DATABASE AND TERMINAL MANAGEMENT FUNCTIONAL DESIGN SPECIFICATIONS IN SUPPORT OF STOCK POINT LOGISTICS INTEGRATED COMMUNICATION ENVIRONMENT (SPLICE)

Joseph N. Reinhart
Ricardo Arana

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DATABASE AND TERMINAL MANAGEMENT
FUNCTIONAL DESIGN SPECIFICATIONS IN
SUPPORT OF STOCK POINT LOGISTICS INTEGRATED
COMMUNICATION ENVIRONMENT (SPLICE)

by

Joseph N. Reinhart III

and

Ricardo Arana

June 1982

Thesis Advisor: N. F. Schneidewind

Approved for public release, distribution unlimited

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Database and Terminal Management Functional Design Specifications in Support of Stock Point Logistics Integrated Communication Environment (SPLICE)	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, June 1982
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Joseph N. Reinhart III, Ricardo Arana	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	12. REPORT DATE: June 1982
	13. NUMBER OF PAGES: 109
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

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

18. SUPPLEMENTARY NOTES

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

Distributed System(s)
Terminal Management
Local Area Network
SPLICE
Data Base Management System(s)
Stock Point Logistics Integrated Communication Environment

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

As a result of the growing demands for Automated Data Processing at the Navy Stock Points and Inventory Control Points, long range plans are being developed around the Stock Point Logistics Interface Communications Environment (SPLICE) concept. This thesis examines the initial design and implementation of the SPLICE Local Area Network (LAN) as proposed by NAVSUP and FMSO, and provides an alternative

20. ABSTRACT (continued)

design and implementation strategy. The alternative design and implementation strategy recommended by this thesis is designed to provide a fully distributed architecture for a Local Area Network and, in particular, the Terminal and Data Base Management components.

Approved for public release, distribution unlimited.

Database and Terminal Management Functional Design Specifications in Support of Stock Point Logistics Integrated Communication Environment (SPLICE)

by

Joseph N. Reinhart, III
Lieutenant, United States Marine Corps
B.S., U.S. Naval Academy, 1977

and

Ricardo Arana
Lieutenant, Peruvian Navy
B.S., Peruvian Naval Academy, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
June 1982

ABSTRACT

As a result of the growing demands for Automated Data Processing at the Navy Stock Points and Inventory Control Points, long range plans are being developed around the Stock Point Logistics Interface Communications Environment (SPLICE) concept. This thesis examines the initial design and implementation of the SPLICE Local Area Network (LAN) as proposed by NAVSUP and FMSO, and provides an alternative design and implementation strategy. The alternative design and implementation strategy recommended by this thesis is designed to provide a fully distributed architecture for a Local Area Network and, in particular, the Terminal and Data Base Management components.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

Stock Point Logistics Integrated Communication Environment (SPLICE) is designed to augment the existing Navy Stock Point and Inventory Control Point ADP facilities which support the Uniform Automated Data Processing System--Stock Points (UADPS-SP). The hardware for the UADPS-SP consists of the Burroughs Medium Size (B-3500/3700/4700/4800) Systems. At present there are twenty new application systems being developed, as shown in Figure 1.1, which will require considerable interactive and telecommunication support. The current UADPS-SP cannot support these requirements without a total redesign effort and will probably require future replacement of the current mainframes. At present, individual project managers are developing the new application systems utilizing a variety of minicomputers which are capable of supporting the required interactive and telecommunications capabilities. This is being done due to the near term needs of the Navy and are scheduled to be implemented within the next four to five years.

There are two driving forces behind the development of SPLICE. First, there is the increased need for the use of CRT display terminals to interact with application logic and

ACRONYM	TITLE
APadE	Automation of Procurement and Accounting Data Entry
CAB	Centralized Accounting and Billing
CLAMP	Closed Loop Aeronautical Management Program
CS	Current Supply
DOSS	Disk Oriented Supply System
E&F	E&F TO Disk-Financial Improvement Project
FAMMS	Fixed Allowance Management and Monitoring System
ICP	Inventory Control Points
IDA	Integrated Disbursing and Accounting
MAPS	Multiple Activity Processing System (Satellite Processing)
MISIL	Management Information System International Logistics
NATDS	Navy Automated Transportation Data System
NAVADS	Navy Automated Transportation Documentation System
NAVSCIPS	Navy Standard Civilian Payroll System
OLA	On Line Autodin
OPTAR	Operating Target Accounting (TRIDENT)
RIP	Receipt Improvement Project
LDS	TRIDENT Submarine Logistics Data System
TOPS	Transportation Operational Personal Property Standards
RM & ME	Requisition Monitoring and Material Expediting

Figure 1.1 Projects Under SPLICE Umbrella

to fetch information from the systems database. Secondly, there is the need to standardize the multitude of interfaces currently existing across approximately sixty supply sites. Centralization of standard processors is desired to help reduce the overall costs of the SPLICE system in terms of processor support, the needed controlled environments and access control. Since the implementation plans have not yet been developed, the SPLICE processors are perceived to be co-located with the Host Burroughs system at each Navy Stock Point (SP) and with the Burroughs and Univac systems at the Inventory Control Points (ICPs).

SPLICE will provide economical and responsive support capabilities for a decentralized telecommunications environment. A "foreground/background" concept will be implemented with SPLICE minicomputers, which will serve as a front-end-processor for the Burroughs Systems via a Local Computer Network (LCN) interface. The term "Local Computer Network" (LCN) is a term used in the NAVSUP and FMSO documentation, but will be referred to as a Local Area Network (LAN) for the remainder of this thesis. The Burroughs will provide background processing functions for large file processing and report generation [Ref. 1]. In order to provide for greater software and hardware maintainability, SPLICE will be developed independently of the Burroughs systems using a standard set of minicomputer hardware and software. This is particularly important when considering the fact that SPLICE will be implemented at some

sixty different geographical locations, each having a different mix of application and terminal requirements.

B. OBJECTIVES OF RESEARCH

The objectives of the SPLICE project at the Naval Postgraduate School, Monterey, California, is to develop alternatives for SPLICE Local Area Networks. This thesis particularly concerns itself with the development of functional design specifications for the implementation of the Data Base and Terminal Management aspects of a functionally distributed Local Area Network in response to the requirements of the Naval Supply Systems Stock Points and Inventory Control Points.

C. PROPOSED NAVSUP IMPLEMENTATION

This section represents the implementation concepts as developed by NAVSUP and FMSO, and not those of the authors of this thesis. The information contained in this section was obtained from a series of reference documents obtained from NAVSUP and FMSO, and is included for background information only.

In terms of Terminal and Data Base Management, the SPLICE processor software will consist of the following "off the shelf" capabilities to be included with the delivery of the SPLICE hardware: Operating System, Processor to Processor interfaces, File Management System, Command Interpreter, Cobol Compiler, Performance Monitor, Report Writer and a Debug Facility. Other SPLICE software requirements will be fulfilled

by either the Terminal Management or Peripheral Management components on the Local Area Network as proposed by this thesis.

The Operating System, which would be provided by the vendor, will include the following features: multiprogramming/ multitasking management, memory management, priority interrupt recognition and scheduling, resource allocation, program to program message management, automatic rerouting of I/O in the event of a failure and reassignment of processes in the event of CPU failure. The requirement of automatic rerouting of I/O and reassignment of processes in the event of a CPU failure is applicable for stated SPLICE high reliability requirements. These requirements become apparent in considering Figure 1.2 which was taken from the SPLICE Software Design documentation of 19 March 1979. This figure is representative of NAVSUP's proposed concept of the SPLICE architecture.

The Processor to Processor interface manager is required to provide high speed, fault tolerant communication between processors on the SPLICE complex. It is designed to provide support for program to program message traffic, the use of shared database accesses and control information in support of user process migration from one processor to another without the need for knowledge of the processes residence within the SPLICE complex.

The file management system for SPLICE will provide device independent interface between I/O, the Operating System and

processing programs. The file management system must be capable of supporting multi-volume files, shadow volumes and volume back-up and restoration functions. It must be capable of providing record level lockout and multiple accesses to files for multiple users. In addition, support capabilities for variable and fixed length record formats must be provided, while the actual location of the data is transparent to the individual user program.

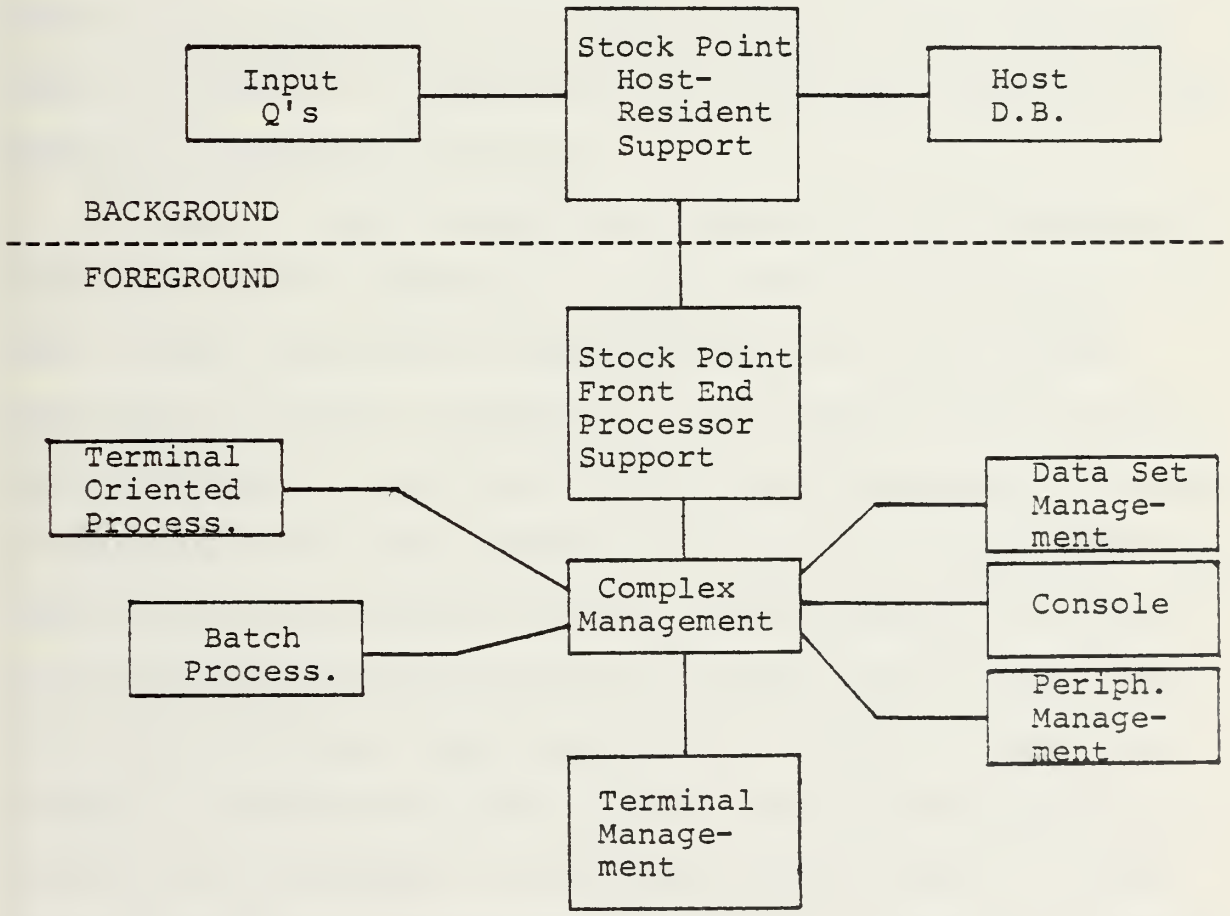


Figure 1.2 SPLICE Software Functions.

A command interpreter must be provided for user operator interface to the SPLICE system complex. The capability must be available from a program module in addition to independent command language interfacing between the user's and the SPLICE system complex. In order to provide for the operational control of the SPLICE complex, while permitting multiple user implementation of command functions, it is necessary for at least two levels of security to be provided: primary system operator control and system user control. The editor should provide the capabilities to create and update user files and should be capable of operating in an interactive mode with the terminals attached to the LAN.

An ANSI 74 Cobol standard is scheduled to be implemented across all SPLICE complexes. In addition to the required ANSI 74 Cobol standard, an appropriate high level program development language and debug aid are required for user program development. The high level program development language is required to be either PASCAL or a vendor oriented ALGOL-like compiler and must meet the minimum levels required to support the Terminal Application Processing System (TAPS), currently being used at various Navy Stock Points. TAPS, if used, is designed to provide both the terminal and database management requirements for SPLICE. In addition to the above, a report writer capability is specified, and if it is other than RPG II, a conversion program must be provided. [Ref. 1].

The central coordinating mechanism employed between each of these functions concerns the Complex Management Component (CMC) of the TAPS system. The CMC coordinates message traffic between the background Burroughs and the foreground terminals, and the allocation of SPLICE resources to include Peripheral Management, Data Set Management, Terminal Oriented and Batch processing, and the SPLICE Complex Operator Console. In addition to maintaining status information on foreground and background processes, the CMC uses a Postmaster Component (PC) which is controlled by the CMC. The PC is called into main memory by the CMC and is used to post messages/records to the Mail Box Storage (MBS) facility. The MBS facility is used by the CMC to temporarily store non-interactive messages destined for the off-line processors and non-interactive responses from a host destined for terminals. [Refs. 1, 2, 3].

There are eight software functions which have been defined for development in order to support the SPLICE concept in addition to the above stated processor software capabilities. The eight software functions are: Terminal Management, Terminal Applications, Data Set Management, Peripheral Management, Batch Applications, Complex Management, Stock Point Front-End-Processor support, and Stock Point Host-Resident support. Each of these functions operate under the control of the Complex Management function, thus providing for a tightly coupled system design. The first six functions are used to provide an application independent environment for LAN processing.

The last two functions are used to support the Stock Point Host interface for foreground/background communication. It must be recognized that the entire SPLICE concept, as presented in the SPLICE System Specifications (Phase I), 2 Feb 1981, and Software Design, 19 March 1979, revolves around a predetermined desire to use TAPS, which is marketed by the Decision Strategy Corporation. The actual implementation characteristics of TAPS can be found in the appropriate vendor documentation.

The TAPS software consists of two major functions which are used to support terminal application functions: processing support and data base management. Terminal application functions concern three general areas: Terminal Oriented Processing, Terminal Data Base Management, and Data Collection. The processing support keeps track of process status information for each terminal, while providing the needed terminal interface requirements particular to individual terminal characteristics. In addition, the TAPS Data Manager (TAPS/DM), provides predefined modules which can be accessed by terminal users without requiring predefined access modules. TAPS maintains the independence of the terminal user and application program from the physical data structure. This is accomplished through the use of the Application Manager (TAPS/AM) which passes predefined transactions to either the command processor, where data retrieval is managed, or to the screen phases defined by the application programmer.

Regardless of whether a user's process modules interface with information from the display screen or the data base, only specific data fields are presented.

The Terminal Manager is responsible for supporting a variety of terminals on the system, and interfacing the data to the system in a standard format. The Terminal Manager uses system tables to control the terminal interfaces with as little operator and application program intervention as possible. Output queuing and frame management will be provided according to the type of data being managed. Functions, such as opening a terminal at a predefined time and automatically retrying and timeouts, will be implemented. A set of terminal queues will also be available for output. Parameter tables, from the environment file, will be used to initialize the control structures for terminal management, to validate terminal access rights, and to control data manipulation.

The configuration table will identify the communication lines attached to the processor. The unique id of each end point associated with the line will be specified, unless the line is of the dial up variety. The configuration table will provide a security restriction for a line which would override the security restriction specified for a unique-id. Automatic communication control will be extended to monitoring terminal usage and directing data to alternative paths. An option to log off terminals which have not been active within a specified time period is to be available. Thus, output directed to one

terminal during prime time may be directed to a different terminal on an off shift. Values from the unique-id table are to be used in the building of the terminal work area at initialization and during connection of dial-up lines. The unique-id entry defines the specific hardware unit in terms of communication management and functional usage. A line connection validation is to be performed for dial-up lines.

The log-on options will vary according to the type of terminal or processor present. The user must provide a user-id and password which will be validated against a user-id table. The characteristics from the terminal's work area will define what processes are allowed from that terminal. The system-id to which the user requests access will indicate what function is desired and the system-id table will provide the mask for the functional type.

Most terminals will operate in either a message mode or a stream mode. In the message mode, each input and output is independent of the other message traffic for the terminal. In the stream mode, all input is directed to a specific destination and only output from that destination will be returned to the terminal. Output from any other origin will be queued for later delivery.

Each communication end-point will have a process id associated with its input. Terminals will be pointed to the operator interface until they are successfully logged-on to the LAN. The term "end-point" as used here represents a particular

user terminal. The log-on procedure will normally cause the process-id pointer to be changed so that all further input from the end-point is routed directly to the desired process. Since a display device is both an input and an output medium, there must be coordination between the user's input and the system's output. How the data is to be processed, and what responses will be generated, affect the options the user has available, and what must be done by the system to properly manage the output. If the processing function does not require a specific screen management, the user may request a mode which best fits the operations being performed. The following display modes are available to be used: line, edit (roll down input top), and frame. Forms mode is used to control user input and differentiate between protected and unprotected data. A terminal operating in forms mode uses control codes to separate a screen into fields of protected and unprotected data. Protected data cannot be altered by the user while unprotected fields are available for entry or update terminals which only send and receive data in the unprotected fields.

The Stock Point system supports the forms mode of operation with separate frame and data management. A frame is transmitted to the terminal to provide a format for input and/or output. The data associated with the frame is then treated as independent messages. Only the identity of the frame is appended. Rather than wait for a response, the user may want to use the terminal for other transactions. When the

frames mode is being used, the correct frame must be present before the output data is transmitted to the terminal.

The separation of data from the constant information enables the system to reduce the amount of network traffic by managing the frames at the point where the terminals enter the system. The frames manager must keep track of the frame currently in use by a display device. The formatting of the screen changes are done at the origin, not the destination. The different characteristics of processing within the system necessitate different forms of output management. Terminals operating in an interactive mode require rapid response time with little or no need to retain an image of the data transmitted; terminal queuing for this mode of operation would be unnecessary because the terminal operator is waiting to receive the data.

Three levels of output queue management have been defined: backlog, restricted and retention queues. The first two queue types share the same file structure and are linked back and forth, depending upon the status of the terminal and the user control. The restricted queue is to be indexed by user-id, unique-id, and originating (return) process. Records in this queue may be accessed by inquiries through the operator interface or they may be output by relinking to the backlog management, also done t hrough the operator interface. A message linkage procedure will be required to gather parts from the same creator and to insure that all parts are present in the

right sequence. The retention queue, by contrast, is to be built sequentially and retained until system maintenance.

Batch output, destined for a terminal, will reside on disk until requested by the terminal operator or scheduled by the system. The format of the data will depend upon the origin. Batch output received over the network will be stored as compacted file segments. The presence and type of data will be noted in the file directory. Batch output destined for a terminal is considered restricted data regardless of the mode of the terminal. Only one batch file may be output at a time. When a file is scheduled for output, the appropriate service routine will be called to access disk, format the data and pass the image for transmission. Batch input from the terminals may be managed by TAPS or the editor, software package depending upon the functions and destination desired. Validation and editing functions should be associated with key entry terminals.

Terminal Data Base Management is performed through TAPS/DM in support of disk files on the SPLICE processors. This is accomplished by means of the parameterized interfacing of the TAPS/AM with TAPS/DM. This permits user queries to the data base without the need for COBOL application programs, although COBOL module interfaces to TAPS/DM via the TAPS/AM are supported. As recognized in the SPLICE Software Design documentation of 19 March 1979, TAPS/DM only manages one open file at a time. Since the concurrent use of multiple files by application

programs is probable, TAPS must have the ability to back-out and/or restart a process if the entire process cannot be completed. The software design problems imposed by the single file support of TAPS can be overcome by using the Variable Indexed Sequential Access Method (VISAM) modules used by TAPS/DM for non-primary files. The restricting factor for using the VISAM methods, under TAPS/DM, is that the application programmer must know the exact data structure of the files and must perform specified procedures in a specified order. The result is a lack of program independence from actual file data structures. Once a data structure is modified, the corresponding corrections must be made to the respective application programs and then recompiled. Increased software maintenance and a decrease in program flexibility and responsiveness will result.

The final area of consideration under Terminal Application functions is data collection. The data collection function, known as data set management, is needed to support batch processing on either the background Burroughs systems or on the foreground Local Area Network. Data collection will include data element validation and batch naming considerations. The need for unique identification, UID, of multiple batch data generated from a process and received by TAPS at different times, must be provided. This provides the capability of having multiple batches of data which can be separately maintained, yet all classified under one generic file name.

Data Set Management, as used in SPLICE, concerns the management of batch files which are to be used either inside or outside of the local system. This is designed to be performed by the Data Management Transfer component which will use two basic control files within the Data Management Subsystem: a single File Directory and a single Environment File. The File Directory will be used to keep track of individual data sets, which are identified as file segments, while the Environment File provides predefined parameters for the processing of the logical files. The data sets are maintained in ancestor order within a logical file. Modification of a file is designed to be controlled through the Complex Manager which updates the File Directory and which may result in further processing of the file to reflect the desired modification. All control information concerning the output medium, or number of copies, are present at the logical file level and, if none exist, then the first controls encountered for the logical file are inserted in the File Directory.

Batch files used inside of the LAN concern both user batch applications and system batch functions such as the transmission of batch print and card files to and from the peripheral manager. The areas of concern for batch applications include compaction, decompaction and disk management of batch files and file segments. The extent to which compaction logic is performed on batch files is dependent upon the control parameters associated with the batch file and possibly upon its destination.

As is the case for some print messages and files destined for the Burroughs TC500/3500 terminals using the SPPROD program, horizontal compaction is required, while for other types of peripheral devices compaction is neither required nor desired. Decompaction of files would simply be performed by attainment of file structure control information at the destination and the subsequent restructuring of the data in accordance with the specified control information. Disk management of batch files should provide for the maintenance of transient files and the allocation of disk space while maintaining unique file names similar to that described under Data Set Management.

D. RECOMMENDED SPLICE FUNCTIONAL IMPLEMENTATION

This section represents the issues identified in the functional implementation of a Local Area Network by the authors, and not necessarily those of NAVSUP. It should be recognized that this section is general in nature and is not intended to include all considerations necessary to fully implement Terminal and Data Base Management functions on a LAN.

In attempting to identify the functional Data Base and Terminal Management requirements for the SPLICE LAN, the issues of a fully distributed versus a partially distributed, and the need for, or lack thereof, applications programs in support of Naval Supply requirements was first considered. NAVSUP depends highly upon the use of application programs, identified in Figure 1.1, to perform data entry and verification,

retrieval, updates and organizational and financial accounting functions, just to name a few. [Refs. 3, 4].

In considering the use of a fully versus partially distributed LAN, the initial functional design basically supports a fully distributed network based upon four primary functional LAN resources. The functional SPLICE LAN resources are identified as the Front End Manager, Terminal Manager, Data Base Manager and Peripheral Manager. A possible implementation of these functional LAN resources is depicted in Figure 1.3, but by no means represents their only implementation.

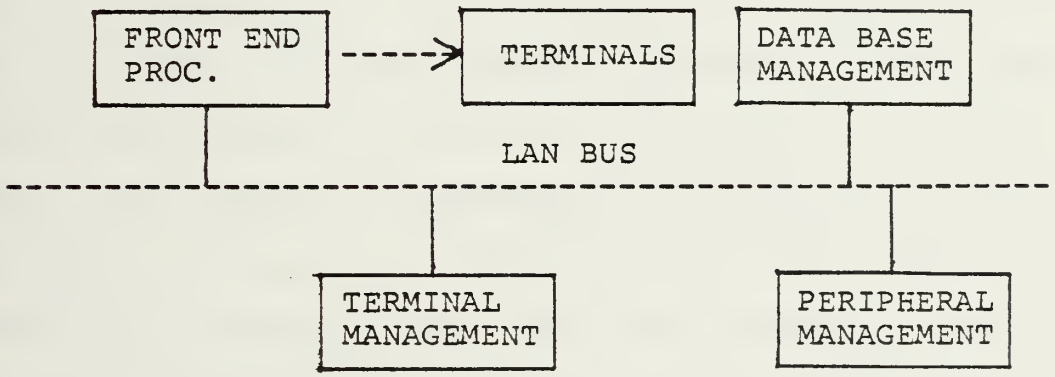


Figure 1.3 Possible Implementation of LAN Functions.

This decision was made in an attempt to provide the greatest degree of functional specification, with a layered approach, without becoming unmanageable. In this respect, the Data Base and Terminal Manager, as are the other LAN functional resources, considered to be implemented at various

nodes on the LAN. In so doing, the LAN should be able to achieve the greatest degree of parallel processing possible. Later designs may require the combining of multiple LAN resources at various nodes due to cost, user response and LAN load considerations. The Terminal Management and Data Base Management implementations are based upon the ISO OSI Reference model, in particular, layers 4 through 7. If desired, the combining of multiple LAN functional resources should be somewhat simplified.

The two primary problems encountered in the design of a fully distributed system were the need for maintaining the physical independence of each functional resource from the node upon which it is implemented and control of the execution of user requirements in a direct, or indirect, sequential manner. The physical independence of functional resources is considered in Chapter II, Section B.1, Physical Naming and Transparency Considerations, while the sequential control of user requirements is considered in Chapter II, Section B.2, the Session Services Subsystem of the LAN. If the physical independence of the functional resources is not maintained, future growth of the LAN and its operation in a partially failed state will become difficult, or impossible.

To say that the approach to the functional implementation of the LAN resources is "basically" fully distributed, is in reference to the Operating System which would support the layered architecture of the LAN functional resources. This thesis

does not consider the use of a fully distributed Operating System in support of the individual LAN functional resources primarily due to the scarcity of such products. Another consideration is that of overall LAN flexibility in providing for the expansion, or possible contraction, of the LAN as the organizational requirements change. By use of a generalized, vendor supplied, Operating System in support of each LAN functional resource, the hardware and its respective Operating System could be replaced without requiring extensive modification of the LAN functional resource itself. By providing well defined interfaces between the Transport Subsystem, Session Services Su bsystem and the Functional Resource Subsystem, only the interfaces need be modified when modifying or replacing the initial operating system and/or hardware. In this sense, each of the three primary subsystems of the DBMS and Terminal Manager could be treated as separate processes by the resident O.S., although the Transport Subsystem may be implemented as a separate hardware component. Interrupt handling and processor time allocation to both the Session Services and DEMS or Terminal Management Subsystems would be provided, and could be somewhat tailored to meet operational requirements, at each node supporting a LAN functional resource without a high degree of dependence upon the particular Operating System used.

The layered structure of each LAN functional resource is depicted in Figure 1.4. The Operating System, Transport

Subsystem and Session Services Subsystem are fully duplicated images of each other. The performance of one system/subsystem at one node/LAN resource is a mirror image of the same subsystem operating at another node/LAN resource under the same environmental conditions.

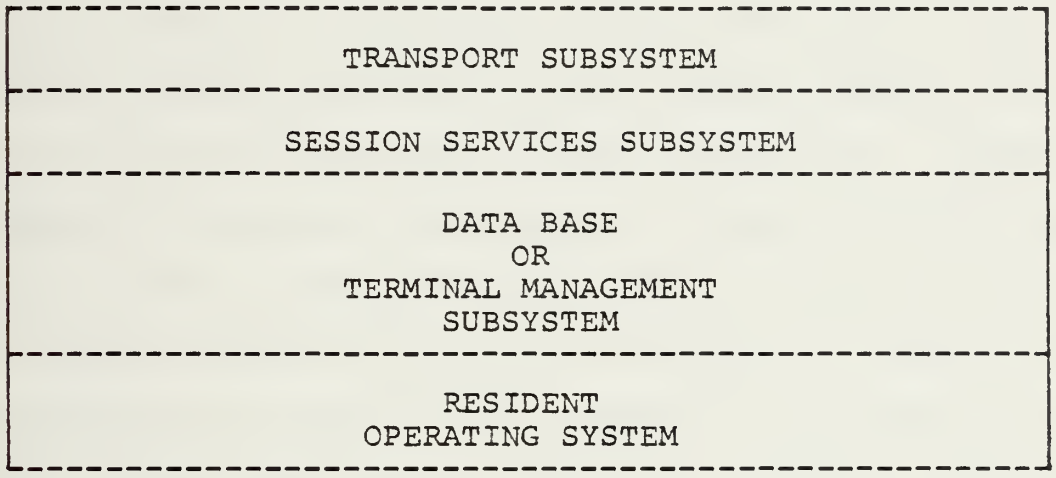


Figure 1.4 Functional LAN Resource Layered Architecture.

In addition to the flexibility in growth, use of a generalized Operating System also provides easy transition of supporting functional resources in the event of a failed node. This thesis promotes the use of duplicated functional resources to provide for graceful degradation of the LAN. "If a system depends upon all nodes being up, it is not a reliable system. If proper back-ups and failsoft levels can be defined so that the system is meaningfully operational with inoperative nodes, a reliable system may be designed from collections of smaller nodes" [Ref. 22]. Although this deals primarily with the

replication of hardware and software functions, there is a great deal of difficulty in coordinating the use of replicated functional resources to service each particular user while ensuring the integrity of a duplicated data base. In order to simplify these issues, this thesis proposes the acknowledgement of a primary DB and Terminal Manager, which could service the DB and Terminal requirements of all users, and a replicated, but passive, secondary DB and Terminal Manager to be used in the event the primary becomes inoperative. Regardless of whether a particular distributed or generalized Operating System is used to support the functional resources of the LAN, the primary objective of the LAN is to provide a single logical and comprehensive operating environment to the users of the LAN.

For primary and secondary functional resources, while the primary functional resource is active, the secondary, or duplicated, functional resource would exist in a passive and possibly inactive manner, at a separate node on the LAN. In the event of a hard failure on the primary node, the secondary functional resource would be made active, thus permitting continued support of organizational requirements in a partially degraded mode. The use of a generalized operating system, (GOS), at each node provides for simplified implementation and effective transition of a functional LAN resource from one node to another. It should be noted that, "Increased reliability comes not from the replication per se of hardware units but from systems designs that provide quick recovery while

guaranteeing integrity" [Ref. 22]. The proposed method of achieving this effective recovery state is discussed in the section on Back-up and Recovery, and also addresses the need for maintaining the concurrency of the primary and secondary data bases.

In terms of interactive application programs, their use is one which reflects more an organizational commitment, rather than a technical and managerial constraint. It is easy enough to conceive of eliminating application programs in lieu of a flexible and naturally structured DB Query language, which could perform data retrievals, inserts, deletions and updates while utilizing a DBMS to provide graphics and computational capabilities for the users of the SPLICE LAN. If designed and implemented correctly, it is recognized that the repetitive and sequential nature of application programs provides a sense of organizational security in that the user is restricted to the design constraints of the application program being used. It is this restrictive quality of application programs which often accounts for required program modification and maintenance in order to institute unforeseen user requirements or future desires. The feasibility and flexibility of using a high level query language must be demonstrated to the using organization in order to provide an incentive for foregoing the further development of the proposed SPLICE application programs.

In recognizing that considerable time, effort and capital investments have already been made towards the development of interactive SPLICE application programs, this thesis proposes a methodology for their support within a functionally distributed LAN. In order to support interactive application programs on a functionally distributed LAN, it would be necessary to utilize a Host Language Interface, supportive of an ANSI 74 Cobol standard, to translate application references to LAN functional resources. The ability to recognize references to LAN functional resources must be incorporated into the resident Operating System at each node. This support would be provided via the interaction of the Session Services Subsystem and the resident Operating System at each node on the Local Area Network.

Since the Session Services Subsystem provides the overall coordinating mechanism in support of user requirements on the LAN, once a user has been authorized access, application programs could be retrieved from the applications library and assigned to any one of the LAN functional nodes via the creation of an appropriate user Controlling Session Service. By identifying the user and the corresponding application program, the Session Service Subsystem, via the user's Controlling Session Service, would provide access to LAN resources by passing messages between the application program and the Session Services Subsystem. It should be noted that the application, i.e., the user's application process, and the Controlling

Session Service would reside at the same node within the LAN, and would be supported by that node's Session Services Subsystem and Operating System. Program interrupts would be handled by the resident operating system which would coordinate the message exchanges between the application process and the Session Service Subsystem, while providing Operating System level diagnostics in the event of a failure of the application process. Operating system diagnostic messages would first be linked to the user's Controlling Session Service by the Session Services Subsystem and relayed to the user via the Terminal Manager for future user action.

In the absence of application programs, support of repetitive data processing activities, in support of user requirements, can be achieved by use of previously developed command files which could be stored and recalled, much like an application program, from a user Command Library. Support of the command files would be performed in the same manner by the Session Services Subsystem as if the user had issued the series of commands from their terminal. The concept of user command files is described by J. D. Mooney [Ref. 17], and provides a method of implementing previously defined, repetitive user actions, thus freeing the user from the requirement of defining each desired action while permitting user interaction at predefined points in the repetitive sequence. Through the implementation of these concepts, it is possible to provide initial support of application programs and additional user

query language capabilities while permitting a gradual transition to purely query based user processes.

E. SUMMARY

The scope of this thesis is oriented around the functional design of the Terminal and Data Base Management components of a Local Area Network (LAN), in support of the SPLICE project at the Naval Postgraduate School, Monterey, California 93940. The proposed NAVSUP implementation of SPLICE reflects a tightly coupled architecture which utilizes a centralized "complex manager" concept. The complex manager performs all required coordination between the SPLICE system components, such as Terminal Management, Data Set Management, Peripheral Management, Terminal Oriented processing and Batch processing components. The recommended SPLICE functional design is directed at eliminating the need for a complex manager by providing the necessary control structures, transport and Session Services Subsystems, for a fully distributed Local Area Network. The recommended SPLICE implementation is based upon the basic design requirements as cited in the SPLICE documents provided by NAVSUP and FMSO, and is not intended to resolve all of the considerations necessary for full implementation.

The greatest distinction between NAVSUP's proposed implementation and that of the authors is the elimination of a centralized complex manager. The elimination of the complex management component of the LAN is supported by the use of

standardized transport and session services subsystems. These subsystems, along with an appropriate protocol, provides for the coordination of LAN functional resource components in support of user queries. The session services and the transport subsystems are covered in Chapter II of this thesis and are designed to support one or more functional resources located at a particular node on the LAN. There are two types of data used on the LAN, control data and user data. Coordination of LAN functional resources is provided by the exchange of control information between a user's session service and the Session Services Subsystem of the appropriate LAN functional resources. Recognition of operational functional resources and the location, address, on the LAN is made by use of a fully replicated resource directory at each node.

• The tools of a generalized DBMS and of a Terminal Manager are covered in Chapter III. Although the basic tools used by the two functional resources are identified and their relationships explained, the primary thrust of Chapter III is concerned with maintaining the availability of functional resources to users of the LAN. To this end, use of fully replicated, but passive, functional resource components are made in order to insure the availability of user services. By use of the control capabilities recognized in the session services subsystem, the failure of any functional resource can be identified and the respective replicated functional resource made active without the need for computer operator intervention. Through the proper

distribution of functional resources across the nodes of the LAN, the failure of any one node would be transparent to the terminal user and also provides a higher degree of overall LAN reliability than does the use of a centralized complex manager.

II. SESSION SERVICES AND TRANSPORT SUBSYSTEM

A. TRANSPORT SUBSYSTEM

The Transport Subsystem represents the fourth layer of the ISO OSI Reference Model. The primary function of the Transport Subsystem is to provide reliable end-to-end communication between the various nodes on the Local Area Network. Since this area of research is designed to be covered in depth by another thesis in support of the SPLICE project at the Naval Postgraduate School, only its requirements in support of the Session Services Subsystem, DBMS and Terminal Management functional resources will be discussed. At this point it will suffice to say that the Transport Subsystem must provide a "transport service" to the Session Services Subsystem. [Ref. 15].

The Transport Subsystem must provide the Session Services Subsystem with complete messages, while hiding all problems encountered with the construction of messages from packets and frames or vice versa. In addition to the above, the need for a high priority message transmission and reception capability is recognized for the communication of LAN resource control messages between the LAN functional resources. Priority control communication between LAN functional resources could consist of the notification of other functional resources that the sending resource is saturated and that further requests

should be delayed, test messages for the identification of non-functioning functional resources, network attention messages used during automatic LAN reconfiguration, and many more. The submission of messages from the Session Services Subsystem to the Transport Subsystem could be performed directly by use of a predefined queue, or by reference to memory locations by indicating a beginning location and message length.

B. SESSION SERVICES

1. Naming and Transparency

The identification of resources has become a central issue in the development of distributed systems in order to provide location independence and the possibility of having multiple copies of the same functionally named resource within the LAN. The transparency of location is the ability of a process, at various levels within a hierarchy of processes, to access data without knowing, explicitly, where it is stored. A process is an active entity, which alone can change the state of the system. A process can change itself, another process, or data. Since the fact that a process changing itself is inherently location independent, only the situation where a process changes another process, or data, is of concern in order to achieve overall location independence of a LAN functional resource. At best, complete location independence and transparency should permit end users, and application

programs, the ability to access and manipulate data regardless of whether it resides locally or at another node on the Local Area Network or at any remote node in the SPLICE system. [Ref. 11].

Location transparency can be implemented by use of a global directory and dictionary, which can be either centralized or decentralized, by use of a set of protocols for communicating between a local data directory and dictionary system or by use of unique resource addresses. "The directory can be centralized at a specific node, replicated at each node or distributed among the nodes" [Ref. 12]. Although the use of a directory and dictionary generically pertains to the accessing of data at local or remote sites, the same concept can be applied to resource identification within a message structured system. Using the concept of a directory or table, which could be either centrally located or distributed, a simple mapping algorithm can be implemented between generic message classes which pertain to one or more functional network resources. This requires a message protocol which provides for the identification of generic categories of network functional resources and particular client requests which the functional resources, the server, can then identify and act upon.

The concept of resource access via unique addresses is one developed by K. Lunn and K. H. Bennett, Department of Computer Science, University of Keele, England [Ref. 13]. Since each node on a network can be identified by a unique

address, the address of a resource consists of the address of the node at which the resource currently resides. In order for a process, the client, to access a resource, the server, it is necessary for it to obtain the server's address. This was done by using both local and global directories, called the Local Available Resource Directory (LARD), located at each node, and a single Total Available Resource Directory (TARD) for the network. Although the authors indicate that this concept was developed specifically for a ring network structure, it can be simulated under other network architectures, such as Ethernet, which uses a bus structure.

The LARD maintains no other information external to itself except for the address of the TARD. When a process sends a find request to its LARD, which contains the resource name, it waits for the LARD to reply with the resource UID (address), or an indication that the resource does not exist locally. The LARD initially checks to determine if the resource is local to the node; if it is not, it sends a request to the TARD which replies to the process with the address of the resource or an indication of it not existing within the network.

Initiation and termination of resource availability at a node is accounted for by making an appropriate entry in the LARD and the TARD. In the ring structure, when a node or resource comes on-line, or active, and does not know the location of the TARD, a message is sent to its neighbor, who passes

it to the next, until the TARD is located and replies with its address to the requesting node or resource. Should the message requesting the location of the TARD circle back to the initiating node, it is assumed that the TARD is disabled or never existed, as in the case of initial network start-up. In this case, a bid message is circulated among the nodes of the network. Each node examines the bid and either passes it on unchanged or increases the bid value and indicates its address. When the bid returns to its originator, it examines the bid and the address of the node with the highest bid, and sends a message to the highest bidder requesting it to create a TARD. At this point, the node initiating the TARD requests information from each LARD and completes its creation process.

Duplicative bidding, node crashes and start-ups can cause problems in the creation of a TARD. When duplicate bids exist as a result of two or more LARDS detecting the absence of the TARD, a LARD, upon receipt of a bid, enters a "bidding mode," which prevents it from changing its bid, thus resulting in at least one bid being higher than all others. Even in the event that two TARDs were to be created simultaneously, the TARD requesting information from the other LARDs would indicate its bid value, and the node with the lower bid value would back down, thus permitting the TARD with the highest bid to remain active. This can happen when a node starts up and misses the first bid, but bids highest on a second circulating bid. Node crashes are taken care of by use of time-outs, where

the originator of a message would prevent indefinite hold-up when the node servicing its message crashes.

In the above discussion on resource identification and naming, there exist problems of directory storage and reference, either in a decentralized or centralized manner, and in maintaining the identity of failed or duplicate resources. The solution to these problems may reside in simply combining the capabilities of the Local and Total Available Resource Directory, and by use of generic functional resource identifications. The generic functional resource UID could be maintained in a table and directory format, Figure 2.1, and whose address could be initially generated from the identification of its physical location, logical location or a combination of both through an address computation algorithm.

Message Type	Status Act/Pas	Resource Unique ID	Resource Control Characteristics
Term.Mngt.	Act Pas	1111 2222	**********
DB Mngt	Act Pas	3333 1111	**********
Per Mngt	Act Pas	2222 3333	**********

Figure 2.1 Resource Directory Table.

The main issue of naming and transparency for the SPLICE architecture and functional implementation is that there should be no difference between local and remote messages. Each message, originating on, or received by a LAN functional resource, and being processed through the Session Services Subsystem, would have its location recognized and be directed to the appropriate local functional resource by use of the Resource Directory.

The resident Session Services Subsystem would have knowledge of its supported functional resource(s). If the message in question is mapped to the resident LAN functional resource, then the message is vectored to the appropriate LAN node. If the message does not map to a resident functional resource, the Session Services Subsystem would reference the Resource Directory to determine the UID of the referenced functional resource. Once the UID of the referenced functional resource is determined, the Session Services Subsystem would construct the appropriate message and pass it to the Transport Subsystem for transmission over the LAN bus. Rather than using a decentralized resource directory, the Resource directory at each node on the LAN should contain an entry for each functional resource, its default protocol characteristics, and the UID of both the active and passive copy of the functional resource within the LAN. Since the total number of LAN functional resources is perceived to be small, due to the vertical partitioning into four primary functions, a centralized

implementation would be extremely efficient while providing a central reference point for reassignment of functional resources to alternate nodes if necessary.

2. Session Services Subsystem

The Session Services Subsystem represents layer 5 of the ISO OSI Reference Model and is utilized in establishing connections between processes in a hierarchical manner, where a user session would refer to the Session Services Subsystem to utilize a network functional resource, such as the Peripheral Management resource of the LAN. In addition to establishing connections between processes, the Session Services Subsystem is concerned with the primary issues of session binding, maintaining multiple outstanding requests on a particular user's session, i.e., process-process or process-multiprocess sessions, and the maintenance of the atomic properties of session transaction. It is recognized that there is no standard Session Layer protocol which is mainly attributed to the primary concern over the lower layer protocols of layer 1 to layer 4. The higher layer standards will hopefully be forthcoming. [Ref. 15].

Session binding is generally concerned with establishing certain conventions about a session between two or more processes, or between a user's process, represented as a session, and LAN functional processes. It typically addresses such issues as full versus half duplex, character code, the presence or absence of encryption or text compression, flow

control window size for use by the Transport Subsystem, and the methods used to recover from transport layer failures. The issues of half versus full duplex, and the character code are not at issue in the original design of the LAN in respect to the Data Base, Terminal and Peripheral management modules. The communication between the LAN functional modules is based upon homogeneous network components, but future growth may dictate the need for character conversion among future nodes/ resources due to expanding organizational needs.

Within the SPLICE LAN, the Session Services Subsystem would be duplicated at each node supporting a functional resource. The Session Services Subsystem would provide two types of control for users of the LAN, general user session service for communication between LAN functional resources, and a Controlling Session Service which would be used to control the overall need of each terminal user. A session service would be created at a node whenever a reference to the functional resource at that node is made by another functional node on the LAN supporting a terminal user. The user's session service would be formed to provide any unique control information in support of a user's request for service between two or more nodes or functional resources. This is particularly necessary if one considers a user's request for a logical file from the Data Base Manager which would be compacted or encrypted. In order for the requesting Session Services Subsystem to properly interpret the logical file, it must first

have knowledge of its structure. The establishment of a session service, in respect to the request for service by a user, provides a method of establishing control and format information prior to the receipt of the requested data.

A Controlling Session Service is a reference block established for each terminal user of the LAN and is used to provide a form of centralized control of a user's session with the LAN. There is only one Controlling Session Service per user, and it may reside at any functional node on the LAN within that node's Session Services Subsystem. Its primary purpose is to provide a reference control point for each user's session and, at the very least, retains all outstanding requests for service for that particular user.

A Controlling Session Service is established after a terminal user has been cleared for access to the LAN by the Terminal Manager. Since each terminal user must gain access to the LAN via the Front End Manager, the Front End Manager would initially direct a user's terminal messages to the Terminal Manager. In that the Front End Manager serves as the entry-point for users of the LAN, a decision must be made as to whether the Controlling Session Services should reside within a single node, or be capable of residing at any node on the LAN. In either case, the Front End Manager must have its destination reference for each user amended to reflect the location of the user's Controlling Session Service, either explicitly from the Terminal Manager or implicitly by use of

a default destination for a single node implementation. The simplicity of having a particular node or module provide user Controlling Session Services is possibly its most attractive feature although the need for additional resource control messages is also eliminated. One of the drawbacks of a single node or module implementation is that user support is dependent upon the availability of that node or module. By permitting user Controlling Session Services to reside at any node on the LAN, the non-availability of any one node would only result in partial degradation of the LAN. It is recognized that a dual processor node, as with the use of a Tandem-like system, a single module implementation may be highly desirable. Restoration of LAN functional resources and any affected Controlling Session Services are covered in Chapter III.

If Controlling Session Services were capable of residing at any node on the LAN, then once the terminal user has been cleared for access to the LAN, the Terminal Manager could elect one of two possible options, depending upon its own load considerations. The Terminal Manager would have the option of either accepting responsibility for that terminal user by establishing a Controlling Session Service, or it could make a request for one of the other functional nodes to establish a controlling session service by issuing a control message to each LAN node. This would be done by issuing a control message to each functional node address supporting an active functional resource. The first functional node responding with a positive

acknowledgement would have its address indicated to the Front End Manager and all future interaction with the LAN would be directed, via the Front End Manager, to the corresponding functional node. It is the Controlling Session Service which actually maintains the sequential execution of a terminal user's commands and the atomic properties of the corresponding transactions within the LAN. It is recognized that LAN functional control messages should not require the establishment of a session service between functional nodes since the protocols for control messages would follow a standard predefined format. Only user data messages need have a session service established.

The LAN must be capable of supporting a variety of terminals, each with different screen sizes and control requirements. Since the initial user access to the LAN is via the Front End Manager, it is felt that the Front End Manager should provide for the identification of a user's generic terminal type in the initial interaction with the Terminal Manager. This identification of generic terminal type would be used by the Terminal Manager in constructing the appropriate control characters for the actual display user data onto the respective physical terminal. In addition, the Front End Manager should provide for the conversion of binary character codes into a standard binary character code representation, such as ASCII, used on the LAN.

The presence or absence of text compression and encryption are characteristics provided by the DBMS subsystem which must be accounted for when communicating between LAN resources. This thesis presents a viewpoint of a hierarchy of processes within each LAN functional resource which are organized from lowest to highest by Operating System processes, LAN functional resource processes and user processes. As used here, a user's process is synonymous with the concept of a user's session within the LAN. Although the need for both batch and interactive user processes has not been proven relative to the use of a well structured and flexible database query language, provisions for its use must be provided in fulfillment of the SPLICE functional design requirements.

The Session Services Subsystem provides the overall controlling mechanism among the clients of the LAN functional resources, i.e., the terminal user, and the LAN functional resources themselves. Regardless of whether a process is based upon an interactive application or upon an interactive session, via the issuance of query language transactions by a user of the LAN, there must be a Controlling Session Service to communicate and control the requirements of the user process(es) between itself and the capabilities, or functional resources, of the LAN. In order to establish communication between the Controlling Session Service, the client, and the Session Services Subsystem of the LAN functional resources, the servers, a simple request-accept message transfer needs

to be performed. The Controlling Session Service would request service from the non-resident Session Services Subsystem of the appropriate functional resource, when an initial reference is made to it by a user session. The request for service, via messages, would include the requesting Session Service ID, the requested Session Service ID, user process ID, and any appropriate control information, such as the use of text compression or encryption. Figure 2.2 represents the general format of a request for the establishment of either a controlling or non-controlling session service.

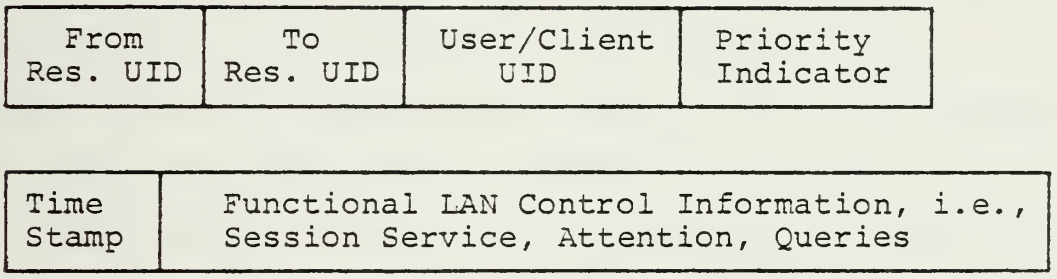


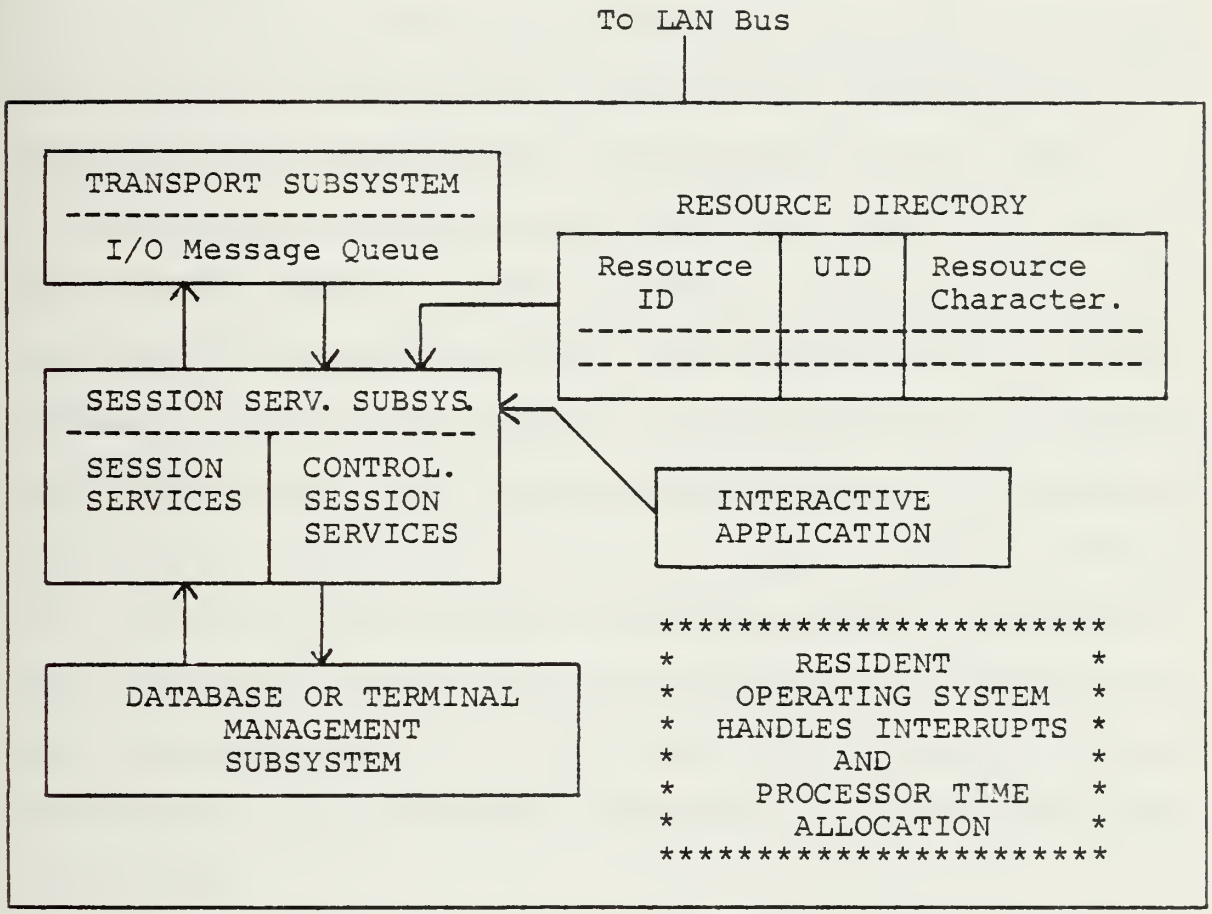
Figure 2.2 General LAN Control Message Format.

The receiving Session Service Subsystem would examine the client's request and establish the necessary address space, and control information for future communication, if it is capable of handling future client requests. An acknowledgement, indicating either acceptance or denial of session support, would then be sent to the requesting Session Services Subsystem and would include any control characteristics which may be necessary for further communication. Once this sequence of

events has been accomplished, the user's process, through the Controlling Session Service, can communicate freely without the need for reconstruction of future session services. Reference to currently supported user sessions would be maintained in an Active Session Table, which would be used in identifying the validity of client access for service by the functional resource.

There are two possible situations which may exist for the interaction of LAN functional resources and user sessions. The term "user session" is used here to refer to either an interactive user process with the LAN via a high level query language or via an interactive application. When a user's session includes an interactive application, both the application and Controlling Session Service may reside within the same LAN processor. The first situation exists when the user's session and the LAN functional resource reside within, or are supported by, the same LAN processor, and is represented by Figure 2.3. In this situation, the requirements for text compression and encryption should not exist and the Session Services Subsystem need only coordinate the resident user's session DBMS or Terminal needs to the resident LAN DBMS or Terminal Management functional resource. The user's session communicates with the resident functional resources via the Session Services Subsystem which insures that the atomic properties of a user's process transactions are maintained. If the user's session includes coordination with an interactive program,

upon fulfillment of the user's session transaction the Session Services S ubsystem communicates the results back to the user process, otherwise the results are communicated back to the terminal user via the Terminal Manager.



Note: Each Session Services Subsystem will contain both Controlling and Non-controlling Session Services.

Figure 2.3 Functional Resource Processor Configuration.

The second situation concerns the communication between a non-resident user process and the DBMS or Terminal functional resource. When a user's process makes a request to a functional resource, the Controlling Session Service utilizes the Resource Directory and matches the transaction resource ID to the UID of the functional resource on the LAN. The Controlling Session Services would be responsible for monitoring the establishment and destruction of a user's sessions among the various LAN functional resources. Only initial references to a LAN functional resource should require the creation of non-controlling user sessions and all future references should only require the identification of the function resource UID and subsequent construction of the appropriate messages required to fulfill the user's session transactions. Once the functional resource UID is determined, the Session Services Subsystem constructs the message in accordance with the characteristics of the LAN functional resource and passes the message to the Transport Subsystem for transmission over the LAN bus.

Upon receipt of a message from the Transport Subsystem, the Session Services Subsystem matches the message user UID to its active session table. If there is a matching entry, then access is permitted and the LAN functional resource is invoked via message. If a match does not exist, the Session Services Subsystem would simply disregard the message, since no subsequent messages should be transmitted unless an acknowledgement

for the creation of a user's session service has been received by the requesting or Controlling Session Service. This provides a type of security, such that information cannot be extracted from a functional node unless a properly established user's session service has first been defined.

It should be recognized that certain requests for service from a user process may require coordination and control of multiple LAN functional resources. In this case the Controlling Session Service Subsystem would be responsible for ensuring that a user process request for service is appropriately broken down into its respective component requests for service and that they are performed in the appropriate order. This often results in a chaining effect, creating a series of client-server relationships. An example may be the request of a user's process for the selection of two data elements, their summation and subsequent printing of the results to a hard copy printer. For the sake of argument, let us assume that the user's process was generated by a user logging onto the LAN via a terminal and that the user's Controlling Session Service was established within the Terminal Manager. The Terminal Manager's Session Services Subsystem could provide either centralized or decentralized control of the user's request for service. For centralized control, the Terminal Manager's Session Services Subsystem would break down the user process request into two separate messages, one for the DBMS and one for the Peripheral Manager, and would insure receipt

of the resulting summation of data items from the DBMS prior to requesting the printing of the results by the Peripheral Manager.

In decentralized control, the Terminal Managers Session Services Subsystem would request the appropriate services of the DBMS manager and indicate the destination of the results, i.e., the Peripheral Manager. Figure 2.4 represents a general message for decentralized control of a user's request for service.

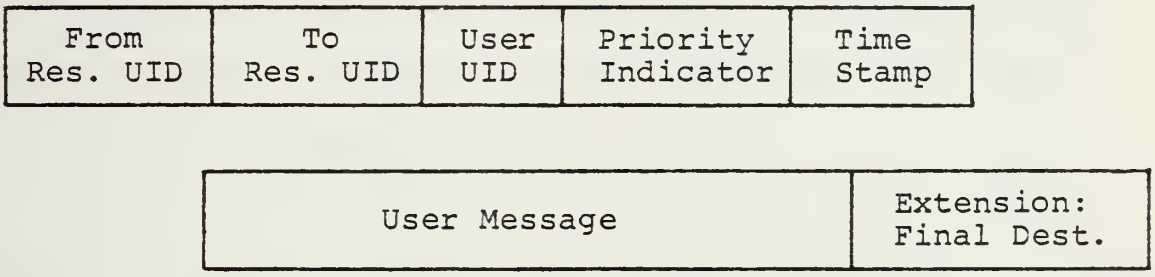


Figure 2.4 Decentralized Message Format.

Since each Session Service Subsystem has access to the control characteristics of each of the LAN functional resources, the decentralized approach would provide the benefits of decreasing the required number of messages to complete the user process transaction. The centralized approach would require the transmission of two messages, assuming that an appropriate user session service has already been established among the appropriate LAN functional resources.

III. DATABASE AND TERMINAL MANAGEMENT FUNCTIONS

A. OVERVIEW

The concepts employed in the recommended implementation of the Data Base and Terminal Management resource requirements for SPLICE center around a highly decentralized and loosely coupled distributed Local Area Network (LAN). It is perceived that the Data Base Management, Terminal Management and Peripheral Management resources of the LAN will be implemented on separate complete processors, which communicate with each other via a bus structure.

Section B will describe the recommended use of Database Management software concepts for the DBMS requirements of the SPLICE LAN. This information has been collected from numerous SPLICE documents provided by NAVSUP and FMSO. [Refs. 1, 2, 3, 4]. It will illustrate and describe the major elements of the Data Base Management resources of the LAN and is not intended to encompass all the considerations necessary to fully implement the proposed design concept. Section C will discuss the concept of Virtual User Terminals as it applies to the Terminal Management Functions of SPLICE. Section D will discuss the issues of Back-up and Recovery as they apply to the Terminal and Data Base Functional Resources. A general model for Back-up and Recovery is provided along with its application to three

possible implementation architectures for dual or primary and secondary LAN Functional Resource Managers.

B. RECOMMENDED USE OF DBMS SOFTWARE

1. Introduction

The recommended functional implementation of Local Area Network (LAN) functions is based upon the horizontal distribution of four major functional resources: Terminal Management, Data Base Management, Peripheral Management, and Front End Process Management. Each of these functional resources operate as peers, i.e., a peer-coupled system, communicating with each other over a bus structure with message passing mechanisms. The main advantages of the peer-coupled system is that it provides the greatest flexibility in overall system design, while providing increased reliability of the LAN overall, in the event of the failure of any node. In addition, a loosely coupled, or peer-coupled, LAN helps the elimination of the bottleneck which is often experienced in a vertically distributed system where a centralized control mechanism is used for resource assignment. [Ref. 6].

This thesis recognizes the following major components and files of the Data Base Management module of the LAN: Data Base Management System, Message Communication and Control Subsystem, User Transaction Log Files, and Alternate Function Log Files. These major components and files are used to support and control user references to the data base while

providing user reference and system state information for back-up and recovery of user processes and the LAN DBMS functional resource.

2. DBMS Subsystem

The criteria for selecting a particular set of data management capabilities is not the concern of this thesis and would most likely be based upon such items as response time, security, ad hoc user query requirements, application independence from physical data structures, data accessibility, sharability and integrity, just to name a few. Although a recommendation for any particular type of DBMS is not made, the benefits of utilizing a DBMS will be discussed along with its interactions with the Session Services Subsystem and the Transport Subsystem as defined by the International Standards Organization (ISO).

The term DBMS, as used in this thesis, will refer to a fully implemented data management facility which is capable of supporting predefined data structures for use by interactive and batch programs, and on demand user queries to the LAN data base. The following is a list of six data management tools considered necessary to provide overall organizational support for highly interactive high level user queries and application programs.

1. Data Dictionaries/Directories (DD/D)
2. Data Definition Languages (DDLs)
3. Data Manipulation Language (DML)
4. Database Query Language (DQL)
5. Database Utilities
6. Data Communication (DC) Facilities.

It is recognized that not all DBMs will incorporate all of these components in separate modules and that future technology will probably lend itself to a hardware/software combination in providing the capabilities as cited by the data management tools. Depending upon the particular DBMS under consideration, various combinations of the above DM tools may be combined, but their general capabilities, and the degree to which they are implemented should be considered in respect to organizational requirements.

The Data Dictionaries/Directories (DD/D) are implemented in a variety of ways and may even be integrated within the DDL and the DML. The Data Dictionary is used to identify and define the data elements contained in the data base, and any relationships which may exist between these data elements. The data directory generally describes how each data element is used and by whom it is used. The data directory may refer to application programs, input or report documents, or simply to job streams, but in general it supports the use of data elements identified by the data dictionary.

A Data Sub-Language (DSL) consists of a Data Definition Language (DDL) for defining data objects (e.g., fields) and a Data Manipulation Language for the processing of data objects. One use of the Data Definition Language (DDL) is for the defining and describing the organization of data by the Data Base Administrator (DBA) or his staff. As identified by the CODASYL group, its main function is to describe the

content and structure of the schema and subschema. Although all DBMSs have a DDL, they vary in the manner in which data elements are described in the data model, hierarchal, network, or relational, and the extent to which complex relationships can be specified by the DBA. The complexity and capabilities of the DDL should be of prime consideration in its use at the Navy Stock Points and Inventory Control Points.

Data Manipulation Languages (DMLs) are used to transfer data to and from the database. It can be accessed by calls from a procedural language. The use of a calling mechanism can be implemented in a message-based communication subsystem which is used to provide communication between processes. It should be noted that the capabilities of the DML directly affect the applications programmer. Since the DDL and DML are closely related, the functionality of each generally determines the degree of responsibility which must be assumed by the DBA or application programmer. As the functionality of the DDL increases, the DBA's responsibilities will likewise increase, thus decreasing the programmer's responsibilities for the use of the DML. Since the data base which will reside on the LAN is limited to twelve transaction types, an increase in the DBA's responsibility should provide for greater programmer productivity and program maintainability over the life cycle of the SPLICE project.

Database Query Languages (DQLs) are generally interactive in nature, although batch DQLs are also available.

Often called "end user facilities," DQL facilities provide direct interaction with the data base schema and permit parametric and Boolean search strategies for data retrieval or updates by approved end users of the DBMS. By providing a user friendly DQL, users can either perform ad hoc queries or can build user command files for repetitive data entry, retrieval and validation. In spite of the twenty new application programs being developed for SPLICE (see Figure 1.1), the need for current and future information requirements by the Navy Supply System can be supported by a fully implemented and varied Database Query Language.

Database utilities cover a multitude of areas, from password security to image management software and database tuning utilities. Database back-up and recovery generally consists of four components: dump files, journal files, utilities and operating procedures. The dump files are used to copy the entire database, with its control structures, while the journal files are used to record transactions processed against the data base. In order to provide for back-up support in the event of a hardware or software failure at the DBMS processor, it is recommended that the database and its structure, along with a duplicate journal file, be maintained at an alternate node in order to provide real-time back-up support of DBMS functions. The alternate node assignment is discussed further in the section on back-up and recovery. The back-up and recovery utilities perform roll forward operations

which update dump files without requiring complete dump file recreation and roll back operations to reverse the direction of erroneous DBMS transactions or bad data. Additional data base software for text and graphic displays, audit trail utilities, data base tuning utilities, database development aids, data base reloading and reorganizing aids and database sizing and responsiveness aids are also available and could prove to be extremely useful in support of LAN life cycle requirements.

The final component of the DBMS concerns the Data Communication (DC) facilities which were originally introduced by IBM in the early 1970s as an integral part of the data management technology. Although DC facilities can be implemented to provide for distributing data bases and establishing network DBMS facilities, DC as used here, refers to the communication facilities used to interface actual DBMS functions with the Session Services Subsystem discussed in the following section [Ref. 9]. Figure 3.1 represents the relationships of the six component tools of a DBMS, as discussed above.

Although the above list of DBMS components may not cover every aspect of DBMS component functions, they are used to describe the types of facilities which must be considered in providing a completely distributed processing system in which the actual DBMS functions are implemented separately from the actual interfacing of the session services and transport subsystems. By restricting the DBMS functional interfaces to the Session Services Subsystem and the Operating System of

the particular vendor, greater flexibility in system design and future growth should be achieved by localizing the requirements for interface modification, should the need exist.

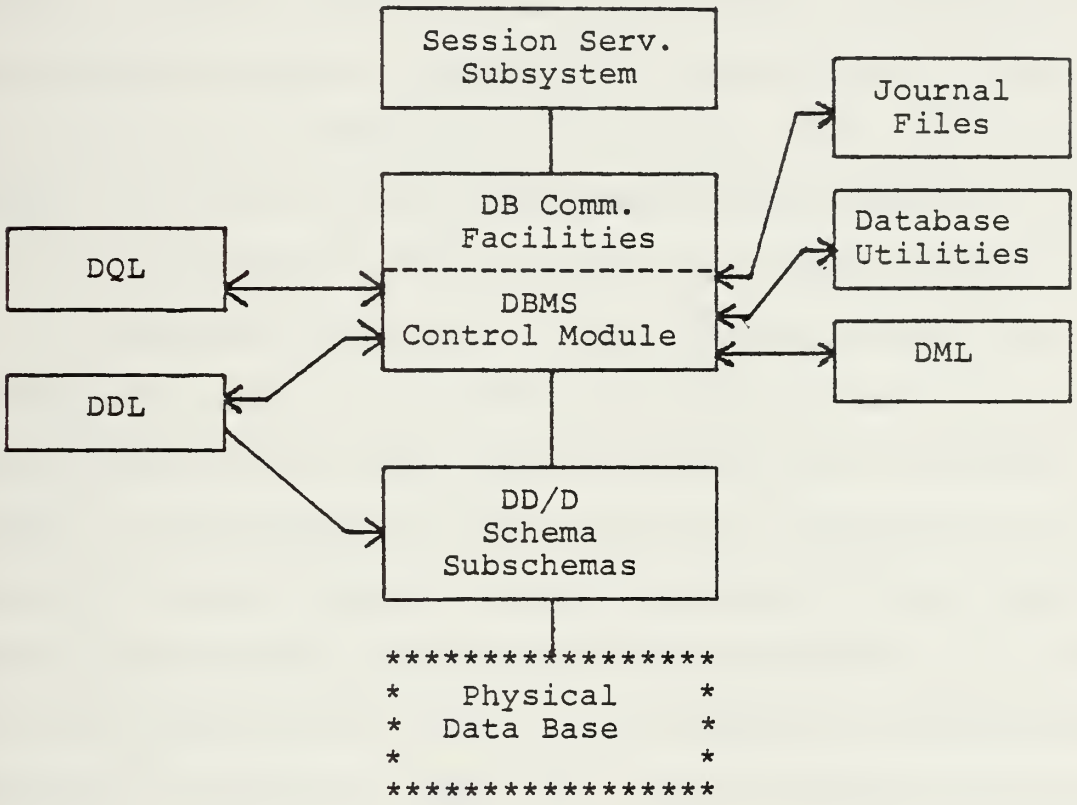


Figure 3.1 General DBMS Component Tools.

In addition to the standard benefits and restrictions imposed by use of a DBMS, the organizational needs of the Naval Supply System and its supporting organizations should be considered in developing a network and the corresponding DBMS functions. Access between Stock Points (SPs) is designed to be performed via the Inventory Control Points (ICPs).

With this concept in mind, and the SPLICE implementation on a LAN, the Stock Points may physically maintain a centralized database, yet the overall Naval Supply database, as accessed over the national network and supported by the Defense Data Network, could be conceived as a physically and logically distributed database. Although the consideration of the National Network implementation is beyond the scope of this thesis, to fully utilize the organizational aspects of a DBMS, use of a "federated database" should be considered [Ref. 10].

Federated databases, as introduced by McLeod and Heimbigner, consist of a number of logical components, i.e., Stock Points, which have their own logical and conceptual schema. Each component schema within the federation is related by use of federal schemas, Inventory Control Points, which are used to express the commonality of data throughout the federation and specifies what data can be accessed or modified by federation components. [Ref. 10]. Since database users or application programs generally access data within the local component, known as locality of reference, a great majority of locally generated transaction requirements could probably be accommodated within the SPLICE LAN. In the event that a federation component requests data maintained by another, or several federation components, the issuing component consults the federal schema to find the data either explicitly or implicitly, based upon the requesting transaction.

C. TERMINAL MANAGEMENT FUNCTIONAL IMPLEMENTATION

The purpose of Terminal Manager is to provide LAN users with the facilities for communicating simultaneously with a large number of processes spread out among various computer systems. A terminal user might need to communicate with the Local Area Network or other local area network (e.g., other Stock Points) through a national network.

This thesis recognizes that the terminal users should be able to manage any number of concurrent processes within that user's session with the LAN, to see multiple outputs on the display device as the user desires [Ref. 23]. To achieve this goal, we use a concept called Virtual Terminal Management, which converts a single physical terminal into multiple virtual terminals, each of which may be written into or queried for input. Virtual terminal management extends the features of the physical terminal by providing extensive editing facilities, the capacity to maintain all or selected output in disk-based data structures and sophisticated mechanisms for the management of the terminal screen.

Virtual terminals are device independent, where the specific characteristics of the terminal at hand are known only to the lowest level of the Terminal Manager. Given an environment in which a high degree of simultaneous activity is possible, one fundamental problem encountered is how to translate that activity into a form comprehensible to the user. It has been thought that terminal users should be able to simultaneously manage any

number of concurrent logical processes, or logical user sessions, with the LAN. This is based upon the belief that people in their daily work routinely handle multiple concurrent tasks, with many interrupts, while realizing some specific tasks and being able to switch back and forth quickly between related tasks. [Ref. 24].

The discipline for user command interaction should be consistent across all user sessions and would include the use of control keys, help facilities, prompting, feedback, etc. This concept works with the following rules:

1. The user must have complete preemptive control of his terminal at all times. He should be able to allocate and arrange the space on his display device at will, by selecting which process to view at any one time.

2. Processes should never depend on the actual mapping of their output onto the user's display device. User sessions may stipulate preferred viewing conditions, but these are applicable only as long as they do not interfere with the first rule.

3. The output of any user session should never be discarded by the Terminal Manager unless specifically requested by the user or by some pre-specified time interval on the system.

The users see a collection of virtual terminals mapped onto rectangular areas of their display device. Each virtual terminal may be thought of as roughly equivalent to an independent physical display device. Only one virtual terminal is termed active at any instant in time, representing the one used for acceptance of input from the terminal keyboard. Through the use of special keys and monitor commands, the

user should be able to switch his attention from one virtual terminal to another. The terminal user may choose to see as little or as much of a particular virtual terminal as he wishes. They may block or discard output, abort or suspend the session associated with a virtual terminal or they may specify at any time to scroll back and forth to review previous material.

It is important to realize that a user may have more than one virtual terminal displayed on his physical terminal at any one time, but user interaction with the LAN is always supported by reference to only one particular active virtual terminal. A virtual terminal has the following components: a line, a Pad and a window. The line is the virtual terminal's source for keyboard input. The Pad is a disk-based data structure used for storing and editing a virtual terminal's output. The window represents a potential mapping of a virtual terminal onto the display device and is managed by a screen handler. A logical session deals with a single virtual terminal and its data structure. Special interface routines distribute the necessary commands to the appropriate logical session represented by a virtual terminal.

The screen handler and a line handler are created when a user first accesses the SPLICE system. The screen handler is responsible for managing screen space of the user's terminal. The line controller is controlled by the terminal input handler, and is responsible for handling all subsequent input requirements for the user. One Pad Handler, associated with each

terminal user, is initiated for each user session with the LAN and satisfies that session's output requirements by sending display commands to the terminal output handler. In Figure 3.2 we can see a virtual terminal controller with its supporting components.

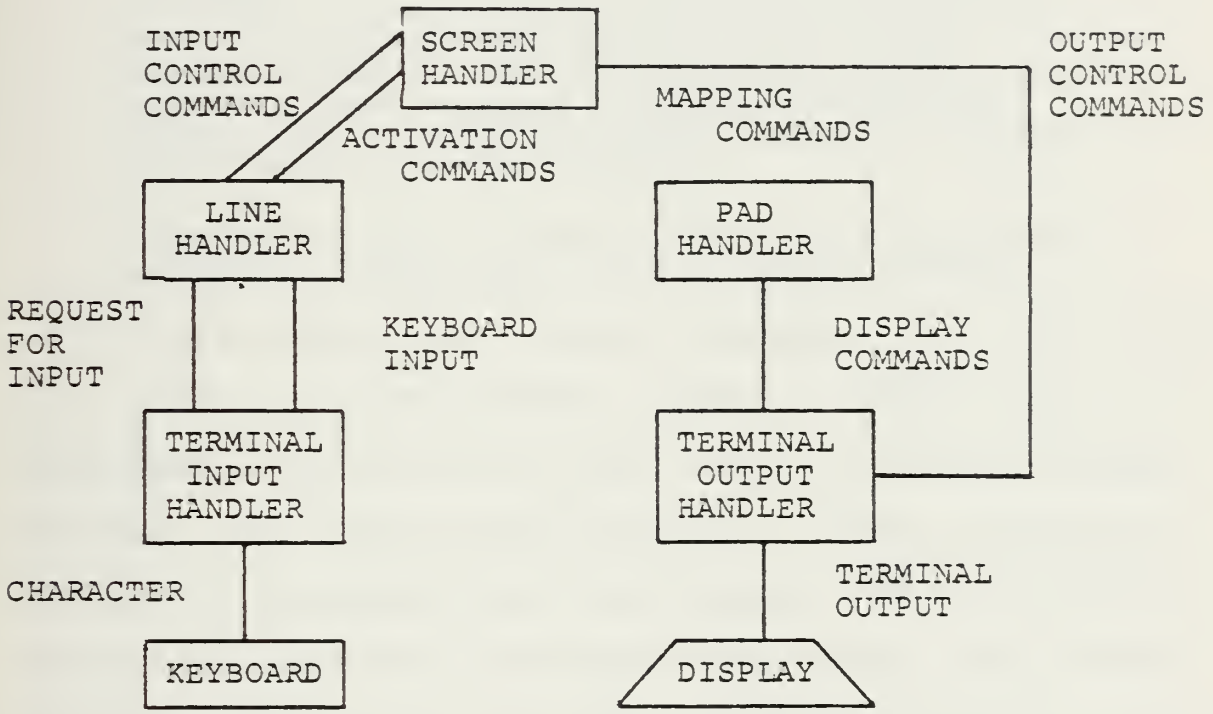


Figure 3.2 Virtual Terminal Controller.

User output can be multiplexed in both space and time by using the two-dimensional features of a display terminal. Input can only be multiplexed in time and this function is performed by processes called line handlers. Line handlers allow user access to support processes such as editors, compilers, etc., and are associated with a logical input device, called a line. The user session owning a virtual terminal may

request that input be collected, via their logical input device, in one of three modes:

1. Character-at-a-time: A single character is returned in response to each request. Echoing is optional.

2. Page-edit: Characters typed by the user are allowed to modify the contents of the virtual terminal until a user specified break character is typed. A user session may also specify the set of acceptable characters, such that any character not in that set would be ignored. The page-edit mode is used primarily for editing files, and is basically a driver for the editing facilities of the Pad-HANDLER.

3. Line-edit: This is used primarily for processing commands, entire text lines or single tokens.

When processing user terminal commands, it is often useful for a user to specify indirect sources of command input, i.e., programmable function keys, macro files, or command language programs. The output capabilities of a virtual terminal are provided by processes called Pad Handlers. The Pad Handlers manage disk-based data structures called Pads, each of which represents the storage and display characteristics of a virtual terminal. A Pad Handler maps a Pad onto specific areas of a user's terminal display by issuing commands to the terminal output handler, seen in Figure 3.2. A Pad provides the ability to hold a number of lines of text, up to some maximum number, based upon the terminals formatting and display capabilities. All changes made to the user's Pad reflect changes made to the user's terminal image on the display screen. A given Pad can be associated with more than one Virtual Terminal supported by

the Terminal Manager. This feature can be useful when a Pad contains format information which is applicable to multiple terminal users. A range of editing features are provided by the Pad and include some or all of the following:

- -Cursor motion by characters, words, lines and pages.
- -Deletion of characters, words, lines and pages.
- -Joining and splitting of lines.
- -Character over-write or insertion.
- -String location and substitution.
- -Text selection and transfer.

The Pad provides all basic text editing facilities for the terminal manager. The Editors of the Terminal Manager can be shared by multiple Pads and implements only the more complex editing commands such as definition and execution of edit macros. By placing all basic edit functions on the Pad-HANDLER, they are available to all Pads representing virtual terminals. The ability to modify the mapping of a virtual terminal to a user's screen, without affecting the Pad's output cursor, is central to the system's ability to display a virtual terminal's past activity. Normally, movement of a Pad's output cursor also changes the mapping of Pad lines onto a window, resulting in a scrolling action, i.e., the most recent data is always displayed. Instead, output mappings may be defined to follow a second type of cursor, the viewing cursor. There are as many viewing cursors as there are virtual terminals for

a given Pad, each under the control of the virtual terminal controller assigned to each user's virtual terminals. The user may detach the viewing cursor of a virtual terminal for the purpose of reviewing past text and perhaps selecting it as input to another logical session.

Screen handlers solve problems associated with a user's desire to see multiple logical session results on their terminal screen. This is accomplished through a hierarchical decomposition of the screen space reminiscent of the way computer graphic systems divide and map data onto graphic output devices. The physical entities dealt with are: screens, images, regions and viewports. The logical entities are superwindows and windows. The mapping from logical to physical entities is achieved through the use of configurations, which is represented in Figure 3.3.

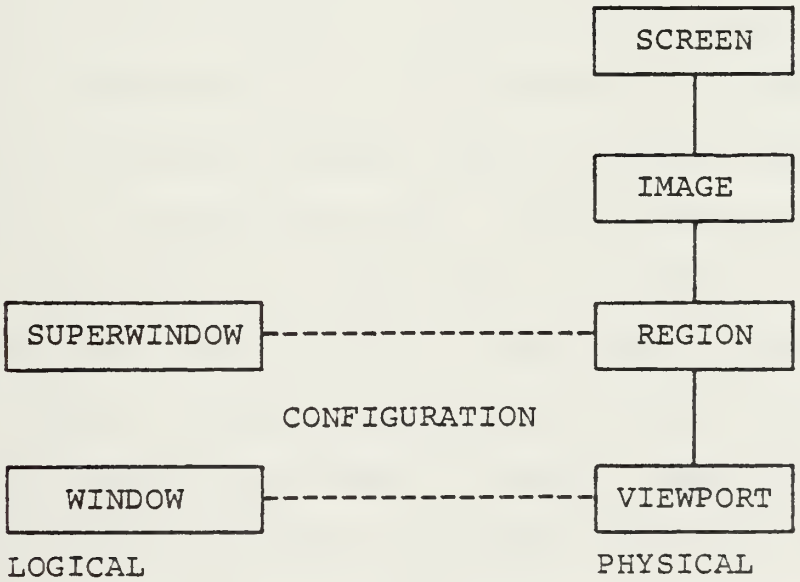


Figure 3.3 Logical and Physical Screen Configuration.

The window is the space component of a virtual terminal that represents a potential mapping of the output contained in a user's Pad onto their display device. Its attributes include preferred contrast, cursor blinking, etc., and upper and lower limits of the window size when displayed. The superwindow is a logical entity that represents all windows associated with a virtual terminal of a particular logical process. The grouping of windows within superwindows insures the contiguous display of all output associated with a particular user or application process.

A screen is the visible space on a display, and has a fixed height and width. Processes outside the user's virtual terminal controller do not know about the screen characteristics unless they explicitly request the display profile from the terminal output handler. Since the realizable display is limited in size, it is desirable to have multiple screen images, each of which may be treated as the physical screen. This is necessary in cases where the user wishes to allocate the entire screen to a particular process, while allowing other processes to continue in a background mode. Those processes may then be mapped to the screen at a later time. An image is what a screen might look like at any one time and contains a subset of those virtual terminals associated with the user's session. The user may define any number of images, swapping between them through the use of specially programmed functions keys. If the user wants to observe multiple processes simultaneously, it

may be desirable to map their output onto the display simultaneously, i.e., map them into the same terminal image.

Each process is allocated to a region of the screen which would be mapped to the virtual terminal associated with that process. Thus, an image is composed of a set of contiguous regions. A region is a collection of contiguous viewports. The size of a viewport is dependent upon the size of its associated region and the bounds specified for its associated window.

A configuration is a description of the way in which a subset of a user's virtual terminals should be displayed. It specifies the relative positions of the windows, their relative sizes as a percentage of the whole and actual viewing conditions. It is through configurations that virtual terminals are actually mapped to the screen. A process may configure its virtual terminals in as many ways as it desires, but is not aware of which configuration is currently active. Only the user is aware of the current configuration presented and can swap configurations via specially programmed keyboard keys.

A line represents a logical keyboard and Pads represent logical displays. Their physical counterparts comprise a terminal. The line handler communicates with the terminal input handler which manages the keyboard. Pad Handlers communicate with the terminal output handler which manages the terminal display. The input and output handlers are the

process-level interface to the interrupt-level I/O handlers. There is exactly one input and output handler for each terminal in the LAN. The physical characteristics of the terminal with which any LAN process is associated may be obtained via declarative keyboard and display profiles. The physical characteristics of a user's physical terminal could be obtained by the mapping of a physical terminal ID onto a generic terminal transformation table. One possible format for the support of multiple types of physical terminals is presented by B. C. Hillsberg, and is presented in Figure 3.4 [Ref. 26]. Each column of the transformation table contains a set of transformation identifications which must be applied to the generic codes to correctly invoke the desired terminal sub-function.

PHYSICAL TERMINAL	SCREEN CLEAR	CURSOR HOME	POSITION CURSOR	CURSOR UP	CURSOR DOWN	CURSOR LEFT
INFOTON	T1	T1	T1	T1	T1	T1
IBM 3101	T1	T1	T2	T1	T1	T1
HDS	T2	T2	T3	T2	T2	T2
TVI-912	T3	T3	T4	T3	T3	T3
ADM-3A	T3	T3	T4	T3	TT3	T3

Figure 3.4 Generic Terminal Transformation Table.

A keyboard is considered to be any device capable of generating distinct signals in response to user input. A keyboard

profile is constructed by the terminal input handler and maps the signals generated by the keyboard into the virtual terminal control functions which they represent. An example of this could be the EBCDIC characters 15 which could be mapped into the function DELETECHARLEFT. Each character may have at most one control function, but a control function may be generated by more than one set of characters.

Characters which are not mapped are given no special interpretation by the virtual terminal controller. Only the terminal input handler relies on particular signals representing a particular control function. The function of a terminal input handler is to collect data from a user's keyboard, usually in response to a request for input from a line handler. The terminal input handler is also responsible for creating and maintaining the keyboard profile.

A display is considered to be any device on which some bounded number of text-lines may be shown simultaneously. The format of display commands have evolved out of a desire to minimize both the amount of state information maintained by the terminal output handler, and make the user's terminal transparent. The commands attempt to incorporate the features provided by most terminals, while leaving out some features provided by more intelligent terminals. The most used commands are: alert (ring bell), clear to end of line, delete line, insert line, clear screen, delete character, insert character, move cursor, write character, and many more.

A display profile would be constructed by the terminal output handler and would contain the height and width of the display and the contrast characteristics supported by the particular user's terminal. The display profile is provided to any user's process upon request and is required by the screen and Pad Handlers. The function of a terminal output handler is to translate the terminal commands into control signals which the particular display can understand. The terminal output handler is also responsible for creating and maintaining the display profile, and providing it to LAN user processes upon request.

There are four general classes of control functions which would normally be provided by the Terminal Manager.

1. Screen and command input functions:
   Aborttask, Autoblock, Changeconfiguration, Changeimage, Changeregion, Changeviewport, etc.

2. Line editing functions:
   Cursorleft, Cursorright, Cursorwordleft, Deletecharleft, Deletewordright, Insertmode, etc.

3. Page editing functions:
   Cursordown, Cursorleft, Cursorup, Splitline, Cursorwordleft, Deletecharleft, Pagedown, Pageup, Joinline, etc.

4. General screen management functions:
   Assignlot, Cancel, Comment, Execute, Expand, Help, Prompt.

It is recognized that the degree of definition for terminal control functions is dependent upon the amount of sophistication desired by the using organization. The control functions described above only represent a general breakdown of the

control functions which should be provided for the support of multiple user processes within each user's session with the SPLICE LAN.

Additionally, the Terminal Manager should provide the facilities for the establishment of user mail boxes to be used for inter-user message transmission. The mail boxes could be used to communicate organizational message text between users of the LAN, and could be used for user queries or for automatic user notification by the Terminal Manager. Automatic user notification could be performed immediately after the users completed the log-on process with the Terminal Manager and before the user begins an interactive session with the LAN.

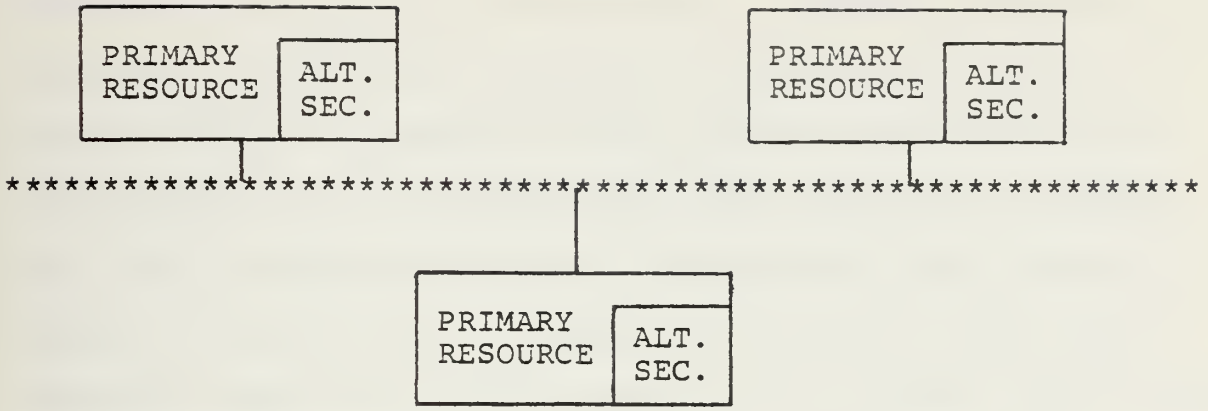
Submission of message text to a particular user, by another, would require verification of the destination user prior to acceptance and subsequent building of an appropriate mail box. Regardless of whether a message text is accepted by the Terminal Manager, acknowledgement or denial of the request should be transmitted to the requesting LAN user. Denials for the building of a mail box should also provide an indication of the reasons for denial. Common or community mail boxes should also be provided in order to transmit non-specific organizational messages to all users of the LAN or a subset thereof.

D. BACK-UP AND RECOVERY

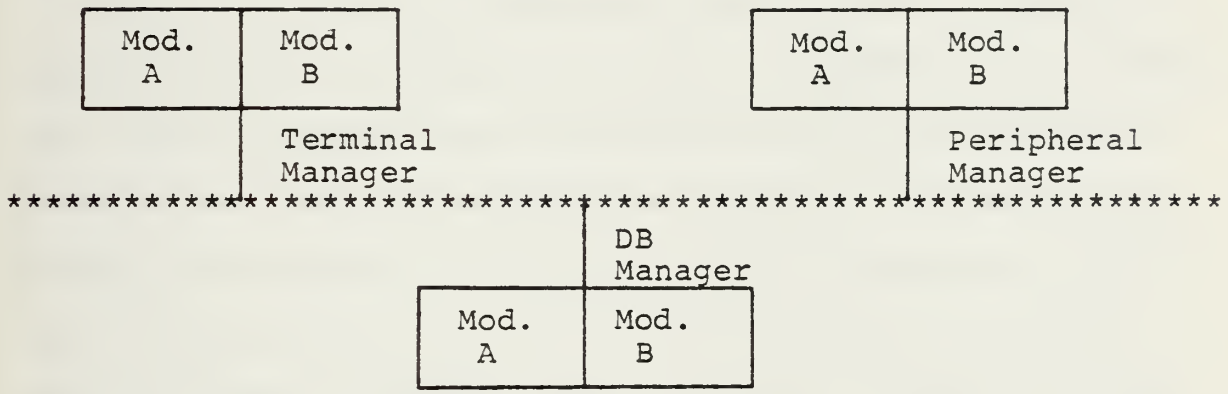
In considering the distribution of functional resources within the LAN, one must provide a mechanism by which failed

LAN resources can be recognized, and the functions restored regardless of the state of the processors upon which they might run. Back-up and recovery is particularly important in support of the SPLICE requirement such that no one failure of a LAN resource will terminate support of the SPLICE organizational requirements. Although there are a multitude of methods which could be employed to provide back-up and recovery, i.e., multiple copies of functional resources across the LAN, use of a semi-vertical partitioning of functional resources is recommended. This is a concept introduced by Professor J. Popek, University of California at Los Angeles [Ref. 16].

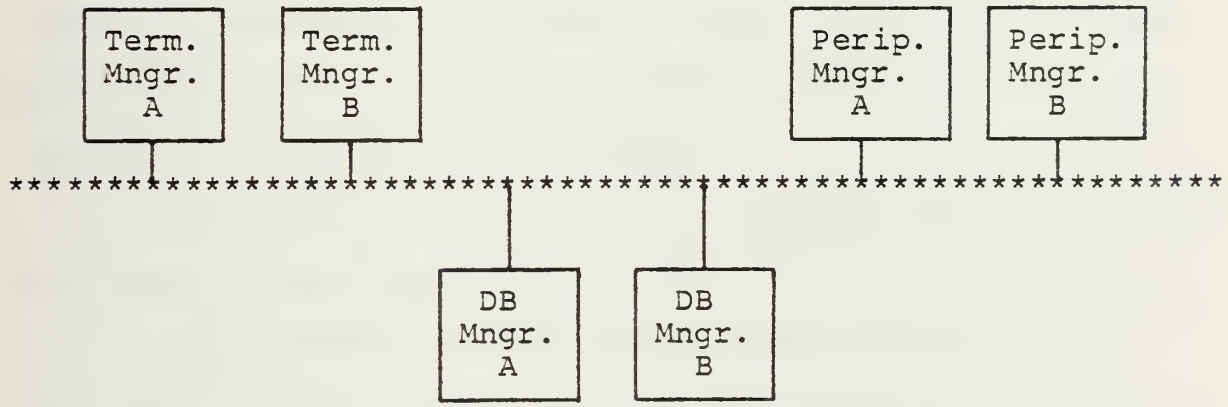
In general, the concept of the vertical layering and partitioning in distributed computing concerns permitting all processors to be capable of any function, thereby provides a type of "graceful degradation" of system performance in the case of malfunctioning nodes and resources. In order to provide for a type of "graceful degradation" when a functional resource is determined to be inoperative, each node would be required to support a secondary and passive LAN functional resource. A passive secondary resource, as used in this thesis, represents a functional resource which is fully duplicated at an alternate node and which does not perform any user tasks unless the primary functional resource becomes inoperative. Figure 3.5 represents three possible methods of physically implementing back-up functional resources to insure the



a. Primary and Alternate Secondary LAN Resource.



b. Tandem Parallel Processor Architecture.



c. Parallel Processing with Independent Processors.

Figure 3.5 Example LAN Functional Resource Implementation.

reliability of overall LAN operation. Diagram A, Primary and Alternate Secondary LAN Resources, illustrates the use of a single complete processor for each LAN resource which would be designated as the primary functional resource. In addition, each complete processor, representing a node, would be capable of supporting a secondary alternate LAN functional resource, should the primary resource, at another node on the LAN, become inoperative.

Diagram B, Parallel Processor Architecture, represents a concept which is currently available in systems such as Tandem. This type of architecture processes each user transaction in parallel and performs intermittent checks to insure the concurrency of results. In the event that a discrepancy should exist between results or one of the processors should fail, the operational processor would continue to support LAN and user functional requirements. Such systems generally provide additional redundancy by double writing files which provides additional back-up in the event that a previously written record or file should become damaged.

Diagram C is an extension of the Tandem architecture, but provides for the duplication of the LAN Transport mechanism. In so doing, support of LAN resource requirements can be achieved even in the event of the node connection to the LAN. This thesis provides a Back-up and Recovery methodology for the Primary and Alternate Secondary LAN Resource support but is also applicable to the other two methodologies. By use of

passive secondary functional resource allocation and the logging of the respective LAN messages, failed resources could be reinitiated on an alternate operating LAN node, once the determination of a failed resource is made. This provides resource capability in depth, without requiring complex load balancing and control in the case of multiple copies of functional resources.

The requirements for the semi-vertical partitioning of functional resources are that each secondary functional resource transport subsystem must, either directly or indirectly, receive functional resource and user state information. This resource and user state information could be attained by the issuance of pertinent state information to the secondary functional resource node by the primary. The Session Services Subsystem of the secondary must direct these messages to a log or journal file for future reference. These files would be used in bringing the secondary functional resource on-line and designating it as the primary for the LAN. In order for LAN functional resources to be able to determine the operability of other LAN functional resources, each functional node would randomly generate control query messages to each primary functional resource, as indicated by their corresponding resource directory. By selection of an appropriate delta "t" for the receipt of a control acknowledgement, each functional resource should be reasonably certain of the operability, or lack thereof, of each LAN functional node

and resource. Since the secondary functional resource operates in a passive role, there is no direct necessity for its functional software to be resident while the functions are being supported by the primary functional resource. Of prime importance is the maintenance of user session messages and a mechanism for the reconstruction of controlling session services for users affected by inoperative LAN functional resources.

When exploring the idea of passive secondary functional resources, there were two major problems encountered. First, the reconstruction of the Controlling Session Services for each verified user of the LAN has to be supported with a minimum of delay. Second, since message communication between functional resources, in support of LAN users, is bound to be extensive for interactive processing, there must exist a method for the purging or masking of user and LAN resource messages which do not change the state of the LAN resource or user state information. To resolve these problems, the functional resources must operate in an interactive fashion on both the user to functional resource level and the functional resource to functional resource level.

For the user of the LAN, it is assumed that the initial interaction will occur between the user and the Terminal Manager, via the Front End Manager. In the initial stages, user verification and security procedures, along with terminal support, i.e., terminal format characteristics, will be performed prior to the establishment of a user's controlling

session service. It should be noted that the Front End Manager acts as a gateway to the LAN and since all user messages are vectored to the Terminal Manager, the ability to reconstruct Controlling Session Services maintains the access security of the LAN. Once the Terminal Manager has verified a user of the LAN, it would generate a message to the secondary functional resource, indicating the result of its interaction. In this manner, the secondary resource would record only the atomic results of the corresponding messages.

If the Terminal Manager were to assume responsibility for providing the Controlling Session Service, then it would generate a message to the secondary resource indicating such control and any coordinating or additional control information established at the session services level. In so doing, should the Terminal Manager become inoperative, the secondary resource could be made current by use of the secondary resource journal file without requiring the examination of sequential message exchange between the user and the primary resource. Since the secondary Terminal Manager is a duplicate of the primary, containing user virtual terminal characteristics, only changes need be identified in order to maintain the concurrency of the user's virtual terminals. Once the terminal user either terminates their LAN session or issues an "end virtual terminal" message, transient terminal format modifications can be purged. Only the results of message exchange between the primary resource and the user are of importance in making the secondary

resource reflect the last recorded state of the primary resource. Controlling session services can be reconstructed and user support maintained with a minimum delay. Once a user session is terminated, the session messages can be transferred to a historical journal file. This permits optimization of the performance of the active journal file, eliminating the necessity for the maintenance of inactive user sessions.

The effect of user sessions in respect to the time dependence of user transactions to other user sessions is of prime importance in maintaining the accuracy of the LAN data base. In terms of primary and secondary functions, the LAN data base is duplicated at the secondary LAN node and all future actions which modify the primary data base must be maintained by the secondary such that their effects can be duplicated in a manner analogous to their actual effect on the primary. Read-only transactions are mutually transitive in that the order in which they are executed has no effect on the state of the data base. Since read-only actions have no effect, there is no need to maintain a record of them except to provide relatively recent reference and historical accounting. Although both the primary and secondary functional DBMSs have complete copies of the LAN data base, the requirements for back-up and recovery encompass many of the problems of a partitioned data base.

A recommended approach to ensuring the consistency of the primary and secondary DB manager is to use a form of two-phase

commit, which was described by Jim Gray, Jr. of IBM [Refs. 12, 21]. Although the secondary data base manager would record all recent message traffic by user UID, it would be prohibitively costly for the secondary to apply each of these against the secondary data base once the primary has failed. By applying a type of time-stamp to each user transaction, both the relationship of transactions to user UIDs and the relative order of user transactions to the user session can be maintained.

The atomic properties of user transactions can be maintained by using the primary Database Manager as the control mechanism for data updates, coupled with time-stamps on user messages. As the secondary records data base messages, the primary coordinates the references of user messages to the physical data base and applies the authorized actions. Once an update to a data item has been applied, the primary Database Manager sends a commit message to the secondary indicating the user UID, start and stop time-stamps, and the results of the user transaction on the data base. Upon receipt of the commit message from the primary, the secondary would purge all user messages encompassed by the start and stop time-stamp and record the resulting value of the data base on the journal file. This creates a type of partitioned journal file containing Controlling Session Service information, ordered user messages in time-stamp order and the resulting atomic properties of applied user transactions, Figure 2.4.

Once the primary Database Manager is recognized as being inoperative, the functional resource making the determination must send a control message to the secondary Database Manager with a time-stamp indicating the approximate time of fault recognition. Upon receipt of this control message, the secondary Database Manager would become resident, apply all entries in the "modified data item log" to the data base, and establish a Controlling Session Service for each user session entry in the active "control session service log" of Figure 2.4. By using the time-stamp of the activating resource control message, the secondary Database Manager would determine a reference point for each user entry in the active "message log," and determine its application to the data base in the same manner as that used by the primary Database Manager. Once all references have been resolved, based upon the initiating resource control message, the secondary Database Manager would proceed to process all preceding user messages. Since the state of the secondary is based upon the resolution of references as determined by the time-stamp of the initiating resource control message, this information must be retained, i.e., recovery control data, along with a log of all subsequent data item updates for the re-establishment of the primary Database Manager, once it becomes operational.

Once the failed primary Database Manager becomes operational, an exchange of information must take place between the primary and secondary prior to the assumption of Database

responsibilities by the primary. The primary Database Manager would initially establish a checkpoint indicating the last message acted upon prior to becoming inoperative. By requesting the recovery control data from the secondary, the primary would resolve any differences which may exist between the secondary's view of the data base at the time of failure and the actual state of the data base on the part of the primary.

IV. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

In reference to the design of a functionally fully distributed Local Area Network, little research of a practical nature has been performed on LAN architectures. The fact that the SPLICE LAN is designed to operate on a bus structure has further narrowed the range of practical design and implementation concepts available to the authors. Two primary problems encountered in attempting to provide functional design specifications were the vertical and horizontal partitioning of functional modules within the LAN and the establishment of acceptable protocols to enforce their interrelationships.

In order to provide the greatest degree of LAN flexibility possible, this thesis attempted to incorporate a horizontal partitioning of functional resource subsystems within vertically partitioned functional modules. The horizontal partitioning of the Transport, Session Services and functional resource subsystems provides a means for future LAN expansion of defined LAN resources without requiring a high degree of modification to the existing LAN architecture. The horizontal partitioning within each LAN functional resource and the distribution of a Session Services Subsystem across all nodes of the LAN is also designed to provide a higher degree of LAN reliability in the face of node failures. The continued

support of organizational needs, in spite of failed functional resources, is considered to be paramount in the design of the LAN. If the operability of the LAN can be terminated by the failure of any single node, then the LAN has the potential to be highly unreliable. It is through the use of vertical partitioning that the reliability of the LAN design is increased. Although it may be more desirable to isolate particular functions and have them performed by a single module, the singular existence of that module, and its availability, may result in the eventual failure of the LAN.

The discussion on back-up and recovery demonstrated the complexity of providing a highly reliable system and was based upon the existence of a uni-processor minicomputer at each node. By use of a Tandem type system, the issues of primary and secondary functional resources would become almost transparent to each other. Such a system would not only provide for a highly fault tolerant and reliable system, but would also eliminate many of the control structures necessary to maintain the integrity of the LAN by implementing these control structures in hardware components.

The Data Base and Terminal Manager subsystems identified in this thesis present a description of their basic tools and capabilities. Although the identification of a particular DBMS was beyond the scope of this thesis, there exists a variety of methods for its implementation on the LAN. Careful consideration of the interrelationships between the Stock

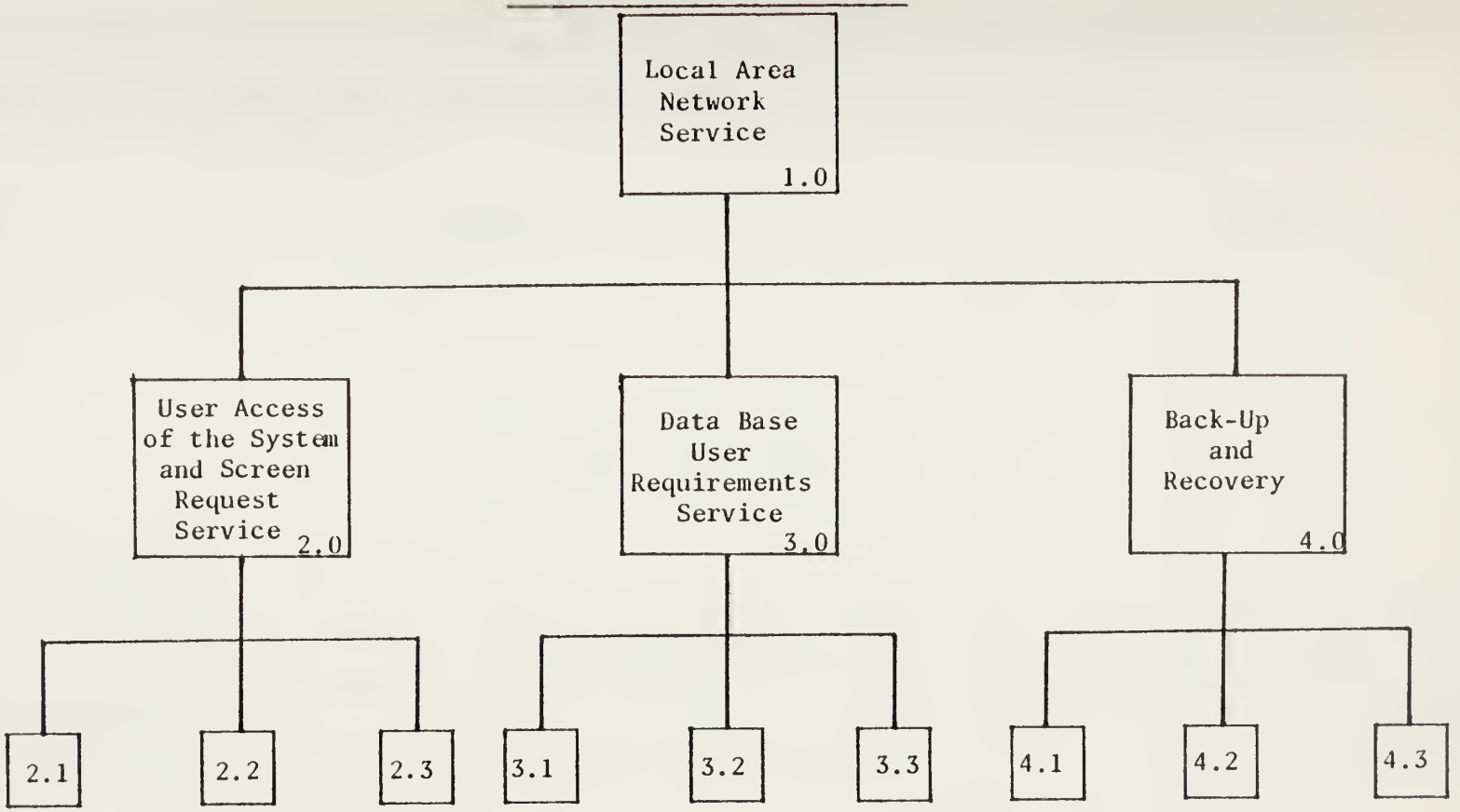
Points and Inventory Control Points should be made in the use of any DBMS to support long range organizational objectives. The Terminal Manager subsystem also presents a variety of capabilities available for implementation on a LAN. Since it is the terminal which provides the physical interface between the users and the LAN, it must be capable of reflecting a variety of user work environments.

B. RECOMMENDATIONS

It is recommended that the functional design of a LAN provide the greatest degree of functional distribution while providing for the vertical partitioning of subsystem functions in order to provide a high degree of flexibility for future growth. It is further recommended that a Tandem type system be used to support the LAN functional resources, thereby eliminating many of the control structures needed to maintain the integrity of the LAN as a whole.

In that this thesis only provides a framework for further functional design considerations, it is recommended that additional research be performed in the areas of security, management of functional resources, load estimations upon each SPLICE LAN, and the capabilities and practicality of using a high level query language vice a high level programming language for interactive data processing.

A VISUAL TABLE OF CONTENTS



2.1 Transport Service

2.2 Session Service

2.3 Terminal Management Facilities

3.1 Transport Service

3.2 Session Service

3.3 Data Base Communication and Control

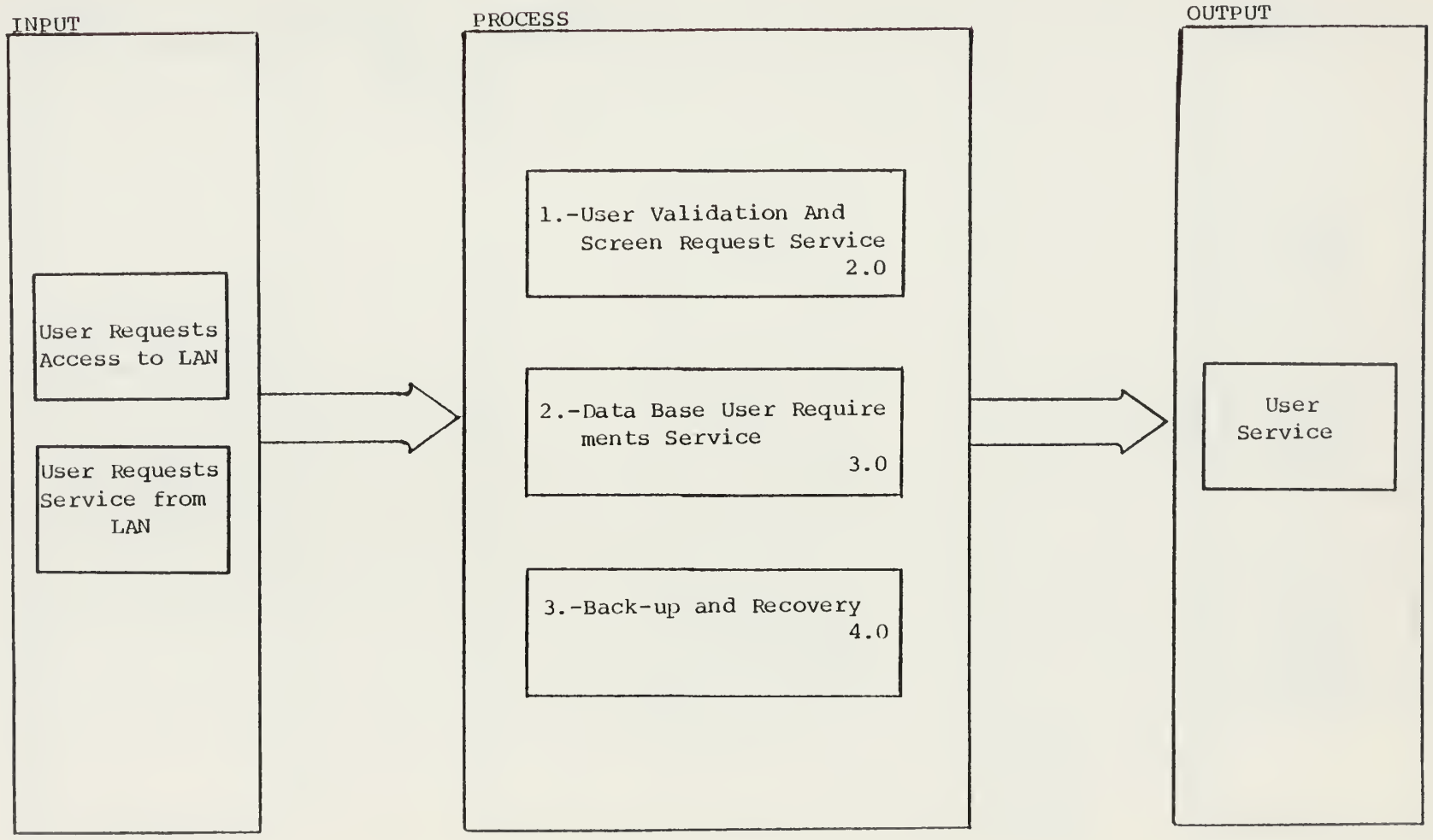
4.1 Detection of a Failed Resource

4.2 LAN Attention Message

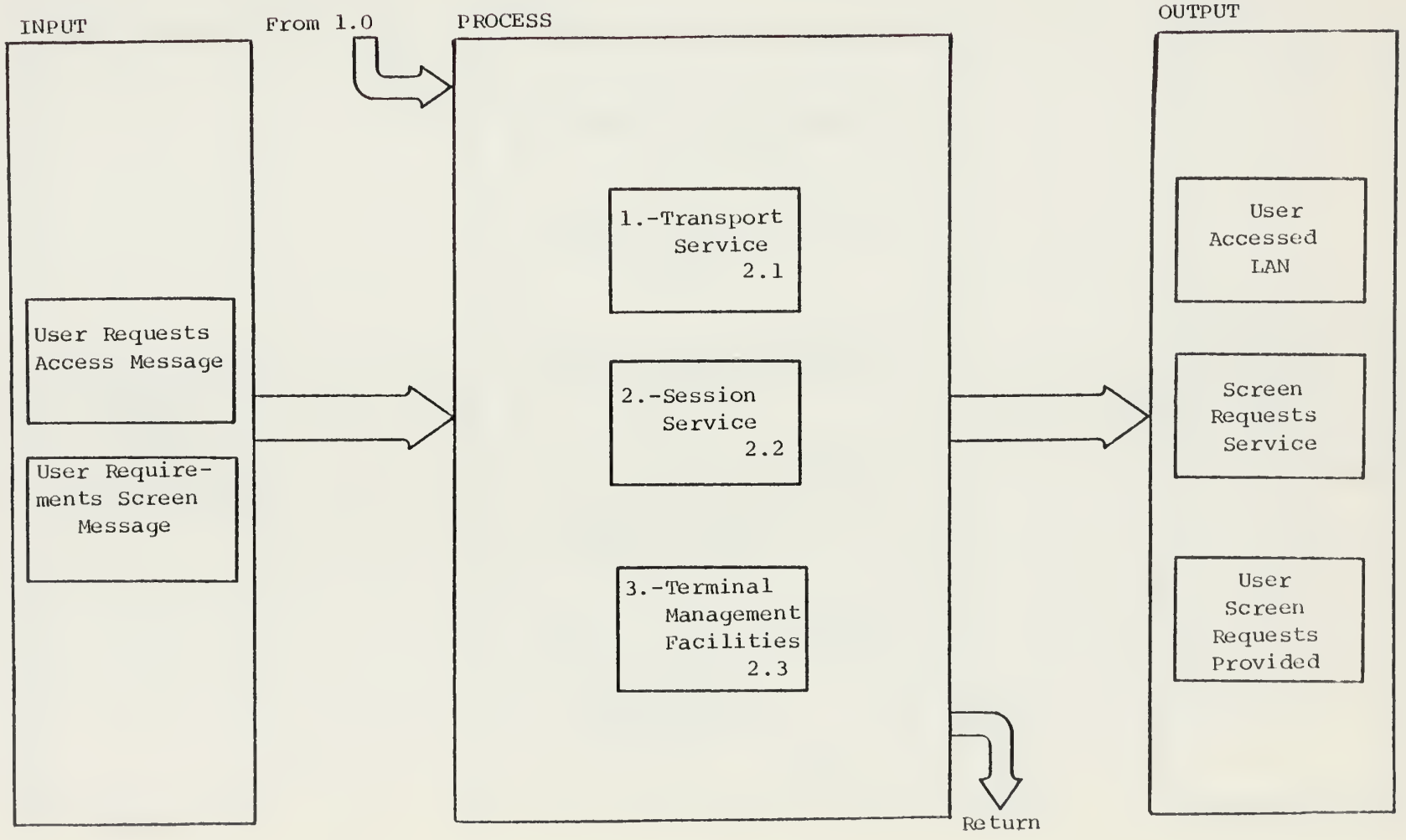
4.3 Back-Up and Recovery Process

Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23,1982

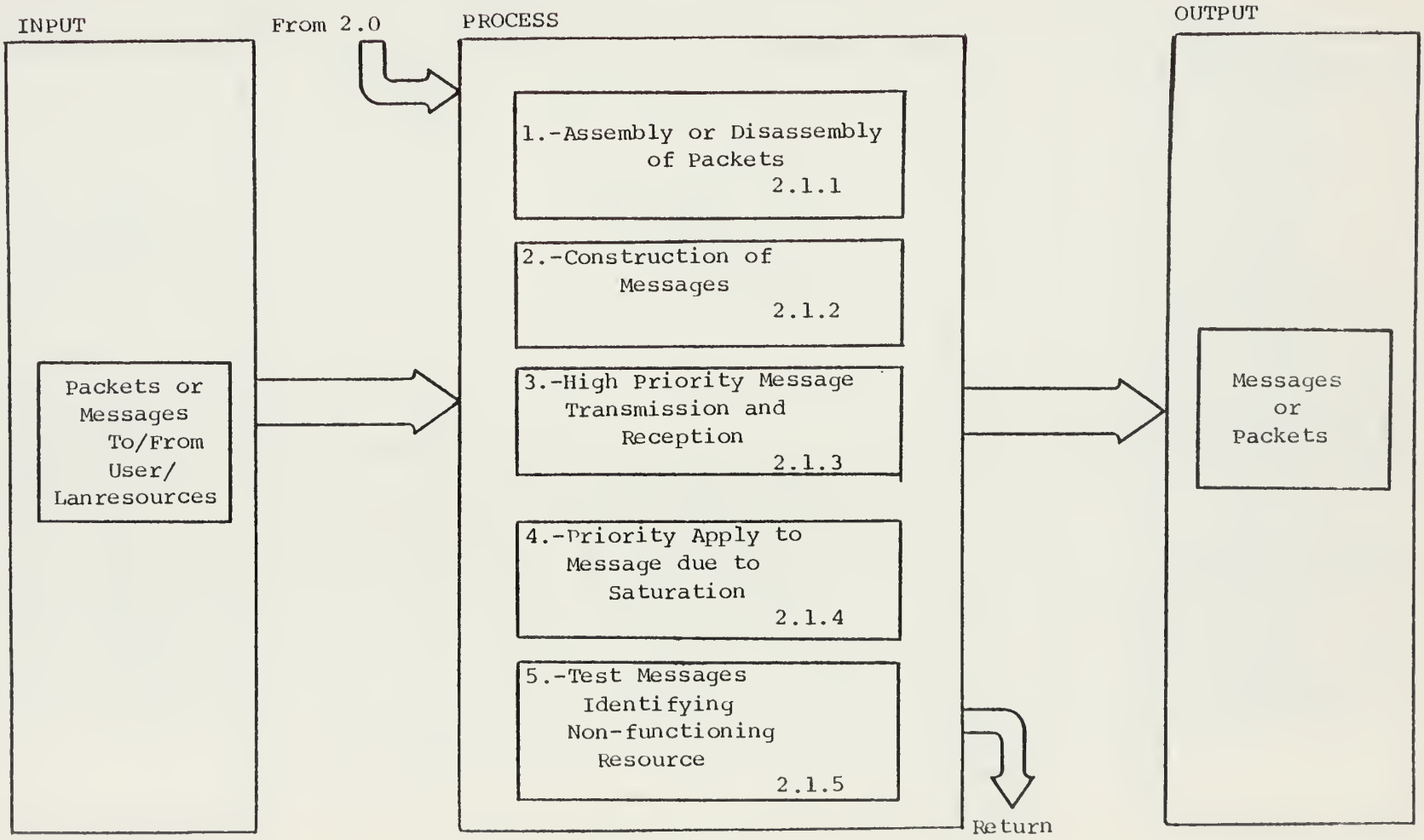
Diagram Id: 1.0 Name:LAN001 Description:Hipo-Overview



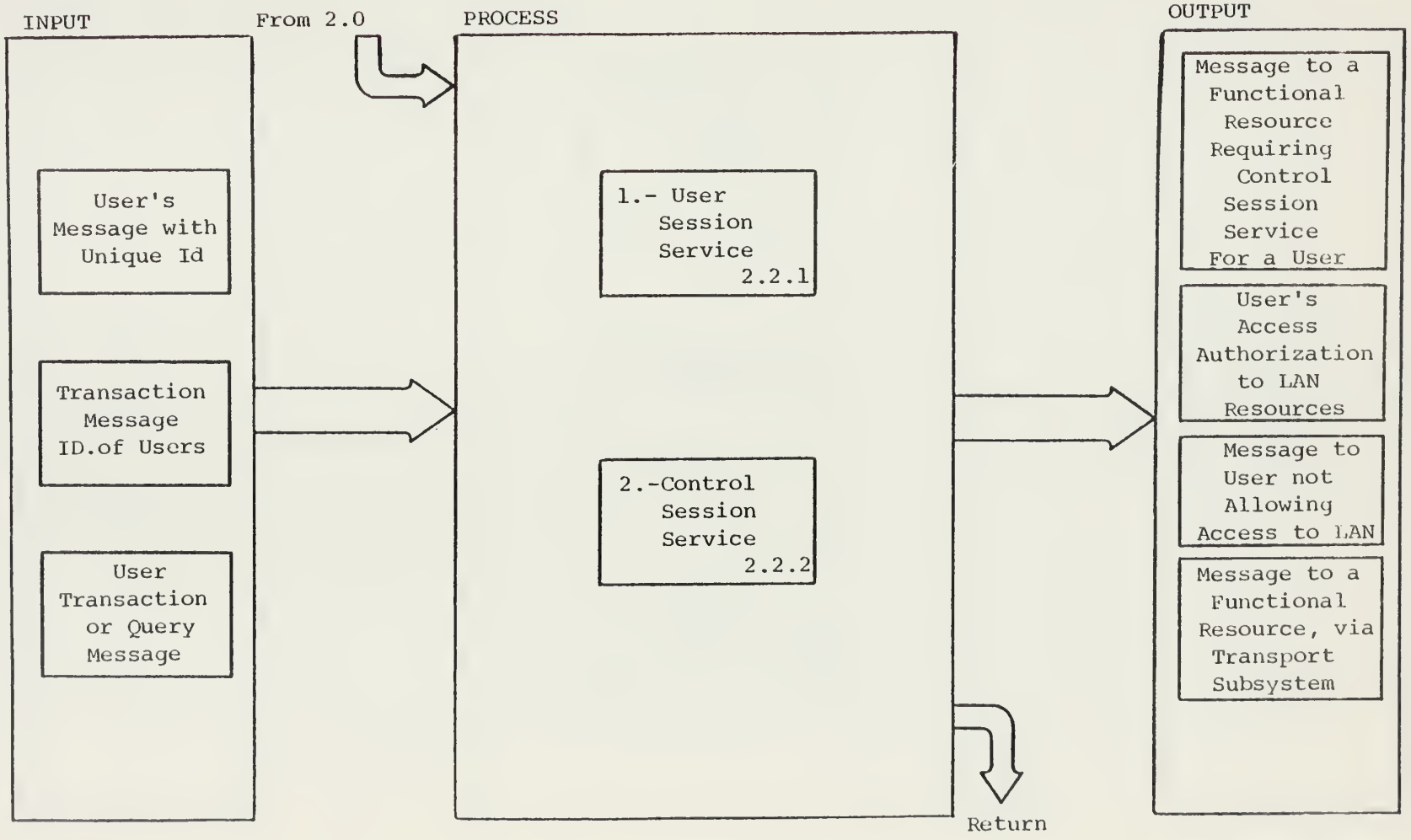
Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23,1982
Diagram Id: 2.0 Name:LAN002 Description:User Validation And Screen Request Service



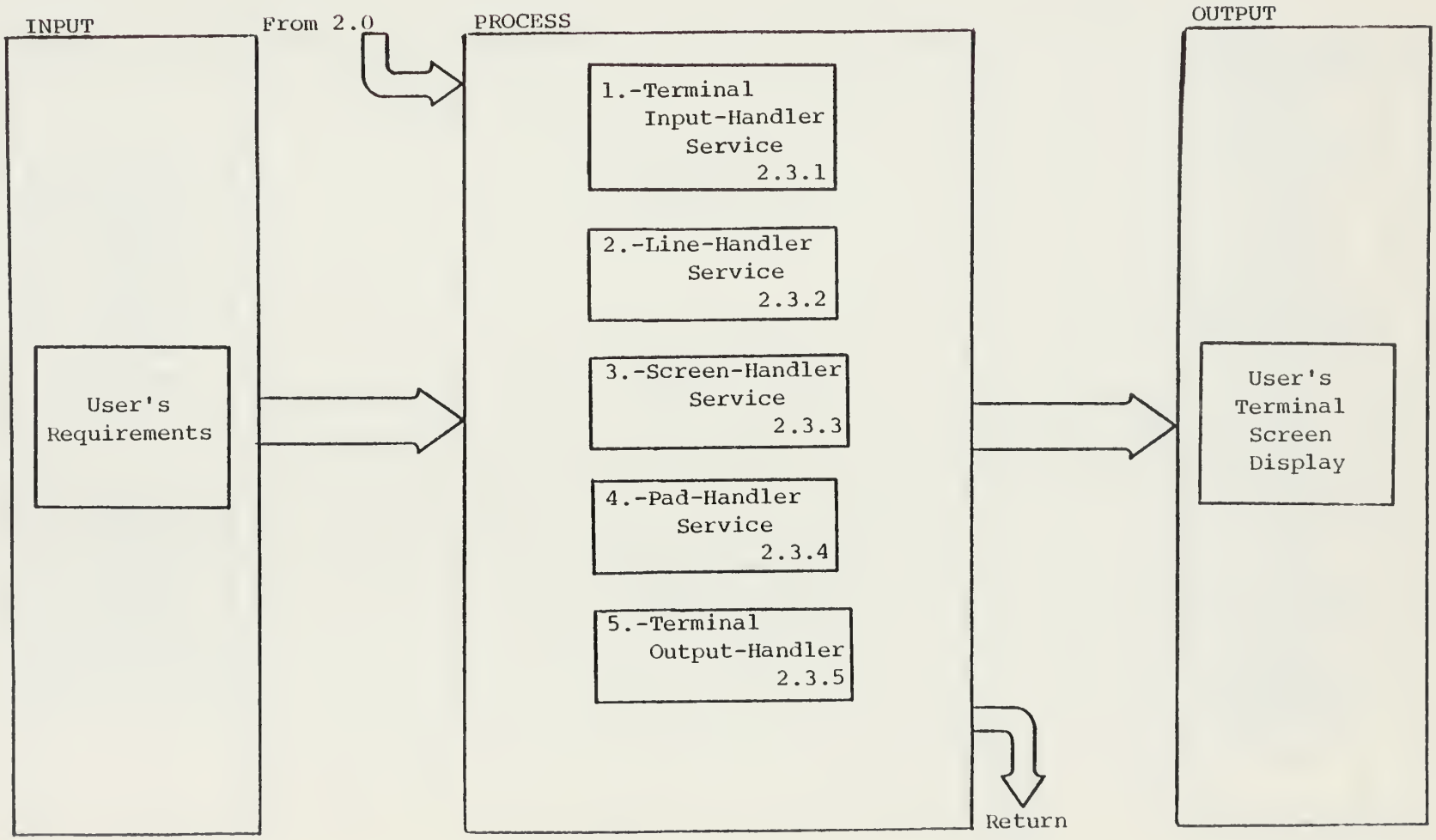
Author: R. Arana/J. Reinhart System/Program: Local Area Network Splice Date: May 23,1982
Diagram Id: 2.1 Name: LAN005 Description: Transport Service



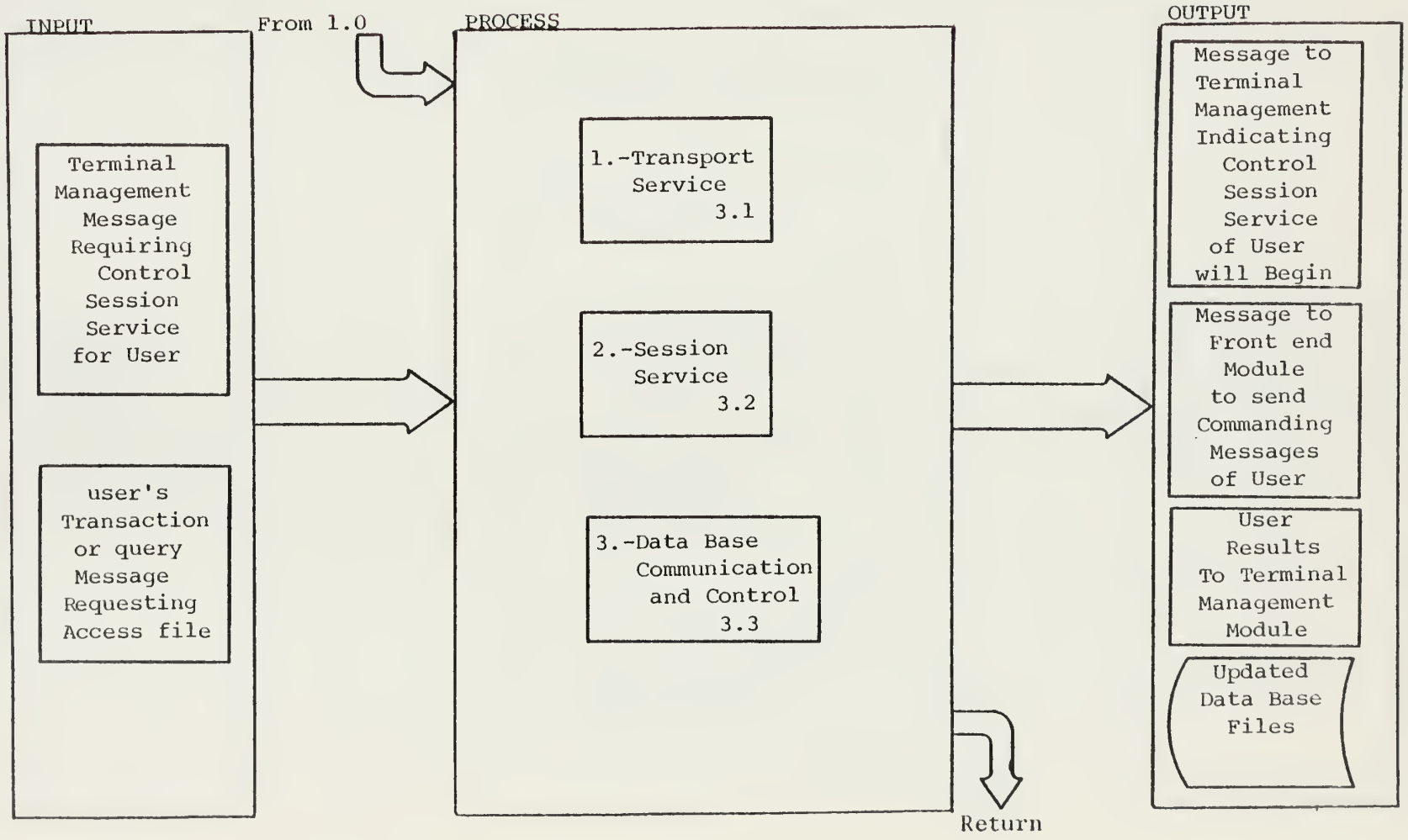
Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23,1982
Diagram Id: 2.2 Name:LAN006 Description:Session Service



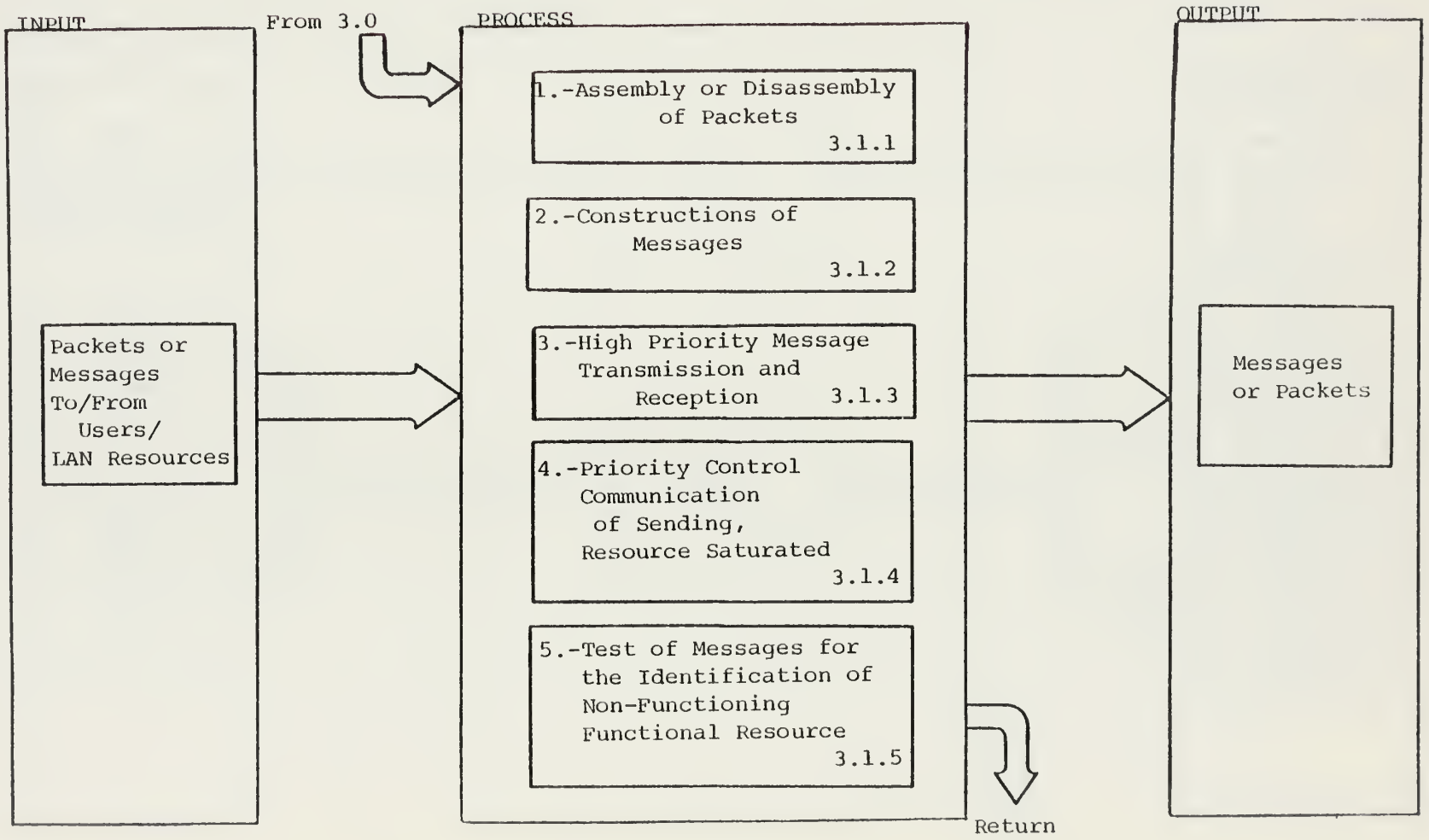
Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23, 1982
Diagram ID: 2.3 Name:LAN007 Description:Terminal Management Facilities



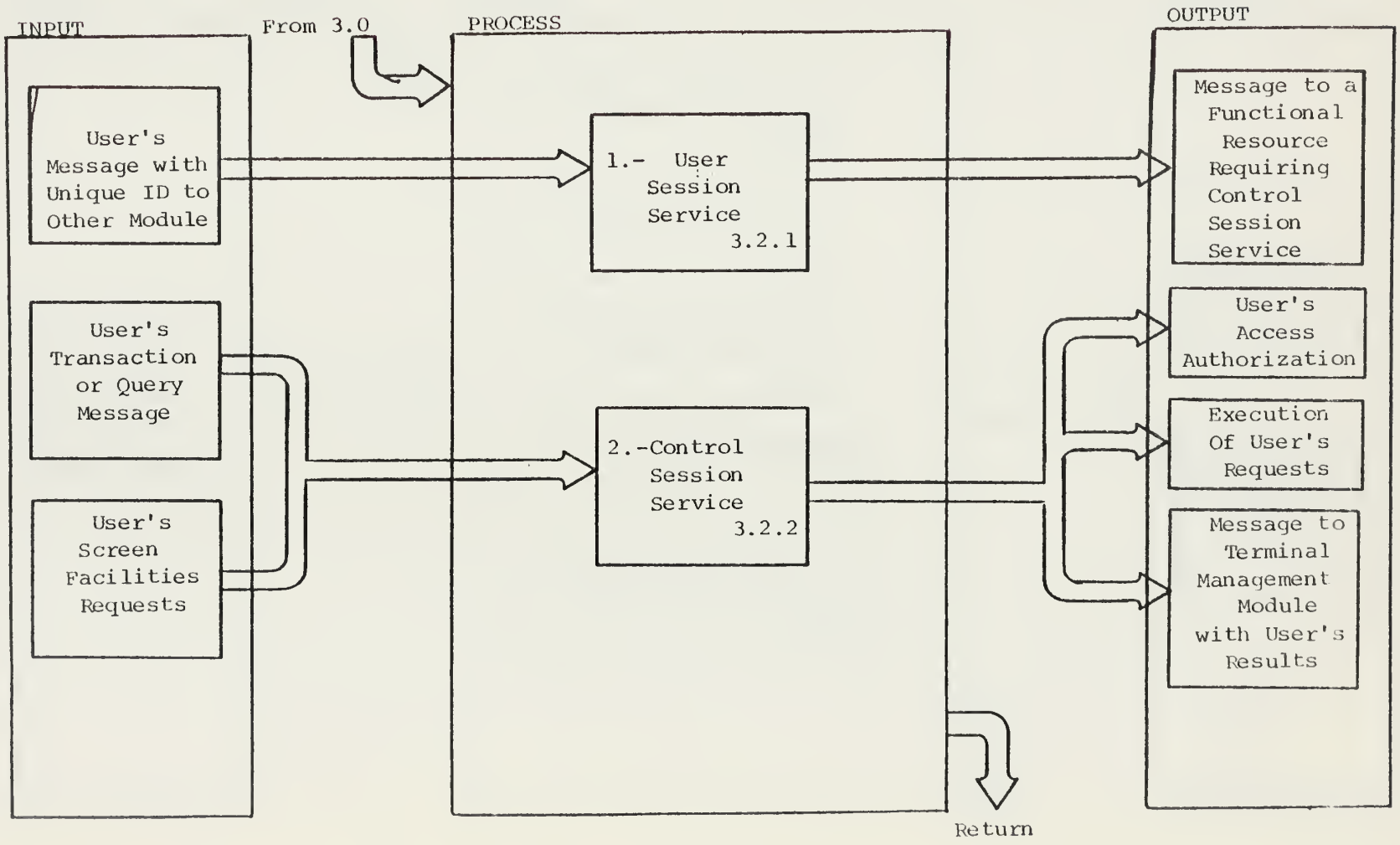
Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23,1982
Diagram ID: 3.0 Name:LAN003 Description:Data Base Management Module (Resource)



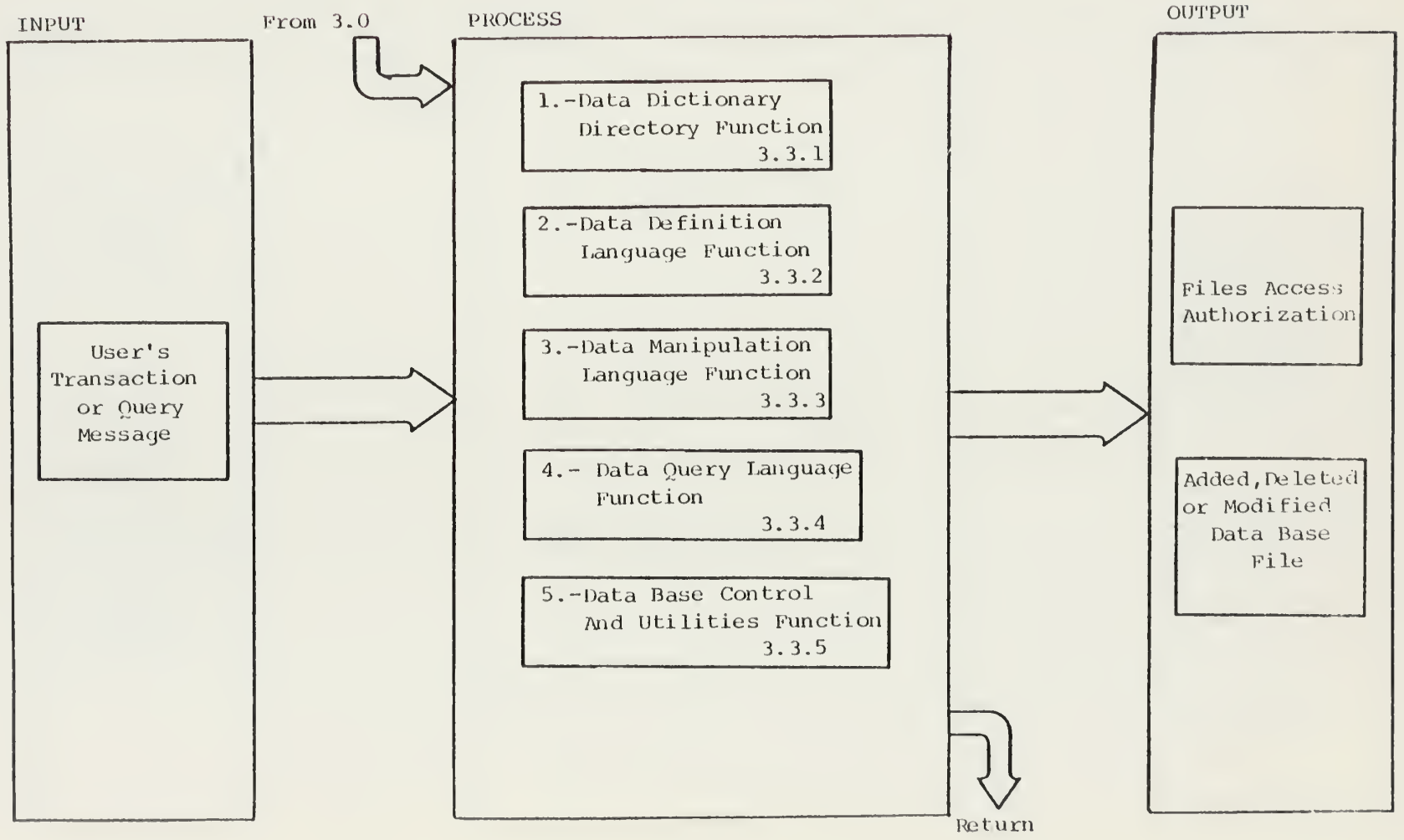
Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23,1982
Diagram ID: 3.1 Name:LAN008 Description:Transport Service



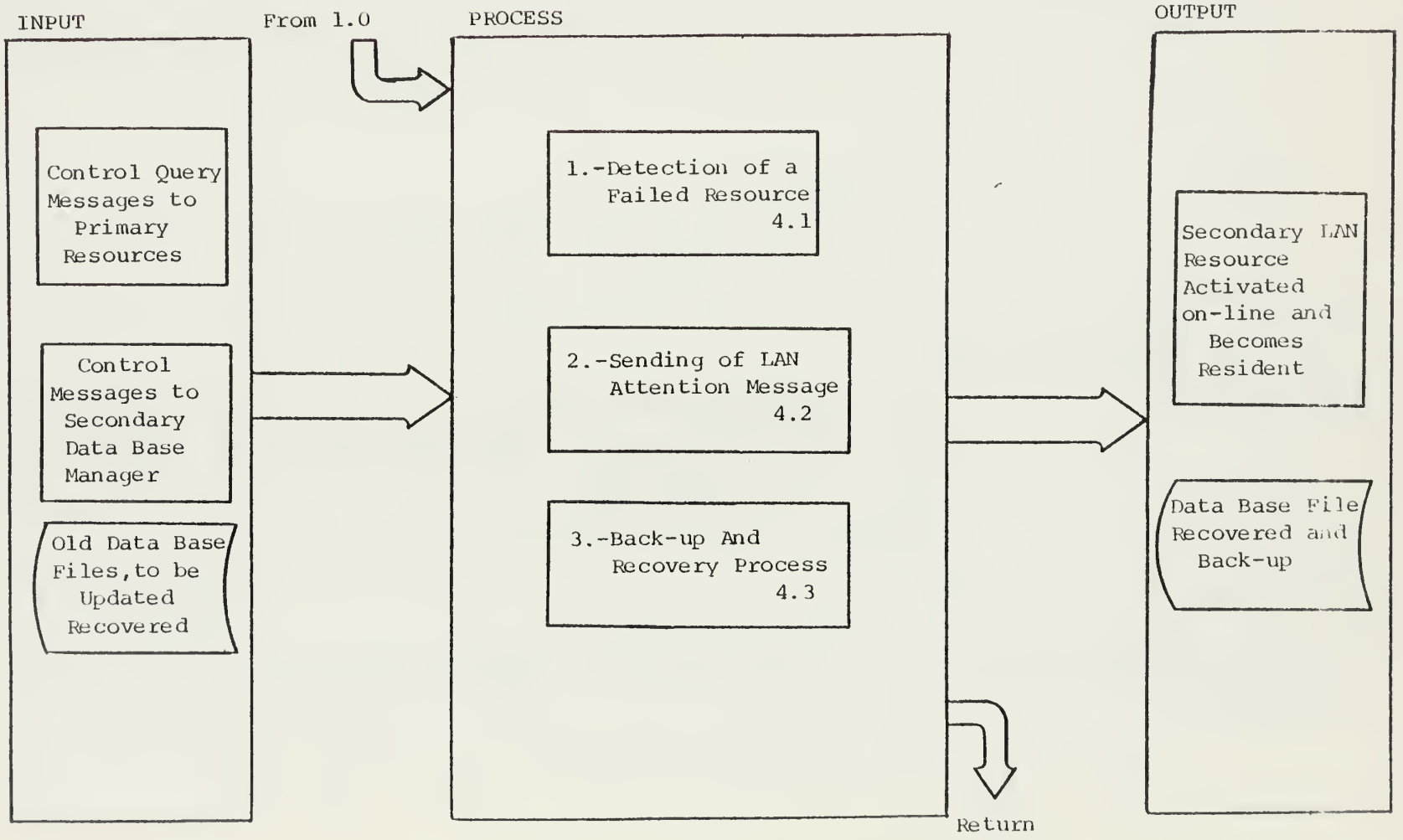
Author: R. Arana/J. Reinhart System/Program: Local Area Network Splice Date: May 23,1982
Diagram ID: 3.2 Name: LAN009 Description: Session Service



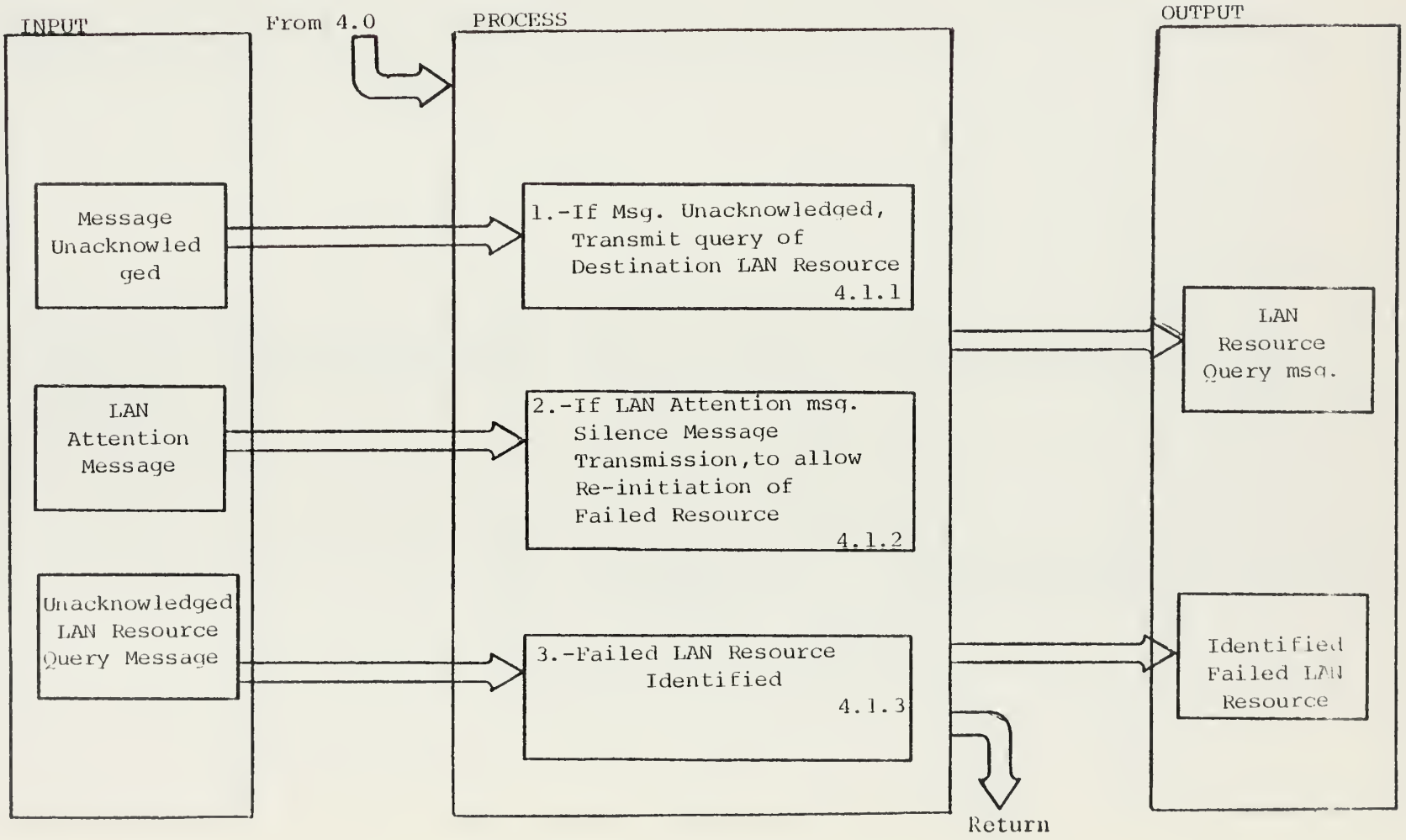
Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23,1982
Diagram ID: 3.3 Name:LAN010 Description:Data Base Communication And Control



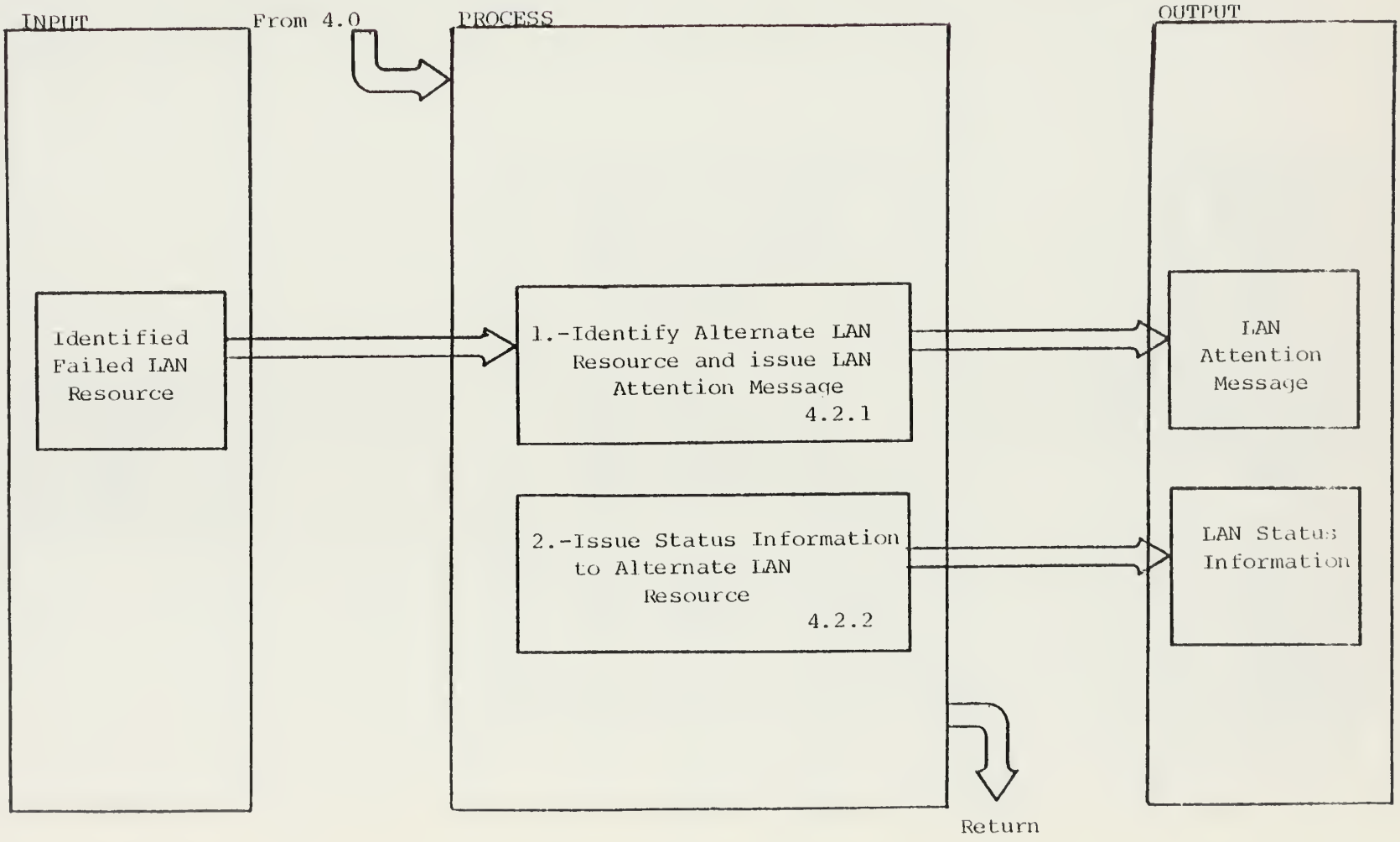
Author: R. Arana/J. Reinhart System/Program: Local Area Network Splice Date: May 23,1982
Diagram ID: 4.0 Name: LAN004 Description: Back-up And Recovery



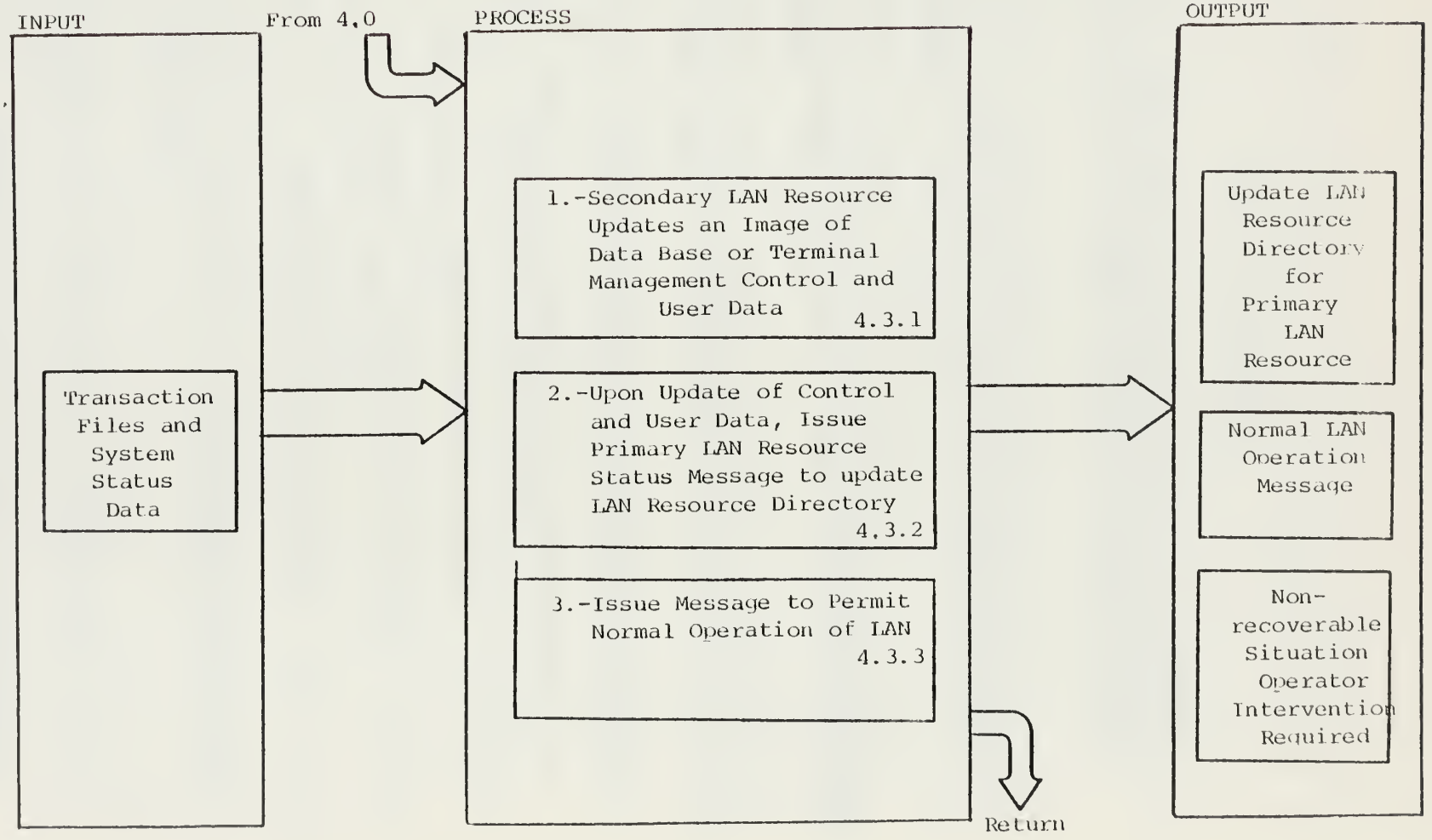
Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23,1982
Diagram ID: 4.1 Name:LAN011 Description:Detection of a Failed Resource



Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date:May 23,1982
Diagram ID: 4.2 Name:LAN012 Description:Send Silent On The Network



Author:R.Arana/J.Reinhart System/Program:Local Area Network Splice Date;May 23,1982
Diagram ID: 4.3 Name:LAN013 Description: Back-up and Recovery Process



LIST OF REFERENCES

1. U.S. Navy Fleet Material Support Office, Environment Division: Code 9441, Stock Point Logistics Integrated Communications Environment (SPLICE), Software Design, 19 March 1979.

2. Fleet Material Support Office, Department of the Navy, Document No. F94L0-001-9260-SS-SU01, Stock Point Logistics Integrated Communications Environment (SPLICE) System Specification, 2 February 1981.

3. Fleet Material Support Office, Department of the Navy, Document No. F94L0-001-9260-FD-SU01, Stock Point Logistics Integrated Communications Environment (SPLICE) Functional Description, 1 May 1980.

4. Naval Supply Systems Command (Code 0415), Logistics Telecommunications Branch, SPLICE Telecommunications Subsystem Project Plan (TSPP), 9 March 1981.

5. Kroenke, D. Database Processing Fundamentals, Modeling, Applications. Science Research Associates, Inc., 1977.

6. Martin, J. Design and Strategies for Distributed Data Processing. Prentice-Hall, Inc., 1981.

7. Ullman, J.D. Principles of Database Systems. Computer Science Press, Inc., 1980.

8. Kroenke, D. Database, A Professional's Primer. Science Research Associates, Inc., 1978.

9. Appleton, D.S. Implementing Data Management, IEEE National Computer Conference, 1980, p. 307-316.

10. McLeod, D. and Heimbigner, D. A Federated Architecture for Data Base Systems, IEEE National Computer Conference, 1980, p. 283-289.

11. Sincodkie, W.D. and Farber, D.J. "SODS/OS: A Distributed Operating System for IBM Series/1," Association for Computing Machinery, Vol. 13, No. 3, p. 46-54, July 1980.

12. Mayer, P.S. Alternative Architectures for Distributed Data Sharing: Functional Issues, IEEE 1980 Computer Conference, Fall, p. 371-377, 1980.

13. Lunn, K. and Bennett, K.H. "An Algorithm for Resource Location in a Loosely Linked Distributed Computer System," Association for Computing Machinery, Operating System Review, Vol. 15, No. 2, p. 16-20, April 1981.

14. Nessett, D.M. "Identifier Protection in a Distributed Operating System," Association for Computing Machinery, Operating System Review, Vol. 16, No. 1, p. 26-31, January 1982.

15. Tanenbaum, A.S. "Network Protocols," Association for Computing Machinery, Computing Surveys, Vol. 13, No. 4, p. 453-489, December 1981.

16. Association for Computing Machinery, Vol. 15, No. 3, Report on the Fundamental Issues in Distributed Computing, July 1981.

17. Mooney, J.D. "USIM: A User Interface Manager," Association for Computing Machinery, Operating System Review, Vol. 16, No. 1, January 1982.

18. Kohler, W.H. Overview of Synchronization and Recovery Problems in Distributed Databases, IEEE 1980 Computer Conference, Fall, p. 433-441, 1980.

19. Garcia-Molina, H. Reliability Issues for Completely Replicated Distributed Databases, IEEE 1980 Computer Conference, Fall, p. 442-449, 1980.

20. Colliat, G. GCOS 8: A Distributed Processing Model, IEEE 1980 Computer Conference, Fall, p. 494-504, 1980.

21. Gray, J. Jr. Notes on Database Operating Systems, IBM Research Laboratory, San Jose, California, Summer 1977.

22. Lorin, H. Aspects of Distributed Computer Systems. John Wiley & Sons, Inc., 1980.

23. Lantz, K.A. and Rashid, R.F. A Virtual Terminal Management System for RIG, Computer Science Department, University of Rochester, New York, May 1979.

24. Martin, J. Computer Networks and Distributed Processing. Prentice-Hall, 1981.

25. Davies, D.W., Barber, D.L.W., Price, W.L., and Solomonides. Computer Networks and Their Protocols. John Wiley and Sons, Inc., 1980.

26. Hillsberg, B.L. "Generic Terminal Support," Association for Computing Machinery, Operating System Review, Vol. 15, No. 2, April 1981.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 54 Department of Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	Prof. Norman F. Schneidewind, Code 54SS Department of Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
6.	1st Lt. Joseph N. Reinhart III, USMC Marine Corps Development and Education Command Quantico, Virginia 22134	3
7.	Lt Ricardo Arana Courrejolles Ministerio De Marina Central De Procesamiento De Datos De La Marina De Guerra AV. Salaverry S/N Lima, Peru	3
8.	Lt Eduardo Bresani Torres Ministerio De Marina Central De Procesamiento De Datos De La Marina De Guerra AV. Salaverry S/N Lima, Peru	1

9. Lt Javier De La Cuba — 1
Ministerio De Marina
Direccion De Abastecimiento Naval
AV. Salaverry S/N
Lima, Peru

10. Lt. Jan Adams, USN — 1
SMC #1942
Naval Postgraduate School
Monterey, California 93940

11. LCDR Jerry Barnes, USN — 1
SMC #2542
Naval Postgraduate School
Monterey, California 93940

12. LCDR Kathleen Barrett, USN — 1
SMC #1087
Naval Postgraduate School
Monterey, California 93940

13. Lt. Sharon Crowder, USN — 1
SMC #2518
Naval Postgraduate School
Monterey, California 93940

14. Capt. Craig Opel, USMC — 1
SMC #1322
Naval Postgraduate School
Monterey, California 93940

15. Ing. Oscar Brain Canepa — 1
Burroughs Del Peru
Au Republica De Chile 284
Lima, Peru

16. Capt. Peter L. Jones, USMC — 1
Marine Corps Central Design and
Programming Activity
Marine Corps Development and Education
Command
Quantico, Virginia 22134

17. Major John Mastrocostopoulos — 1
DDB/GES
BST 902
Athens, Greece

18. Lt Col Joseph F. Mullane Jr., Code 0309 — 1
Marine Corps Representative
Naval Postgraduate School
Monterey, California 93940

19. Prof. Norm Lyons, Code 54Lb — 1
Department of Administrative Sciences
Naval Postgraduate School
Monterey, California 93940

Thesis
R32984 Reinhart
c.1
198917
Database and terminal management functional design specifications in support of stock point logistics integrated communication environment (SPLICE).