses that can be expected.

2.2.1 NORMAL RETIX RESPONSE

The following is the format of the response string returned by the File Server at the remote node when RETIX successfully retrieves an item, or if it was unable to retrieve the referenced item under normal conditions:



where,

* STAT1 is the high order byte of the status field and is equal to X'80'.

  [1 Byte]

* STAT2 is the low order byte of the status field with the following bit assignments

  Bits 0 - 5 Reserved and currently set to 0.

  Bit 6 Analogous to the RTNFLG bit in GLOCK/GLOCK. (RTNFLG will be replaced by SB61 for new GLOCK routine).

  Bit 7 Analogous to RMBIT for RETIX. Set if an item was found, zero otherwise.

  [1 Byte]

* C1, C2 the echoed primary and secondary command fields from the RETIX command string.

  [1 Byte each]

* TO-ADDR Is the application-level address (node #, line #) of the user requesting the RETIX.

  [4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the responding File Server at the Remote Node.
[4 Bytes]

* Remote Group FID the FID of the group to which this item belongs at the remote node.
[4 Bytes]

* COUNT the count field for this item.
[4 Bytes]

The ITEM-ID and the ITEM-BODY are variable length strings each terminated by an attribute mark just as they would be in the normal file structure. If an item was retrieved, the entire response string will be terminated by a segment mark.

2.2.2 FAILED RETIX RESPONSES

The following is the format of some of the other RETIX responses possible :

STAT , C1 , C2 , TO-ADDR , FROM-ADDR , CAUSE [Parameters]

where,

* STAT is the status field. For the failed condition, STAT is set equal to X'A000'.
[2 Bytes]

* C1, C2 the echoed primary and secondary command fields from the RETIX command string.
[1 Byte each]

* TO-ADDR Is the application-level address (node #, line #) of the user requesting the RETIX.
[4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the responding File Server at the Remote Node.

[4 Bytes]

* CAUSE this field qualifies the reason why an item was not retrieved.

[2 Bytes]

The following numbers are used for the CAUSE field (values are in decimal format) :

CAUSE = 2001 aborted to the debugger at the remote node.

= 2009 a group format error was encountered at the remote node.

See the section on handling group format errors for further details.

A command reject response is returned if the RETIX could not be performed for any other reason.

2.3 ITEM/GROUP LOCKS WITH RETIX

The protocol for interfacing with the group and item lock handler needs to be further defined and will be included in a later revision of this document.

3.0 SEQUENTIALLY RETRIEVING ITEMS WITH GETITM

When a call is made to the GETITM subroutine, a check is made to see if the referenced file is at a remote node. If it is a local file, normal processing continues. If remote, a GETITM command is transmitted to the remote node on which this file resides.

3.1 THE GETITM COMMAND

The format of the GETITM command is as follows :

IND , C1 , C2 , TO-ADDR , FROM-ADDR , E#

where,

* IND is the indicator field. Always set to zero.
[2 Bytes]

* C1 the primary command field. C1 is set equal to X'03' for GETITM type commands.
[1 Byte]

* C2 the secondary command field qualifies the variation of GETITM that is to be executed. The following is the bit assignment in the C2 field :

  Bits 0 - 6 Reserved. Currently always zero.

  Bit 7 Analogous to the DAF7 bit. Zero in this location indicates that this is the first call to GETITM.

[1 Byte]

* TO-ADDR Is the application-level destination address (node #, line #) of the File Server at the Remote Node.
[4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the user making the call to GETITM.
[4 Bytes]

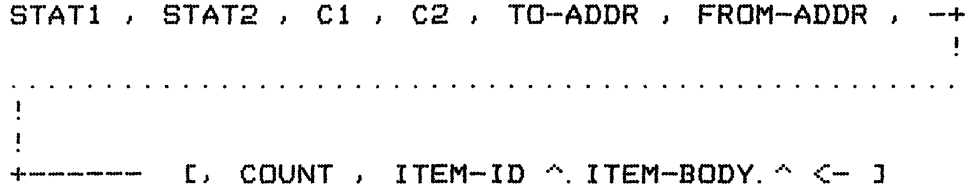
* E# the entry number in the remote node's Open-File Table used to reference the given remote file.
[2 Bytes]

3.2 THE GETITM RESPONSE

After having processed the received GETITM command, the remote node returns a GETITM response. The following subsections describe the types of responses that can be expected.

3.2.1 NORMAL GETITM RESPONSE

If an item was retrieved or if all the items in the referenced file have been sequentially retrieved, the GETITM response has the following format :



where,

* STAT1 is the high order byte of the status field and is equal to X'80'.
[1 Byte]

* STAT2 is the low order byte of the status field with the following bit assignments

Bits 0 - 6 Reserved and currently set to 0.

Bit 7 Analogous to RMBIT for GETITM. Set if an item was found, zero if there are no more items.
[1 Byte]

* C1 , C2 the echoed primary and secondary fields from the GETITM command.
[1 Byte each]

* TO-ADDR Is the application-level address (node #, line #) of the user making the call to GETITM.
[4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the responding File Server at the Remote Node.
[4 Bytes]

The COUNT, ITEM-ID and ITEM-BODY have the same format as in a RETIX response.

3.2.2 FAILED GETITM RESPONSES

The following is the format of some of the other GETITM responses :

STAT , C1 , C2 , TO-ADDR , FROM-ADDR , CAUSE [, Parameters]

where,

* STAT is the returned status. For a failed response, it is set equal to X'A000'.
[2 Bytes]

* C1 , C2 the echoed primary and secondary command fields from the GETITM command.
[2 Bytes]

* TO-ADDR Is the application-level address (node #, line #) of the user making the call to GETITM.
[4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the responding File Server at the Remote Node.
[4 Bytes]

* CAUSE this field qualifies the reason why an item was not retrieved. The following values are used :
[2 Bytes]

CAUSE = 2001 Aborted to debugger at the remote node.

= 2009 A group format error was encountered at the remote node and a GFE parameter list is passed on via this response.

See the section on handling of GFE's for further details.

= 2034 Remote node out of disk space.

A command reject response is returned if the RETIX could not be performed for any other reason.

4.0 HANDLING GFE'S DURING RETIX & GETITM

The sections of the system that take care of this GFE error handling (eg. RETIX-BSL, RETIX-XMODE, GNSEQI-BSL, etc.) need to be changed so that appropriate parameters are returned by the File Server Process to identify the location of the group format error via a failed RETIX or GETITM response.

5.0 OVERFLOW HANDLING DURING RETIX & GETITM

The sections of the system that take care of overflow handling (eg. BMSOVF via XMODE) need to be modified so that if there is not enough disk space on the remote node, a failed RETIX or GETITM response can be returned by the File Server Process with the appropriate CAUSE field.

Farokh Deboo

PRELIMINARY DESIGN SPECIFICATION
FOR UNLOCKING GROUPS & ITEMS
IN A REMOTE FILE

[Revised January 27, 1984]

C O N T E N T S

1.0 OVERVIEW

After having locked a group or an item in a remote file using variations of the RETIX or UPDITM subroutines, a user can unlock the group or item by making a call to the GUNLOCK or the IUNLOCK subroutine respectively.

1.1 RELATED DOCUMENTATION

1. User Level Documentation for the Ultimate Network.
2. Internal Specification for Remote File Access.
3. Specifications on the Open-File Table.
4. Design Specification on Base, Modulo & Separation and Related Tables for Remote File Access.
5. Specifications on the Remote Open/Close Commands.
6. Design Specification for Making and Breaking Connections with a Remote Node.
7. Design Specification for Retrieving Items from a Remote File.
8. Internal Specifications on "UPDITM" System Routines.
9. Specifications for Group/Item Lock Tables.
10. Design Specification on the Management of Overflow Space for Remote File Access by the User Process.

1.2 TERMINOLOGY

Commas are used to indicate concatenated fields in the command and response formats. When referencing bits in a

field, bit 0 will indicate the most significant bit.

2.0 THE UNLOCK COMMAND

When a call is made to either the GUNLOCK or the IUNLOCK subroutine, the RECORD field is checked to ascertain whether or not a remote file is being accessed. If the file is local, normal processing continues.

If it is found to be a remote file, the user process prepares to transmit an UNLOCK command to the referenced remote node (see also section 2.1). This command has the following format :

IND , C1 , C2 , TO-ADDR , FROM-ADDR , E# , RGFID , ITEM-ID ^

where,

* IND is the indicator field. Always set to zero.
[2 Bytes]

* C1 the primary command field. C1 is set equal to X'05' for UNLOCK type commands.
[1 Byte]

* C2 the secondary command field qualifies whether a group or item unlock is to be carried out. It has the following bit assignment :
[1 Byte]

Bit 7 0 for Group Unlock
1 for Item Unlock.

Bits 0 - 6 reserved and currently equal zero.

* TO-ADDR Is the application-level destination address (node #, line #) of the File Server at the Remote Node.
[4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the user requesting the UNLOCK.
[4 Bytes]

* E# the entry number in the remote node's Open-File Table used to reference the given remote file. [2 Bytes]

* RGFID The starting FID of the referenced group at the remote node. [4 Bytes]

* ITEM-ID the variable length Item-Id of the item to be unlocked . The Item-Id is terminated by an attribute mark.

2.1 GUNLOCK IMMEDIATELY AFTER A CALL TO RETIXI

In general, after making a subroutine call to RETIXI, on a successful return, the user process gets pointers into the retrieved item and both this item and its corresponding group are locked. The user then is responsible for copying out this item and unlocking the group.

When a RETIXI is performed on an item in a remote file (see Design Specification on Management of Overflow Space for Remote File Access), the File Server at the Remote Node has already copied out the item into some work-space and has then unlocked the corresponding group on its side.

So when the requesting user issues the call to GUNLOCK, some internal processing is performed via the LFID Table at the local node, however an explicit GUNLOCK command does not need to be transmitted since the group has already been unlocked at the remote node.

3.0 THE UNLOCK RESPONSE

After having processed the received UNLOCK command, the remote node returns a UNLOCK response. The following sections describe the types of responses that can be expected.

3.1 NORMAL UNLOCK RESPONSE

The following is the format of the response string returned by the File Server at the remote node when UNLOCK successfully unlocks a group or item.

STAT , C1 , C2 , TO-ADDR , FROM-ADDR

where,

* STAT is the high order byte of the status field and is equal to X'8000'.
[2 Bytes]

* C1, C2 the echoed primary and secondary command fields from the UNLOCK command string.
[1 Byte each]

* TO-ADDR Is the application-level address (node #, line #) of the user requesting the UNLOCK.
[4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the responding File Server at the Remote Node.
[4 Bytes]

3.2 FAILED UNLOCK RESPONSES

The following is the format of some of the other UNLOCK responses possible :

STAT , C1 , C2 , TO-ADDR , FROM-ADDR , CAUSE [Parameters]

where,

* STAT is the status field. For the failed condition, STAT is set equal to X'A000'.
[2 Bytes]

* C1, C2 the echoed primary and secondary command fields

from the UNLOCK command string.
[1 Byte each]

* TO-ADDR Is the application-level address (node #, line #) of the user requesting the UNLOCK.
[4 Bytes]

* FROM-ADDR Is the application-level address (node #, line #) of the responding File Server at the Remote Node.
[4 Bytes]

* CAUSE this field qualifies the reason why an item was not retrieved.
[2 Bytes]

The following numbers are used for the CAUSE field (values are in decimal format) :

CAUSE = 2001 aborted to the debugger at the remote node.

CAUSE = 2034 out of disk space at remote node.

A command reject response is returned if the UNLOCK could not be performed for any other reason.

Farokh Deboo

PRELIMINARY DESIGN SPECIFICATION ON

THE MANAGEMENT OF OVERFLOW SPACE FOR REMOTE FILE ACCESS

BY THE USER PROCESS

[Revised October 25, 1983]

C O N T E N T S

1.0 MANAGEMENT OF OVERFLOW SPACE

When creating commands, the user process obtains overflow frames as necessary, formats these commands and enqueues them on the transmit queue. Flags in this queue entry's qualifier field indicate whether or not overflow frames have been used for this entry, and the comm-spooler process is responsible for releasing these frames back to overflow.

When receiving responses via the receive queue, it is currently assumed that all responses are returned to the user via overflow frames. These frames are broken down into three categories as follows :

1. Frames containing data in response to a command not requiring a group or item lock (eg. RETIX, GETITM).

2. Frames containing data in response to a command requesting a group or item lock (eg. RETIXU, RETIXI).

3. All other cases.

Frames included in category 3 are discarded after their contents have been examined. The following sections describe the other two categories.

2.0 FRAMES REFERENCED BY THE OVFSTRT POINTER

In the case of data received in response to a command not requiring group or item locks (items retrieved with a call to RETIX OR GETITM), only the image of the last item retrieved is saved. The previous chain of overflow space is released when a new chain is received.

The starting FID of the last chain received is saved in the OVFSTRT field of the BMS Table entry corresponding to the remote file being accessed. The chain referenced by OVFSTRT is released to overflow when the corresponding remote file is closed.

3.0 FRAMES REFERENCED BY THE LFID TABLE

Frames containing data received in response to a command requesting group or item locks are all saved.

These frames are released under the following conditions :

(i) If the LFID Table already points to an older image of the same item in the same remote file, the older image of that item is discarded.

(ii) After successfully executing a remote UPDITM with the U or D option, the chain of frames (if any) corresponding to the updated/deleted item is released.

(iii) When an explicit UNLOCK command is issued all the chains corresponding to the referenced remote group FID are released.

(iv) When a remote file is closed all chains corresponding to the referenced BMS Table entry are released.

The Local FID (LFID) Table described in the following section is used to keep track of these frames.

3.1 THE LFID TABLE

Each user can have one LFID Table. The LFID Table is created the first time a user retrieves a remote item with a group or item lock and is released to overflow when the user issues a request to close all remote files (eg. during wrap-up).

Each entry of the LFID Table is 10 bytes in length and has the following fields :

* The starting FID of the chain corresponding to this entry

  [4 Bytes]

* The corresponding BMS Table entry number

  [2 Bytes]

Each entry in the LFID Table will be assigned an index number starting with the number 1. The LFID Table will have a 10 byte header consisting of the following :

* The last entry in use in the LFID Table
[2 Bytes]

The remaining space in the header is reserved for internal use.

Only the latest copy of a given item-id is retained for a given remote file via the LFID Table. If a call is made to UPDITM with the "G" option (leave group locked) and there is no corresponding entry in the LFID Table for this item-id, a new entry is created which points to the starting FID of the UPDITM response received. The corresponding item-id is then written into this frame at the position it would normally have been for a RETIXU response.

3.2 THE FORMAT OF "RECORD" FOR REMOTE GROUP & ITEM LOCKS

When a call is made to RETIXU, RETIXI, RETIXUX, RETIXIX or UPDITM with the "G" option, RECORD is returned with the following format :

* Bit 2 is always set to 1.

* Bit 3 is 0 if in response to a request for group lock only. Bit 3 is 1 if in response to a request for an item lock.

* Bits 4 to 15 contain the index of the corresponding LFID Table entry for this user.

* Bits 16 to 31 contain the entry number in the BMS Table for the referenced remote file.

4.0 RELATED DOCUMENTATION

* Internal Specification for Remote File Access.

* Design Specification on Base, Modulo & Separation and Related Tables for Remote File Access.

* Design Specification for Retrieving Items from a Remote File.

* Internal Specifications on "UPDITM" System Routine.

THE ULTIMATE CORPORATION
Revision 0, for your comments
by C. Carmichael

The File-Server Processor, also known as the FSP, is the phantom process which provides transparent access to files and items for remote processes on the Ultinet communications network. The FSP runs in background mode and is initialized and started up at network generation time. It is assigned the third to the last communication line when the system is configured.

It is possible for the FSP to abort while servicing a data request from a remote node. To ensure that the FSP handles abort conditions correctly the FSP's USER tally in its PCB is set to -2. The system debugger then checks for that value in USER and sends the FSP to the appropriate error-recovery code. The FSP then sends the error message to the requesting node. The format of the response is:

1. Status = x'A000', a failed command response (2 byte).
2. Primary Command echoed back to requestor (1 byte).
3. Secondary Command echoed back to requestor (1 byte).
4. Cause = '2001' Aborted to debugger at the remote node.
5. Abort Type (1 byte):

    0 = ILLEGAL OPCODE
    1 = RTN STACK EMPTY
    2 = RTN STACK FULL
    3 = REFERENCING FRAME ZERO
    4 = CROSSING FRAME LIMIT
    5 = FORWARD LINK ZERO
    6 = BACKWARD LINK ZERO
    7 = PRIVILEGED OPCODE
    8 = REFERENCING ILLEGAL FRAME
    11 = RTN STACK FORMAT ERR

6. Program Counter, Register 1 (4 bytes).
7. Register Number, where applicable (1 byte).

The File-Server Processor, also known as the FSP, is the phantom process which provides transparent access to files and items for remote processes on the Ultinet communications network. The FSP runs in background mode and is initialized and started up at network generation time. It is assigned the third to the last communication line when the system is configured.

It is possible for the FSP to encounter a GROUP FORMAT ERROR while servicing a data request from a remote node. To ensure that the FSP reports this error correctly the FSP's USER tally in its PCB is set to -2. The standard system GFE interfaces then checks for that value in USER and sends the FSP to the appropriate error-reporting code. The FSP then sends the error message to the requesting node. The format of the response is:

1. Status = x'A000', a failed ~~command~~ response (2 byte).

2. Primary Command echoed back to requestor (1 byte).

3. Secondary Command echoed back to requestor (1 byte).

4. To-address (4 bytes).

5. From-address (4 bytes).

6. Cause = '2009' GFE encountered at the remote node.

Presently within the standard O/S routines there are 3 places which report GFE's. These are GFE, WSPACES-II, and ARITH. These modes will be modified to check for a -2 in the processes USER tally and return to the FSP's code. Two of these modes build information in the OB work area to be reported along with the GFE message. When this data is present it is also passed to the remote node as an ASCII string terminated with a x'FE'. If there is no message a x'FE' will be returned.

THE ULTIMATE CORPORATION
Revision 0, for your comments
by C.Carmichael

SPECIFICATIONS ON FILE-SERVER GFE HANDLING

Farokh Deboo
Revised February 6, 1984

THINGS THAT NEED TO BE FIXED/MODIFIED ON THE SYSTEM

1. Modify sections of code (eg. Basic CLEARFILE) that directly access local filespace to go through regular file access routines.

2. Write code to implement CLOSE for Basic.

3. Add CLOSE routines to application software packages?

4. Need to define interfaces for File-Restore, etc. routines that expect IR, R14, SR4, etc. pointing to end of group when RMBIT = 0 for a remote RETIX/GETITM.

5. May need changes similar to above for bisynch ?

6. Add check to prevent user from doing a GET-LIST from a Remote File in mode SAVE-LIST subrtn PFILEOPEN.

7. Add check in Basic Compiler COMO to prevent compiling a Basic program in a Remote File.

8. Add check in Basic Run Time BRP00 (after DICTOPEN in !RUNCAT) to prevent running a Basic Program compiled on a Remote File.

9. ISTAT needs to check for local/remote B, M, S before calling HASH.

10. FIX-FILE-ERRORS needs to check for local/remote B, M, S before calling HASH (mode GFMT2).

11. SEL-RESTORE should check for local/remote B, M, S after calling OPEN (mode DLOAD4).

12. Have a check in HASH for local/remote B, M, S before doing anything.

13. Change system routines like the assembler with chaining option, etc. as necessary to be able to work with the remote GETITM code.

14. Can a "SORT ONLY" be done with just a series of GETITM calls and no RETIX's on the sorted item-list as is currently being done?

15. Fix mode GFE for existing bugs.

16. EBASE, MOD, SEP must always point to the ERRMSG file (CJM suggestion).

I/GLOCK : RECORD = FID
T4 = Node #
T5 = PIB #
(BmsBEG → ItemID)

10-10-83

Group Locks

I. Tables

A. Eight tables with 46 locks each for a total of 368 group locks.

B. Table Structure

1. LOWER section of table (bytes 0-49)
   a. Byte 0 = Table lock byte
   b. Bytes 4-49) = LSB of FID (46 entries)
   c. Byte 50 = Lower Table Scan Delimiter

2. UPPER section of table (bytes 51-501)
   a. Forty-six 10 byte entries

Byte No.	Assignment
0	Status (1 byte)
1	Locked items counter (1 byte)
2-3	Item lock offset (2 bytes)
4-5	FIB of group requesting lock (2 bytes)
6-7	PIB (2 bytes)
8-9	Complementary Node number (2 bytes)

Item Locks

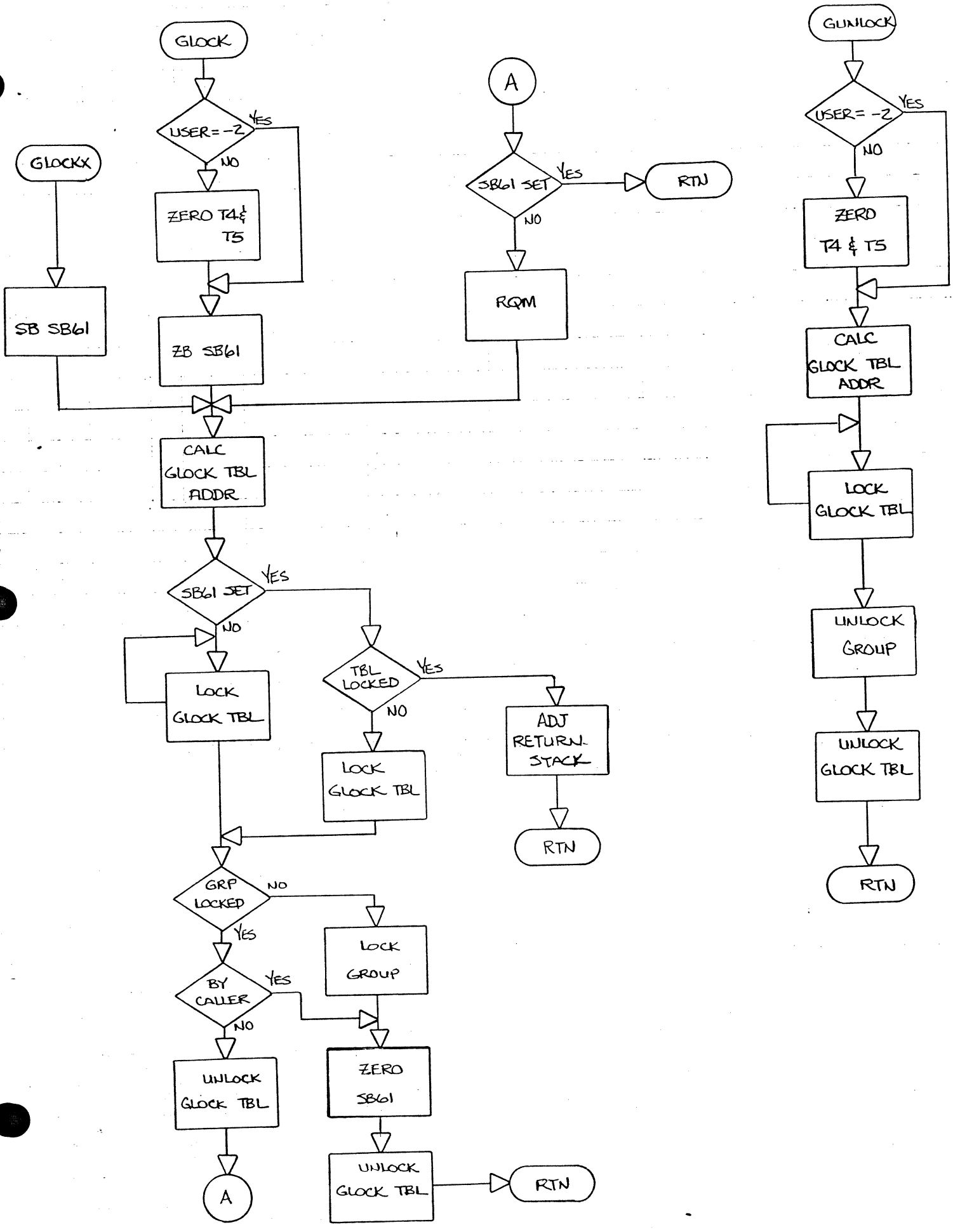
I. Table

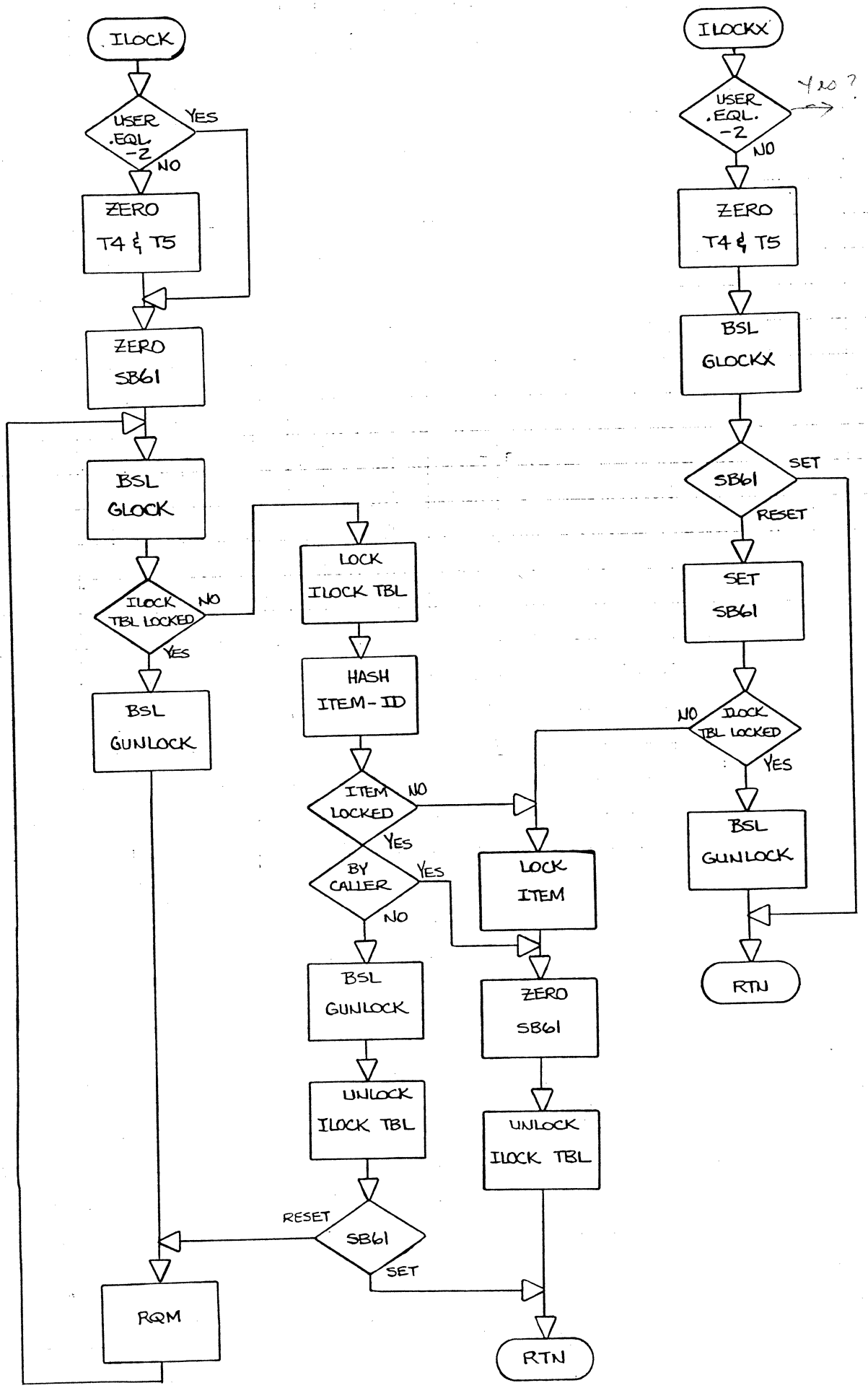
A. Twelve frames with 50 locks each for a total of 600 item locks.

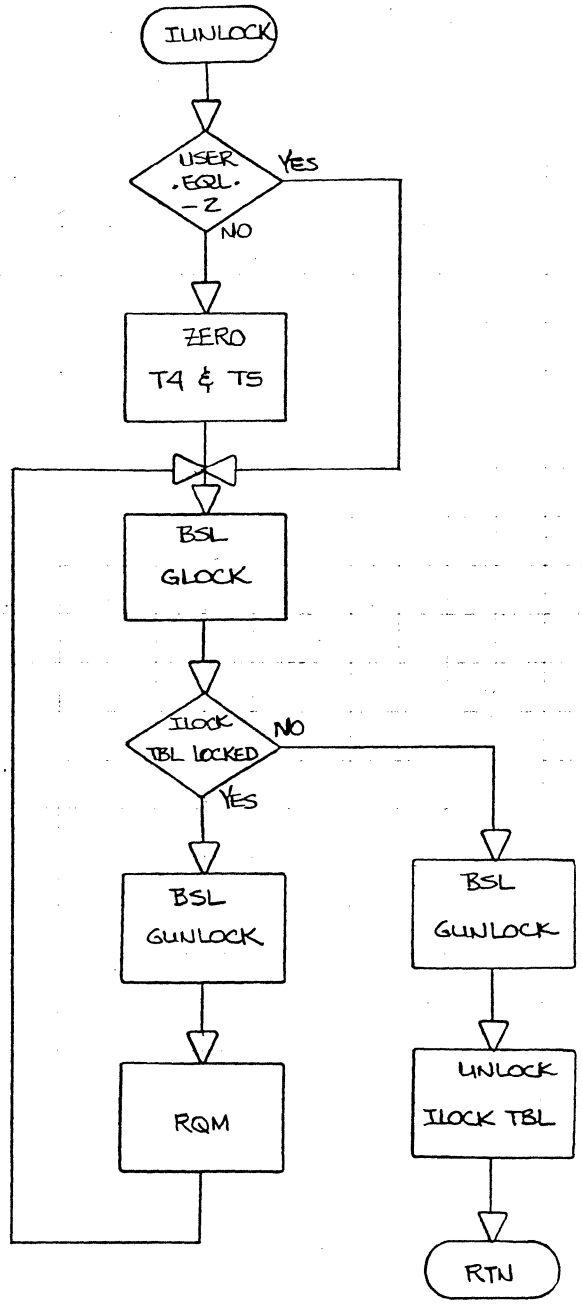
B. Table Structure (12)

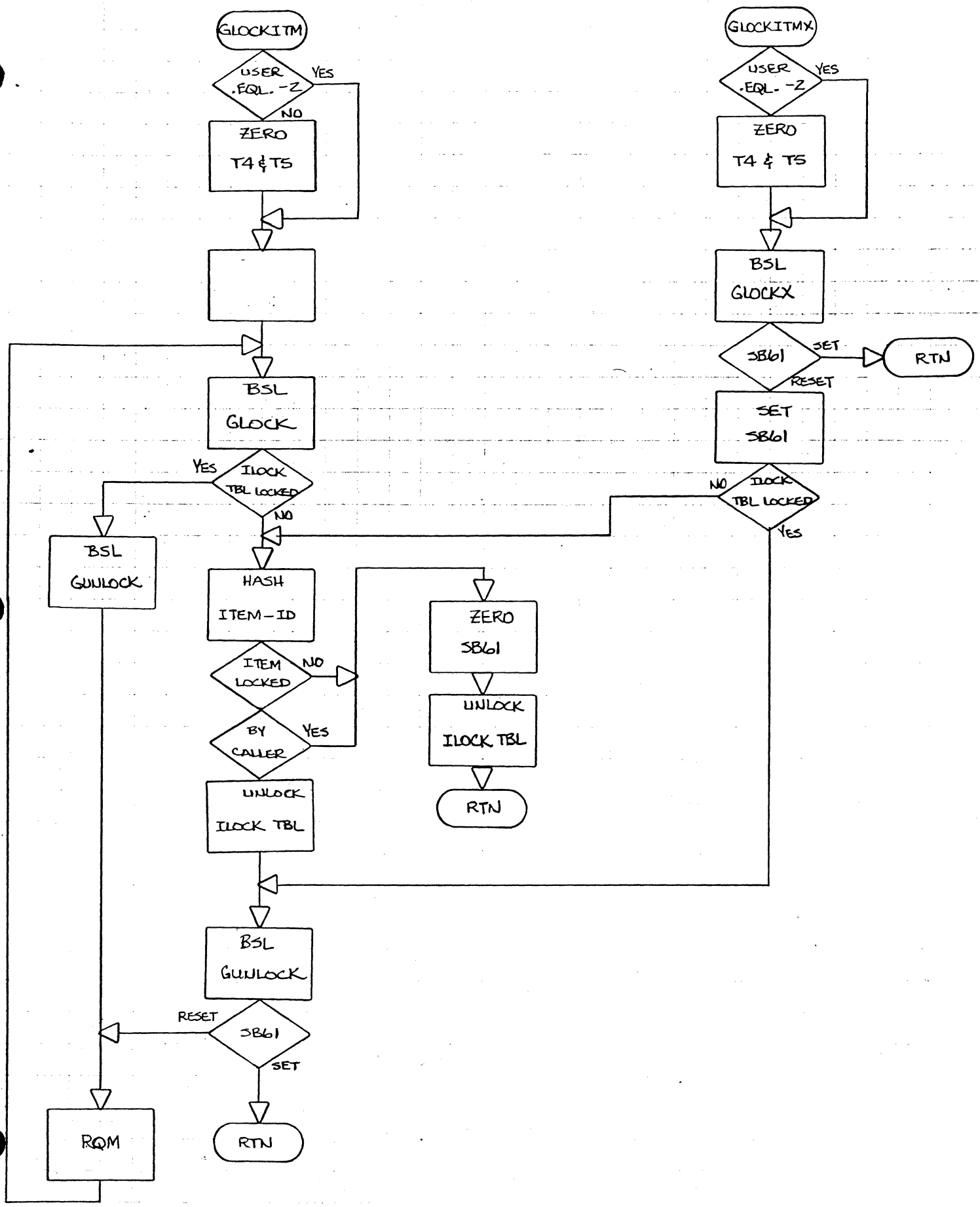
1. Bytes 0 - B = Frame Links
2. Bytes 0 - 511 = Fifty 10 byte entries
3. Entry format:

Byte No.	Assignment
0-1	PIB (2 bytes)
2-3	Forward link (2 bytes)
4-5	Complementary Node number (2 bytes)
6-9	Item-id Hash value


GLOCK
USER = -2
YES
NO
ZERO T4 & T5
GLOCKX
SB SB61
ZB SB61
A
SB61 SET
YES
RTN
NO
RQM
CALC GLOCK TBL ADDR
SB61 SET
YES
NO
LOCK GLOCK TBL
TBL LOCKED
YES
NO
LOCK GLOCK TBL
ADJ RETURN STACK
RTN
GRP LOCKED
NO
YES
BY CALLER
YES
NO
LOCK GROUP
UNLOCK GLOCK TBL
ZERO SB61
A
UNLOCK GLOCK TBL
RTN
GUNLOCK
USER = -2
YES
NO
ZERO T4 & T5
CALC GLOCK TBL ADDR
LOCK GLOCK TBL
UNLOCK GROUP
UNLOCK GLOCK TBL
RTN


ILOCK
USER .EQL. -2
YES
NO
ZERO T4 & T5
ZERO SB61
BSL GLOCK
ILOCK TBL LOCKED
NO
YES
BSL GUNLOCK
RQM
LOCK ILOCK TBL
HASH ITEM-ID
ITEM LOCKED
NO
YES
BY CALLER
YES
NO
LOCK ITEM
BSL GUNLOCK
UNLOCK ILOCK TBL
RESET
SB61
SET
ZERO SB61
UNLOCK ILOCK TBL
RTN
ILOCKX
USER .EQL. -2
NO
ZERO T4 & T5
BSL GLOCKX
SB61
SET
RESET
SET SB61
NO
ILOCK TBL LOCKED
YES
BSL GUNLOCK
RTN


ILUNLOCK
USER .EQL. -2
YES
NO
ZERO T4 & T5
BSL GLOCK
ILOCK TBL LOCKED
NO
YES
BSL GUNLOCK
RQM
BSL GUNLOCK
UNLOCK ILOCK TBL
RTN


GLOCKITM
USER .EQL. -Z
YES
NO
ZERO T4 & T5
BSL GLOCK
ILOCK TBL LOCKED
YES
NO
BSL GUNLOCK
HASH ITEM-ID
ITEM LOCKED
NO
BY CALLER
YES
UNLOCK ILOCK TBL
ZERO SB61
UNLOCK ILOCK TBL
RTN
BSL GUNLOCK
SB61
RESET
SET
RQM
RTN
GLOCKITMX
USER .EQL. -Z
YES
ZERO T4 & T5
BSL GLOCKX
SB61
SET
RTN
RESET
SET SB61
ILOCK TBL LOCKED
NO
YES

8-23-83

****
*
*
* U L T I M A T E C O R P O R A T I O N *
*
* SYSTEM SECURITY PROJECT PROPOSAL *
*
*
****

1 Logon Protection

Logon protection is divided into 3 major areas and its goal is to reduce the possibility of an unauthorized person being allowed entrance onto an ULTIMATE system. An account's password is encrypted, making it more difficult for someone to locate the password for an account. Logon attempts are recorded and lines (terminals) disabled when too many logons are attempted. This feature prevents someone or some device from trying many password combinations to gain entrance onto an ULTIMATE system. The user can define the lines that are allowed to logon to each system account. If someone knows the account name and password but doesn't have access to a line that allows entry to the account, he cannot logon.

1.1 Encryption of Password

When a user creates an account or updates the password associated with an existing account, the update account processor uses the account name and the password to calculate an encryption value that is checked by the logon processor to determine if a logon should be allowed.

The encryption algorithm is as follows:

a)The account name and password are concatenated and a cylic redudancy check (CRC) is made of the resulting string.

b)The four byte CRC result is expanded to an eight byte ASCII Hex version of the same number.

c)The account name - password string is XOR'd with the eight byte ASCII Hex string a byte at a time. The process being repeated until all bytes of the account name - password string are XOR'd. The resulting string is the first part of the encrypted password.

d)A CRC is taken of the string produced in c) above. The 32 bit original CRC is next XOR'd with the 32 bit CRC derived from the encrypted password. This is the second part of the encrypted password (hereinafter called the "key").

When the user attempts to logon, the logon processor performs the same encryption method on the user entered account name and password and compares the calculated encryption value with the encryption value stored with the account. If a match occurs, the user is allowed to logon. For this purpose the key is not used.

If it is desired to recover the password from the encrypted version, the following procedure is followed:

a)A CRC is taken of the first part of of the encrypted password. This value is XOR'd with the key. The result is expanded to an eight byte ASCII HEX value which is XOR'd repetitively with the first part of the encrypted character string until all characters are processed. The result is the original account name - password.

For this system to work, the encrypted data cannot be allowed to contain values of X'FB' or greater. This will not happen so long as the most significant bit in every original password byte is zero. This is true for all normal and displayable characters.

Currently, an account's password is stored in attribute 7 of the account's item in the system dictionary. The encrypted password value may be stored in a less recognizable way. Random data may be stored in other attributes of the system dictionary item. The encrypted password may be in a system dictionary item, but the operating system changed to not display it. A location other than the system dictionary may contain the passwords.

1.2 Recording of Logon Attempts and Line Disablement

The existing ULTIMATE system allows a user an unlimited number of logon attempts. This security enhancement restricts attempted logons by recording information related to the logon and disabling lines that attempt too many logons. A line characteristics processor is provided to prompt the user for all line parameters (ie. baud rate, parity) via a menu. The user can specify in the line characteristic information, that is stored in the dictionary of the ACC file, the acceptable number of logon attempts and the amount of time a line should be disabled that violates the acceptable number of logon attempts. The logon processor saves the line number, time of day, entered account name, and entered password in the ACC file for an unsuccessful logon attempt. This ACC file information may be secured by the MASTER account by not allowing any account (other than the MASTER account) access to the information. For a more detailed description of this MASTER account restriction, see the section entitled "Shutdown of Adjustments to Account Protection". The logon processor maintains a count per line of the total number of unsuccessful logon attempts over a certain period of time (ie. day). When a successful logon occurs on a line, the unsuccessful logon count is reset to the user specified count in the dictionary of the ACC file. If the count is exceeded on a particular logon attempt, the logon processor executes a monitor call (MCAL) instruction that puts the process (line) to sleep, disables the break key from waking up the sleeping process, and turns off the line's data terminal ready (DTR) signal. This line disablement affects the line that caused a runout of the logon attempt count. Other lines are still able to logon. After a specified period of time (ie. line disablement sleep time), the unsucessful logon attempt count is reset regardless of whether the line is enabled or disabled due to an unsuccessful logon attempt violation. For a description of enabling a disabled line by the MASTER account, see the section entitled "Enable a Disabled Line". The logon processor treats the MASTER account's password as an acceptable password for any system account. If the user attempts to logon to any system account, the MASTER account password can be entered and the logon processor allows the logon attempt.

1.3 Restriction of Accounts on Lines

The ULTIMATE system basically treats all system lines the same, providing the same capability on each line. If a person has access to a system line, he has the capability to logon to any system account, which is a problem for timesharing environments. This security enhancement allows the user to specify the lines on the local ULTIMATE system or the node names of the remote ULTIMATE systems that can logon to an account. Some or all of the following forms may be provided to allow the user flexibility in entering the line/node logon information :

* Range of Lines
* All Lines Within Range Except Line Number
* All Lines Except Line Number
* <NOT> Line Number
* Pattern Match on Node Name

When the user enters an account name at the "Logon Please" prompt or with the LOGTO verb, the logon processor retrieves the specified account's line/node logon information to determine whether the user's line is allowed to logon to the account. If the logon isn't accepted, the logon processor outputs the "User-Id" message and reprompts the user for a different account.

2 File Protection

File protection is divided into 2 major areas and its goals are to reduce the possibility of unauthorized persons from being allowed access to restricted files and to simplify the method of specifying restrictions on files. Adjustments to file definition items (D items) is restricted to a few select system processors, reducing the possibility of users bypassing file security by creating their own file definition items. The create file and create account processors incorporate menus, prompting the user for the file and account security parameters, and the update and retrieval lock codes contain the account names and node names that are allowed access to files. These features should provide an easier to understand method of specifying file security.

2.1 Update File / Account Processor

The existing ULTIMATE system doesn't provide a processor whose purpose is to adjust the parameters that define a file or an account. If a user wants to change the update or retrieval codes associated with a file or the password associated with an account, he edits the file's or account's D item and changes the appropriate attribute. When users are allowed to directly update D items, security is easily circumvented. Because one goal of this security project is to secure D items (see the section entitled "Securing of D and CC Items") by not allowing users to directly update D items, a processor is needed to allow the user to update parameters associated with a file or account.

This security enhancement provides a generalized D item updating processor that takes parameters, defining a file or account, that are passed to it, obtains file space from overflow if the file or account is being created (and not updated), and builds a D item that contains the parameters for the file or account. A special option of the D item updating processor, which is used by the create account process, treats the inputted parameters as parameters defining an account and updates the D item in the system dictionary. A capability of the MASTER account prevents system accounts, except the MASTER account, from adjusting a file's or an account's D item parameters by the D item updating processor (see the sections entitled "Shutoff of Adjustments to Account Protection" and "Shutoff of Adjustments to File Protection").

The create file process involves the execution of a basic program that prompts the user in a menu for the following parameters that define a file :

* Modulo and Separation or Estimated Number and Size of Items
* Update and Retrieval Codes
* Conversions and Correlatives
* Justification

* Length

* Reallocation Parameters (Attribute 13)

The modulo and separation parameters are only specified when creating a file and not when updating parameters associated with an existing file. If the user specifies the estimated number and size of items, the basic program automatically calculates the optimum modulo and separation. The basic program validates the inputted parameters and transfers control to the D item updating processor, that updates the file in the data base.

The create account process involves the execution of a proc or basic program that prompts the user in a menu for the following parameters that define an account :

* Account Name

* Update and Retrieval Codes

* Password

* System Privileges

* Modulo and Separation or Estimated Number and Size of Items

* System Lines / Node Names Allowed to Logon to Account

* Size of Workspace

* Justification

* Length

* Restart Option

* Recording of Accounting Information Option

The modulo and separation parameters are only specified when creating an account, not when updating parameters associated with an existing account. If the user specifies the estimated number and size of items, the proc or basic program automatically calculates the optimum modulo and separation. The proc or basic program validates the inputted parameters, indicates the parameters that are to be encrypted by the D item updating processor (ie. password), calls the D item updating processor that updates the account in the data base, and calls the copy processor to copy items from the NEWAC file to the account's master dictionary (for creating an account, not updating an existing account).

2.2 Securing of D and CC Items

If users can create file definition items (D items) with any base, modulo, and separation by using the editor, very little protection exists for file information. Even if an account has password protection and a file within the account has update and retrieval code protection, a user can easily bypass the protection. If a user is able to logon to any system account that has privileges to update items with the editor or to run a basic program and knows a file's base, modulo, and separation (from a STAT file listing), he can create a D item and access any item within the desired file, circumventing any file protection. If a user knows the system dictionary's base, modulo, and separation, he can create a D item to access any account's password. A user can experiment with various base values until he locates the desired file information. A user can run another user's basic program by creating a catalogued basic item (CC item) with the starting frame and size (number of frames) of the basic program's object code.

This security enhancement restricts the updating of file definition items (D items) to the create account, create file, and file restore processors and the updating of catalogued basic items to the basic compiler. No file definition or catalogued basic item updating is allowed by the editor or basic runtime. The create file and create account processors allow updates to existing file definition items, but don't allow adjustments to the base, modulo, and separation values. The file definition and catalogued basic item structures are slightly modified, having the count field's most significant bit set on. When the system encounters a file definition or catalogued basic item with the count field's most significant bit set off, it is treated as a meaningless item, not an item that defines a file or basic program. The system open processor (GBMS) doesn't establish the elements of BASE, MODULO, and SEPAR for the meaningless item, rejecting the open and disallowing retrieval of items from the file. To provide this non-standard item structure, a routine is implemented, which is separate but similar to the operating system's update processor (UPDITM), that updates items into the system data base and sets on the most significant bit of the item count field. The D item updating processor (see section entitled "Update File / Account Processor"), file restore processor, and basic compiler call this non-standard update routine, instead of calling UPDITM or using the history string update processor.

Before UPDITM updates an item, it calls RETIXU to lock the group in which the item hashes, locates the item within the group, and establishes elements (R6, SR4, SIZE) that define the location and size of the item in the group. The item's count field is loaded in SIZE. After calling RETIXU, UPDITM determines if size is negative (or the item count field's most significant bit is set on). If SIZE is negative, the update is rejected and an appropriate error message is displayed, before returning to the routine that called UPDITM. UPDITM can create a file definition or catalogued basic item (D or CC item), but the non-negative count field makes the item ineffective.

File save creates a file definition segment (D segment) on tape for a file definition item with a negative count field. If a file definition item doesn't have a negative count field, file save creates an item segment (I segment). When file restore processes a D segment on tape, it calls the non-standard update routine, which sets on the item count field's most significant bit. There are no differences in tape formats for file save and restore, allowing the user to upgrade to the system security release by restoring any of his existing file save tapes.

If a field support person must create a file definition item (D item) to recreate a damaged system's data base, he can use the editor to create an ineffective D item and use the debugger to set on the item count field's most significant bit to make the D item effective. Sometimes a user wants to remove a file when its integrity is questionable, but doesn't want to affect the integrity of the rest of the system data base. A special option of the delete file processor removes a file definition item and doesn't return the file's frames to the system overflow table.

2.3 Update and Retrieval Locks

A file's update and retrieval lock codes are located in attribute 5 and 6 of the file definition item. When a user logs on to an account, the logon processor loads the LOCKSR storage register with the address (1 byte before the item-id field) of the item in the system dictionary for the account. When a user attempts to retrieve file information, the open processor locates the retrieval lock codes of the user's account (pointed to by LOCKSR) and the file and allows the retrieval if the retrieval lock codes match. If the user attempts an update of the file information, the update lock codes must match. If a match doesn't occur, the retieval or update is rejected.

The update and retrieval lock code scheme becomes very complicated when many different combinations of users have different file access privileges. The condition results in the data base having many lock codes and no one understanding the restriction and privilege of each one. If there is suspicion that unauthorized people are using restricted files and the lock codes need to be adjusted, it can be difficult coordinating the lock code adjustment. This lock code scheme gets even more complicated when communications allow the connection of more than one ULTIMATE machine with the sharing of data bases.

This security enhancement maintains the concept of update and retrieval locks, but the lock codes contain a different type of information. The lock codes specify the account name(s) and/or node name(s) that can retrieve or update file information. When a user logs on to an account, the logon processor copies the account name and node number to the user's workspace and loads LOCKSR with the address of the copied information. When the user attempts to open a file for retrieval, the open processor determines if the user's account name and node number violate the file's retrieval lock codes. If a violation occurs, the file isn't opened. The

open processor converts a node name from a file's lock code to a node number by retrieving the item in the NODES file with the item-id equal to the nodes name. Attribute 1 of the NODES file item contains the node number. The I-AM item in the NODES file contains the node number for the local machine. The communication file server must maintain the interface of LOCKSR pointing at a user's account name and node number when calling the open processor. The MASTER account has unrestricted retrieval and update privileges for any file on the local node, which is accomplished by the open processor bypassing all lock code checks.

The update and retrieval lock codes have the following format :

Account Name \ Node Name] Account Name \ Node Name] ... ^

The following table shows different combinations of lock codes and explains their meaning :

Lock Code	Description
null	No Locks
SYSPROG or SYSPROG\	SYSPROG Account on Local Node Allowed Access
SYSPROG,DEANER\BLUE	SYSPROG and DEANER Accounts on BLUE Node Allowed Access
\GREEN	All Accounts on GREEN Node Allowed Access
DEANER\BLUE]CJM\GREEN	DEANER Account on BLUE Node or CJM Account on GREEN Node Allowed Access

Even though the lock code for no locks is null, the update file and account processor menus require that the user enter something more than carriage return or new line to specify no locks.

Pattern matches and <NOT> type conditions are provided for the account name parameters of the lock codes. The user can specify all accounts that have a specific pattern (ie. 1st 3 characters are "ABC") in the account name or all accounts except accounts with the specified account name(s).

An execution lock with the same format as update and retrieval locks could be specified in catalogued basic items (CC items). When a user attempts to execute a basic program, the basic runtime checks for a violation on the account name and/or node name in the execution lock of the catalogued basic item. If a violation occurs, the basic runtime rejects the attempt to run the program.

3 System Protection by the MASTER Account

The security of an ULTIMATE system revolves around the MASTER account, which allows the person, responsible for system security, to enable or disable system features on system accounts. This feature disabling capability provides flexibility in selecting the desired level of system security. Sophisticated users can find ways to bypass any security, if debugger privileges are provided. If the system data base contains sensitive information, the system's debugger (assembly code) privileges should be turned off. Other disabling features are offered that make the system more secure. If a system contains information, that anyone should be allowed to access, all features can be enabled.

3.1 Shutoff of Assembly Code Operations

This security feature removes the capability of performing any assembly code operations from any account, except the MASTER account. The shutoff includes all debugger functions other than the basic functions (END, OFF), the DUMP processor, the MLOAD processor, and the ABSLOAD processor. The break key is disabled during the ABSLOAD process to prevent users from adjusting assembly code with the debugger.

3.2 Shutoff of Adjustments to Account Protection

This security feature prevents the adjustment of account parameters (ie. password) located in the system dictionary and restricts the retrieval and update of information in the ACC file (ie. unsuccessful logon attempt information, line disablement information for logon attempts, charge units, and logon times) by all accounts, except the MASTER account. The disablement applies to the creation of new accounts and adjustments to existing accounts and allows adjustments to line characteristic information (ie. baud rate) other than the line disablement information in the ACC file.

3.3 Shutoff of Adjustments to File Protection

This security feature prevents adjustments to file update and retrieval lock codes by all accounts, except the MASTER account. The open processor allows special file access privileges for the MASTER account by circumventing file update and retrieval lock code checking, whether the file protection shutoff is enabled or disabled.

3.4 Enable a Disabled Line

When a line's number of unsuccessful logon attempts is violated, the violating line is put to sleep with the break key disabled. A LINE-ENABLE verb can be executed from the MASTER account to wake up a line that violated an unsuccessful logon attempt number, restoring the ability to logon from the line. If all lines are disabled, a coldstart is required. The user can prevent all lines from becoming disabled by not specifying unsuccessful logon counts for lines (ie. line 0) that the user always wants to be enabled.

4 Item Locks

The purpose of this security enhancement to the group lock table is three-fold. The first is the enlargement of the table beyond the 101 entries limit. The second is to add the necessary data to the table to handle multiple machine access requirements. The third is to add the capability of locking items. The group lock data in the present table is split into 2 parts. The first section holds the low-order FIDs of groups which are locked. This section is scanned for a match and then compared with the upper tally of the FID which resides as an offset into the second section of the table. The information presently stored within the second section is the upper tally of the group FID, a status byte, and the PIB number. The new group/item lock scheme is an expansion of the present approach. The major changes are the storage of additional group lock data and the inclusion of an item lock table to store item lock information. The following information is stored in the new group lock table :

* Status Byte (1 Byte)

  00 - This table entry isn't used.
  X1 - Item lock(s) are in effect.
  X2 - Group lock is in effect.
  1X - Locally requested group/item lock.

  Note : Item lock may precede locking a group, but an item may not be locked while group is locked.

* Locked Items Counter (1 Byte)

  This byte is used as a counter of the number of items which are currently locked within the specified group.

* Item Lock Offset (2 Bytes)

  This number is a byte displacement from the start of the item lock table (contiguous block of frames) to the top entry of the item lock chain associated with the group lock table entry. If the number equals X'FFFF', no item lock chain exists for the group lock table entry.

* Group FID (2 Bytes)

  This number always refers to the beginning frame id of a group/item which is locked on the local node. The low order byte is in the first section of the table and the upper tally is in the second section.

* PIB Number (2 Bytes)

  User requesting that the group be locked.

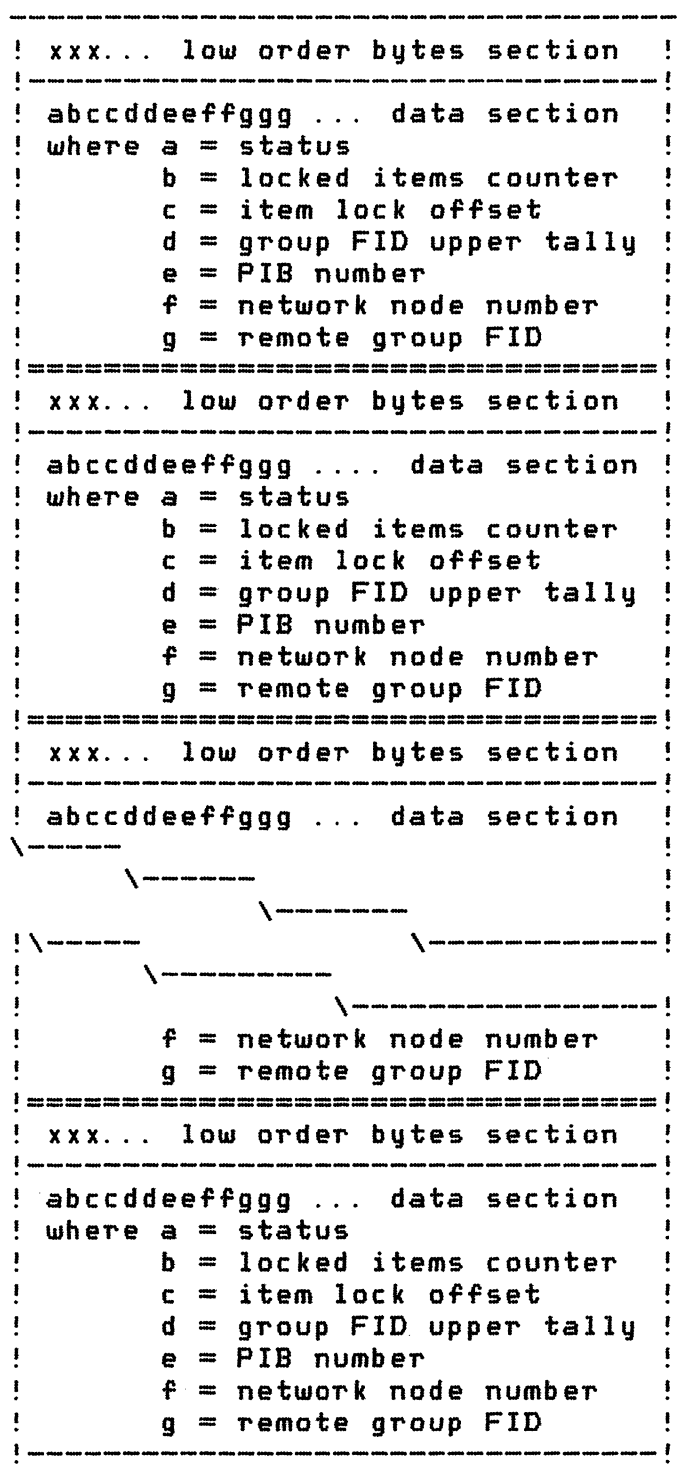
* Complementary Node Number (2 Bytes)

This number is only used with networking group locks and reflects the counterpart node involved in a networking transaction.

* Remote Group FID (3 Bytes)

This number is only used with networking and is the beginning FID of the group which is locked on a remote machine. A remote group FID only exists with status "1X", where the requesting node needs to have the remote group FID number in order to issue an unlock command to the remote node.

The 3 low order bits of the FID are used as an index into 1 of 8 preassigned group lock table frames. Each of these frames resemble the present group lock table in that each frame is divided into 2 sections. The first section contains the low order byte of each locked FID. The second section is indexed off the first section and contains the rest of the group lock data (see map on following page). Expansion of the $GLOCK byte into 8 lock bytes corresponding to each of the 8 group lock table frames may be necessary for efficiency.

Group Lock Table Frames	Format
0	
1	
2	
7	

Each of the 8 group lock table frames hold 36 entries for a maximum of 288 simultaneously locked groups. In addition to the group lock table frames, there is a contiguous block of linked frames which is reserved for item lock information. Each of the 228 group FID slots have a displacement from the start of the item lock frame block to the top entry of a chain of item lock entries, representing items locked within the group. If this displacement equals X'FFFF', no item lock chain exists for the associated group. A pointer to the top entry of a chain of available item lock entries is maintained to obtain unused item lock entries for the insertion of item locks into a group and to return unused item lock entries to the available chain for the deletion of item locks from a group. The following information is stored in the item lock table :

* Forward Link Displacement (2 Bytes)

  This number is the displacement from the start of the item lock table to the next entry in the item lock chain for the group. The last entry in an item lock chain contains a forward link displacement equal to X'FFFF'.

* PIB Number (2 Bytes)

  User requesting that the item be locked.

* Complementary Node Number (2 Bytes)

  This number is only used with networking item locks and reflects the counterpart node involved in a networking transaction.

* Item-Id Hash Number (4 Bytes)

  This number is a hash value derived from the item-id of the locked item. The HASH routine in the DISKFIO-I mode contains the hashing algorithm that should be considered to calculate the hash number. A hash number equal to zero could indicate that no item lock is contained in the item lock entry. If the hashing algorithm calculates a hash number of zero, the hash number is readjusted to a value of one. If multiple entries exist in a group's item lock chain and an item is to be unlocked from the chain, the IUNLOCK routine could zero the entry's hash number, instead of returning the entry to the available chain. When all item's are unlocked from the group (ie. item count = 0), IUNLOCK could return the group's entire item lock chain to the available chain.

Item locks are only used by the READU, READVU, MATREADU, WRITEU, WRITEVU, and MATWRITEU statements in basic. For the READU statement, the basic runtime pops the item-id off the stack and calls RETIXI. RETIXI locks the group and item and

locates the item in the file, pointing registers (ie. R6,SR4) at the item. If RETIXI cannot lock the item because the group or item is locked by another process, it hangs until the item can be locked. A RETIXIX routine is identical to the RETIXI routine, except it returns to the calling routine if the item cannot be locked. After returning from RETIXI, basic copies the item to workspace and calls GUNLOCK to unlock the group, leaving the item locked. Once the group is unlocked, however, other processes can update items within the group, invalidating the registers that point to the item. The rule for reading items for update is the item must be copied to workspace before unlocking the group.

For the WRITEU statement, the basic runtime pops the item-id and the new item body off the stack and calls UPDITMI. UPDITMI calls RETIXI, which locks the group and the item and locates the item within the group. After returning from RETIXI, UPDITMI updates the item in the file and calls GUNLOCK to unlock the group, leaving the item locked. After unlocking the group, UPDITMI can either return to the calling routine or unlock the item before returning to the calling routine. If unlocking the item is done within UPDITMI, the CH8 element is checked to determine if the unlock is desired. The basic runtime loads CH8 before calling UPDITMI. For WRITE, CH8 is loaded with "U". For WRITEU, CH8 is loaded with "G". UPDITMI unlocks the item if CH8 equals "G". UPDITMI calls IUNLOCK to unlock the item and returns to the basic runtime's calling routine. If the item isn't to be locked by UPDITMI, UPDITMI returns to the basic runtime calling routine after unlocking the group. The basic runtime checks the type of basic statement being executed and calls IUNLOCK if the item is to be unlocked.

The item and group lock routines may provide the ability to reserve entries in the lock tables. This feature prevents the condition in which a group is locked on a remote node but the lock cannot be honored because a group lock cannot be obtained on the local node. To prevent this inefficiency, the operating system reserves a lock on the local node before attempting to lock on the remote node. After the remote node lock is accomplished, the local node locking routines must be called to adjust the reserved status to a locked status and to provide the locking information. A problem exists because a FID is needed to lock a group and a local FID doesn't exist on a remote transfer until the transfer occurs.

The following examples are provided to aid in understanding group and item locks :

	Group Lock Table							Item Lock Table				
	Status	PIB #	FID #	ILtoecmk	Node #	Remote	FID #	Status	Item-id	Hash	PIB #	Node #
Node A, PIB 9 regular group lock.	02	09	1234	0	0	0						
Node A, PIB 9 requests group lock on node B.	12	09	2345	0	B	3456						
Node B honoring the request from Node A.	02	09	3456	0	A	0						
Node A, PIB 9 item locks 'ABC' (hash = 2512).	01	09	1234	1	0	0		01	2512		09	0
Node A, PIB 9 item locks 'DEF' on Node B (hash = 8125).	11	09	4567	1	B	9876						
Node B honoring the item lock request from Node A.	01	00	9876	1	0	0		01	8125		09	A

5 Tape Protection

This security feature provides protection of saving information from an ULTIMATE system to tape by requiring users to enter passwords. If a user attempts to save an account that has password protection in its system dictionary item, the account save processor prompts the user for the password. If the user enters an invalid password, the account save request is rejected. If the user enters the correct password, the account save request is accepted and the account save processor prompts whether the user desires to save a password on tape. If a password is desired on tape, the user can request to use the account's password or can enter a different password. The default (by entering carriage return) is to not save a password on tape. The user interface for account save has the following prompts :

```
>ACCOUNT-SAVE
PASSWORD:

FILE-SAVE TAPE LABEL =

ACCOUNT-NAME =
PASSWORD (Y/N):
PASSWORD OR [CR] (USE ACCOUNT'S PASSWORD):
```

If a user attempts a file save and the SYSTEM item in the system dictionary contains a password, the file save processor prompts the user for the password. If the user enters an invalid password, the file save request is rejected. Users on the MASTER account aren't required to enter passwords for account saves or file saves.

The account restore processor doesn't prompt the user for the password of the account on tape and accept or reject the account restore request depending on the entered password. However, the account restore processor does prompt the user for the password to be restored with the account onto the ULTIMATE system. The user can accept the password from the tape or can enter a different password. The default (by entering carriage return) restores the password from the tape. The user interface for account restore has the following prompts :

```
>ACCOUNT-RESTORE account-name

ACCOUNT NAME ON TAPE:
PASSWORD OR [CR] (USE TAPE'S PASSWORD):
```

6 Verb and User-Exit Protection

The ULTIMATE operating system has very little security associated with the execution of asembly code. If a user has system privileges 1 or 2, he is allowed to create and execute a verb that starts executing at any entry point in frames 1-4095. This capability allows a user to experiment with executing various assembly code entry points and to discover undocumented features of the ULTIMATE operating system. This security enhancement establishes a VERBS file in the MASTER account, which contains the verb definitions and user-exits that are allowed to be executed on the ULTIMATE system. When a user enters a TCL statement, the operating system retrieves the entered verb in the user's master dictionary. The retrieved verb from the master dictionary contains the item-id(s) of the item(s) in the VERBS file containing the verb's modeids (frame numbers and entry points of assembly code to execute). If the verb's item-id doesn't exist in the VERBS file, execution of the verb is rejected and the "VERB?" error message is displayed. If the verb's item-id exists in the VERBS file, the operating system obtains the verb's modeid in attribute 1 of the VERBS file item and starts assembly code execution at the modeid location. If the VERBS file contains the items with item-ids equal to the modeids in the user's verbs, the user's verb definitions (in the master dictionary) require no adjustments. Users can switch the numeric modeids in their verb definitions to symbolic names. After an item is added to the VERBS file with an item-id equal to the symbolic name and attribute 1 equal to the modeid location to start assembly code execution, the user can execute the verb with the symbolic name. After a user's verbs are transformed from using modeids to symbolic names, the user is never required to change his verb definitions (only the VERBS file needs adjustment). If verb definitions change on an ULTIMATE release, the upgrade procedure requires upgrading the MASTER account's VERBS file, not the user's verb definitions. Because verb and user-exit modeids are only in the VERBS file, a user is restricted to only start assembly code execution at the locations defined in the VERBS file, assuming the debugger privileges are disabled. These VERBS file items can be secured by update and retrieval lock codes or a restriction could be considered to only allow the MASTER account to update or retrieve information in the VERBS file. It may be desirable to have 2 verb definition files, one for ULTIMATE supported verbs released to customers and another for verbs developed and supported by dealers and customers (ie. USER-VERBS file).

7 Peek

This area needs to be addressed because the ability to view characters being input or output on another terminal allows users to obtain security parameters (ie. passwords) and restricted information in the system's data base.

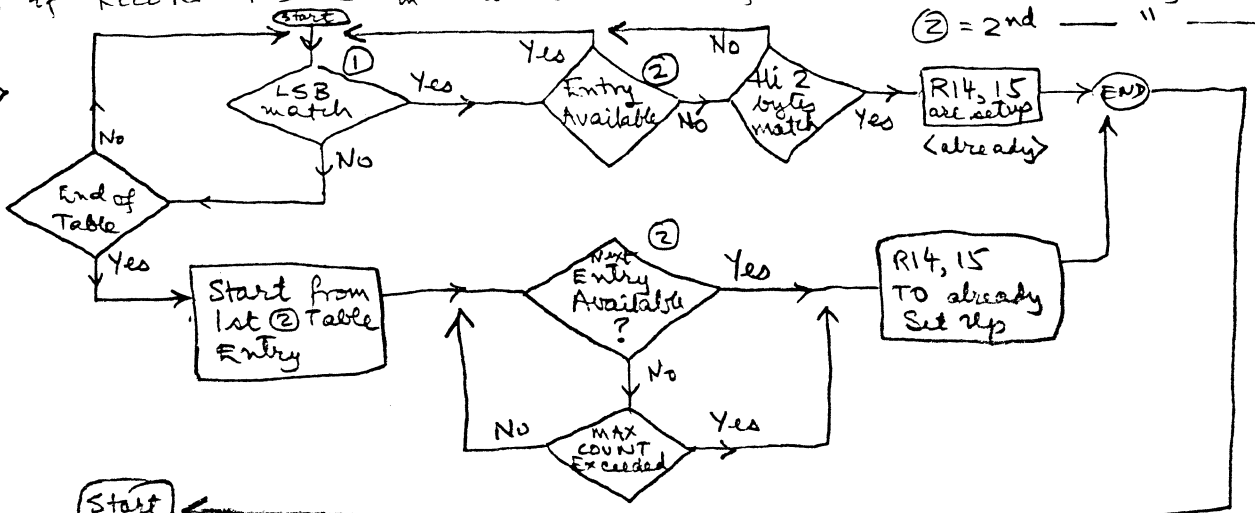
12-28-82

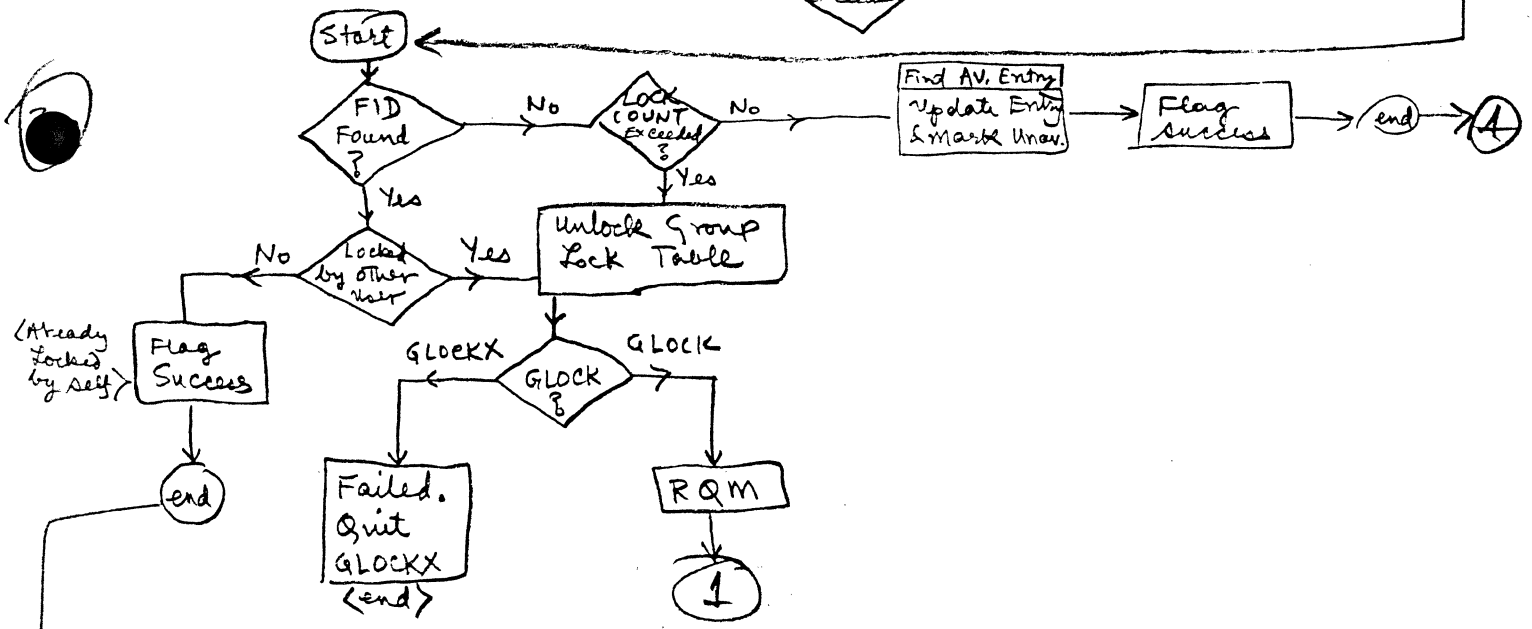
Farokh Deboo

Procedure: (Current) <! GLOCK / ! GLOCKX>

1. Locks the group lock table, etc. [R2 ; HO ← PIB⊕] <! LOCK>

2. Checks if RECORD FID is in the table as follows [① = 1st part of Table ② = 2nd — " —]





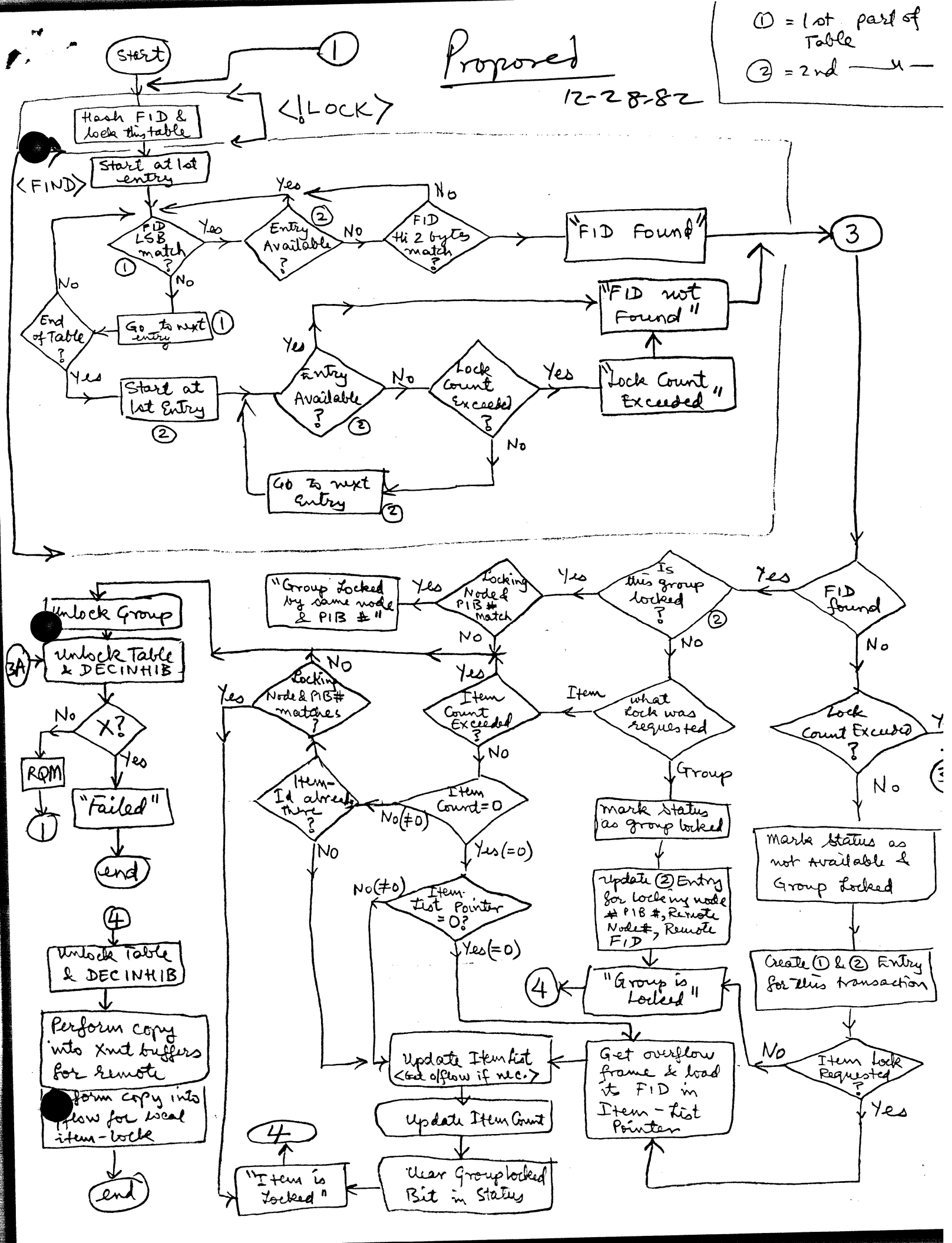
4. Unlocks the group lock table & decrement INHIBIT.

Proposed

12-28-82

① = 1st part of Table

② = 2nd —"—



* * * M E M O * * *

DATE: 4-26-83
TO: DENNIS A. AND FAROKH D.
FROM: CAROL C.
RE: COMM-UTIL FRAME IN THE SM FILE

THIS FRAME HAS 2 ENTRY POINTS DEFINED IN COMM-DEFS SO FAR:

$GET-I-AM WHICH RETRIEVES THE I-AM ITEM FROM THE NODE FILE.
$OPENNODE WHICH OPENS THE NODE FILE.

THERE ARE NO INPUT INTERFACES NEEDED. THE ELEMENTS AND OUTPUT INTERFACES USED ARE THE STANDARD ONES USED IN "OPEN" AND "RETIX". "SR1" IS USED WITHIN COMM-UTIL FOR TEMPORARY STORAGE. THE ROUTINE WILL EITHER RETURN TO THE CALLER IF SUCCESSFUL OR IF THERE IS AN ERROR IT WILL RETURN TO TCL WITH EITHER ERRMSG 2331 "NODE FILE MISSING" OR 2332 "I-AM" ITEM IN NODE FILE MISSING."

11-3-83

610	VNET0	1B3
611	VNET1	1B7
612	VNET2	1C9
613	VNET3	1D9
614	VNET4	1DB
615	VNET5	1B7
616	VNET6	147
617	VNET7	1B7
618	OPEN2	157
619	VNET9	195
630	VNET-RETIX	19D
631	VNET-LOCKS	12B
632	VNET-UPD	C3
633	VNET-GETITM	CD

Farokh Deboo
Aug 11, 1983

PRELIMINARY NETWORKING TEST PLAN FOR USER SIDE

I. NETWORK GENERATION

1. Verify that the BMSPTR entry in the COMM-CB gets loaded and the BMS Pointer Table is created and initialized to an all zeroes state.

II. MAKING A SESSION CONNECTION

1. Flush the input queue by attempting to retrieve Q-entries from the receive queue and taking the following actions:

   (i) If no Q-entry is retrieved, continue with the next step of sending a CONNECT-REQUEST.

   (ii) If a Q-entry is received from the node that corresponds to the connection that we are trying to establish, ignore this Q-entry and continue flushing the receive queue for this line.

   (iii) If the received Q-entry corresponds to another node than above, execute OTHER-SUB2 and continue flushing the receive queue for this line.

2. Send a CONNECT-REQUEST to the Remote Node addressed by the Q-Definition item.

3. Execute CHK-REPLY with time-out.

4. If a DATA Q-entry for this connection is received, ignore it and go back to CHK-REPLY with time-out.

5. If a Time-out occurs, send a DISCONNECT-REQUEST, release the BMS entry currently in use and exit. Use the NODENAME in the Q-Definition item for messages to the user.

6. If a non-DATA Q-entry is received for this connection take the following action. Whenever a NODENAME needs to be displayed in messages to the user, use the NODENAME in the Q-Definition item.

   (i) If a CONNECT-CONFIRM is received, flag the BMS entry to indicate that a connection has been established and start the application-level task at hand.

   (ii) If one of the expected DISCONNECT-INDICATION's is

received, release the BMS entry in use and exit.

(iii) If a DISCONNECT-INDICATION with an unexpected CAUSE field or any other non-DATA Q-entry is received, send a DISCONNECT-REQUEST, release the BMS entry in use and exit.

III. REMOTE OPEN

It is assumed that OPEN has detected a Remote OPEN operation at this stage.

1. The first time OPEN is performed within a given session, the following points need to be verified:

   (i) Make sure that the BMS Pointer Table has been created at Network Generation time. If not created, exit without doing any more remote processing

   (ii) The CSB and the BMS Table + Scratch Frame are created from overflow space. The BMS Table Header should be initialized to an all zeroes state and the environment should be saved in the CSB.

   (iii) If the Overflow space is not available the standard NOSPACE routine should be executed.

   (iv) If the NODE file cannot be opened for whatever reason, the overflow space obtained should be returned.

   (iv) If successful, the Pointer Table Entry should be updated for this line.

2. Exit if NODE File has missing/invalid item corresponding to the NODENAME in the Q-Definition Item.

3. Grab the next available entry in the BMS Table, unless exceeded the maximum number of OPEN Files allowed in which case exit.

   . Need more details for this check

4. Check that the VIROVF XMODE entry works while grabbing entries in the BMS Table.

5. Go through the section on MAKING A CONNECTION if a connection doesn't already exist with the referenced remote

node. Otherwise, fall through.

6. Format the OPEN command. Check the IS string for "," in case of a DICT,DATA entry in the IS string. Transmit to the addressed node.

7. If SAVE = 1, grab a secondary BMS entry. Check that the VIROVF XMODE routine works. Make sure that if the OPEN command has to be retransmitted (eg. for running out of disk space condition) that another secondary BMS entry is not grabbed.

8. Execute CHK-REPLY without time-out to see what response has come through.

9. If a non-DATA Q-entry is received, execute OTHER-SUB (which will cause a disconnection). Release the BMS entry (entries) and exit.

10. Execute ANL-RESP on the received response. If the subroutine returns here, it means that none of the error conditions expected by ANL-RESP have occurred.

11. Disconnect and release the grabbed entries if received a response with

    (i) Bad STATUS field (only high byte being checked here), or

    (ii) Incorrect length.

12. If the STATUS corresponds to success, do the following

    (i) Copy out parameters corresponding to the first OPEN-FILE entry received (there could be two).

    (ii) Release the secondary BMS Table entry if no longer required.

    (iii) Otherwise copy out the parameters corresponding to the second OPEN-FILE entry received.

    (iv) Set up BASE, FBASE, DBASE, etc. as necessary

    (v) Point IR into the Q-Definition item as follows

        (a) After the File Name if it is a single File specification.

        (b) At the comma between the DICT,DATA file specification if IS was also pointing to a DICT, DATA specification.

        (c) After the DICT,DATA if IS was pointing to a single file specification.

(vi) BMSBEG points one before the File name as follows

(a) If IS was pointing to DICT,DATA it will be the DATA file name in the IS string.

(b) If IS had a single level file specification but the Q-Definition item had a DICT,DATA File name, it will be the DATA File name in the Q-Defintion item.

(c) If both IS and the Q-Definition item had single level file specifications, it will be the File name in the Q-Definition item.

(vii) Point BMS to the last character of the File name copied into the BMS workspace.

(viii) Point IS at the character (normally blank) following the File name in the IS string.

(ix) Set RMBIT = 1.

13. If the STATUS corresponds to failure, do the following

(i) If the ERROR# is not one of the expected ones, disconnect. Release the BMS entries grabbed. Mark all others belonging to this connection as disconected and unavailable.

(ii) If the ERROR# does not correspond to "run out of disk space at remote node" or "entered debugger at remote node", release the grabbed entries but don't disconnect. That means that the BMS Table must have at least one connected entry for this connection, even if it is not associated with an OPEN File.

(iii) If failure due to "run out of disk space at remote node", resend the OPEN command if user wants to try again. Else branch to GO.END which is responsible for an orderly wrap-up of all activities including CLOSing all remote files OPENed by this line.

(iv) If failure due to entered debugger at remote node, display the details of the abort, disconnect, release the grabbed entries in the BMS Table and mark all other entries for this connection as discnnected and unavailable. If the reason for the abort doesn't match one of the expected values, the response will be treated as an illegal response.

(v) The failure due to remote GFE needs to defined.

14. Receiving a response with an illegal status field will

cause a disconnectipn with the referenced node. The grabbed BMS Table entries will be released and all other entries for this connection will be marked as disconnected and unavailable.

IV. REMOTE CLOSE (single file)

1. Execute CHK-LOCAL. If referencing a local file quit. If referencing a remote file but there is an error, the reason for the error is displayed and then the routine quits. If there was no error, continue.

2. Send a CLOSE command.

3. Execute CHK-REPLY without time-out.

4. If non-DATA Q-entry received, execute OTHER-SUB and quit.

5. Otherwise, execute ANL-RESP.

6. If status of response indicates success, do the following

   (i) If any of the following errors occur, disconnect with the referenceed node, mark all the corresponding BMS entries as disconnected and unavailable:

      (a) Response has illegal length, or

      (b) OPEN-FILE entry # incorrectly echoed.

   (ii) Release the chain referenced by OVFSTRT. + Unlock

   (iii) Leave the connection up. This is done as follows

      (a) If there are other BMS entries using this connection, release the entry corresponding to the file CLOSEd.

      (b) Otherwise, flag the entry to indicate CONNECTED but not associated with any OPEN file.

7. If status of response indicates failure, currently only checking for "entered debugger at remote node" failure. Any other failed responses treated as illegal responses.

8. If status is neither of above, received an illegal response. Generate a DISCONNECT-REQUEST and flag all the corresponding BMS entries as disconnected and unavailable.

V. CLOSE ALL FILES

1. If BMS Pointer Table not created, exit.

2. If Pointer Entry for this line doesn't exist, exit.

3. Send DISCONNECT-REQUEST's to all the connected nodes and release any overflow space accumulated for these connections.

4. Release the BMS Table and the CSB to overflow and flag this line's pointer entry to indicate that the tables are no longer set up.

XX. MISCELLANEOUS SUBROUTINES

XX.1 OTHER-SUB2/OTHER-SUB

OTHER-SUB is identical to OTHER-SUB2 except that the user has already ascertained that the received Q-entry is a non-DATA Q-entry.

1. If the Q-entry corresponds to a connection that no longer exists, ignore it.

2. If the Q-entry is from a line other than the File Server Process, send a DISCONNECT-REQUEST to the File Server Process of that node, mark all BMS entries corrsponding to this connection as disconnected. Do not release these BMS entries.

3. If a DATA Q-entry is received, send a DISCONNECT-REQUEST and mark all BMS entries corresponding to this connection as disconnected. Do not release these entries from the BMS Table however.

4. If one of the expected DISCONNECT-INDICATIONS's is received, mark all BMS entries corresponding to this connection as disconnected but unavailable.

5. If a DISCONNECT-INDICATION with an unexpected CAUSE field or any other Q-entry is received, send a DISCONNECT-REQUEST and mark all BMS entries corresponding to this connection as disconnected but unavailable.

XX.2 CHK-REPLY

This routine runs with or without time-outs depending on the value of the TMOUT-BIT.

1. If the received Q-entry if not for the node corresponding to this connection, execute OTHER-SUB2 and examine the next received Q-entry.

2. If the received Q-entry is from the correct node but not from the File Server Process, send a DISCONNECT-REQUEST to the File Server on that node, mark all the corresponding BMS entries as disconnected and unavailable, and exit. If CHK-REPLY called from within OPEN, the currently grabbed BMS entries are released.

3. If checking for time-out and a time-out does occur, exit.

4. If the received Q-entry is from the File Server Process of the correct node, exit.

XX.2 ANL-REPLY

If any of the following errors is detected, the subroutine will not return to the calling program but will send a DISCONNECT-REQUEST (and if called from OPEN, will release any grabbed BMS entries for that particular OPEN call). BMS entries other than those grabbed during an OPEN will be marked as disconnected but will not be released. If none of these errors is detected, the subroutine will return to the calling program.

1. Response Length less than that ~~specified in C-LENGTH.~~ MIN-LENGTH

2. TYPE-BIT = 0 (All responses must have STATUS field with TYPE-BIT = 1).

3. Responses with incorrectly echoed primary and secondary command fields (except for COMMAND-REJECT responses due to illegal primary or secondary commands received at the remote node).

4. Receiving COMMAND-REJECT responses of incorrect length.

5. Receiving COMMAND-REJECT responses that have an illegal ERROR#.

XX.3 CHK-LOCAL

1. If the referenced file is flagged as local, no processing is carried out.

2. If remote, check for the following error conditions :

   (i) No pointer Table created (ie. network generation no longer valid).

   (ii) The Pointer Table entry for this line has not been set up (this is done the first time a remote OPEN is executed).

   (iii) If trying to access a file OPENed by another line.

   (iv) Illegal displacement into BMS Table specified in BASE (either less than or equal to zero, or greater than LASTENT).

   (v) The displacement specified by BASE into the BMS Table corresponds to an unOPENed file.