DATA EXCHANGE BETWEEN FULLY DISTRIBUTED HETEROGENEOUS MILITARY SUPER NETWORKS

by

THOMAS RUSSELL KELLY

A.B., Wofford College, 1964

A MASTER'S REPORT

submitted in partial fulfillment of the

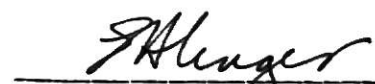
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas
1980

Approved by:



Elizabeth A. Unger

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

No effort of this nature can ever be accomplished totally by one person. Yet, to list all of the people who assisted me in this endeavor, I fear I would have to add twenty more pages. For this reason, I will limit my comments to those individuals whose efforts were most helpful. First, I would like to acknowledge the assistance, encouragement, patience, and understanding of Linda, my life partner, best friend, and wife. Without her support, I could not have finished this report. Second, I wish to express my deepest thanks to Dr. Elizabeth A. Unger, my major professor, who is not only my mentor, but also my friend. Third, to the other members of the Leavenworth Seven Plus Two, I can only say that it's been both a pleasure and a real experience. Fourth, I wish to thank Jim Ratliff and all the others who keep the Computer Science Department's computer system going. Their support eased the typing burden immensely. Finally, to all the others too numerous to mention, I wish to express my heartfelt thanks.

CHAPTER 1

INTRODUCTION

1.1 Overview

Advances in computer hardware, software, and communications technologies over the past decade coupled with users' requirements have fostered a milieu in which two seemingly contrasting fields, i.e., computer networking (decentralization) and data base management (integration), have been melded. Distributed data base management, the synthesis of these conceptual antinomies, is today one of the fastest growing areas of interest in computer science. In the past five years, significant progress has been made toward achieving the goal of a completely generalized distributed data base management system (here after referred to as DDBMS). Yet, despite considerable efforts in both the academic and the business/industrial communities, the realization of a generalized DDBMS is still several years away. The reasons, although varied, revolve around the fact that a number of key technical issues still are unresolved. This report focuses on one of these issues--data exchange.

Data exchange, of course, is not a new problem. Computer facilites have been exchanging data for many years using standardized transaction formats and external media (cards, tapes, etc.). As anyone who has ever been involved in this method of data exchange can attest, it is both a time and resource consuming process that requires much human intervention. The advent of networks allowed users via communication linkage to query files/data bases resident on other systems within the network, provided the user had knowledge of the distant facility's file management, data management, or data base management system. The problems inherent in this approach are obvious; however, they were manageable provided the network consisted of only a few facilities at which all data was centralized. Distributed processing and more particularly DDBMSs, which promise among other things data distribution invisibility, clearly make the old data exchange methods obsolete. Within fully distributed networks, data exchange may involve data conversion or translation that is both automatic and transparent to the user. This report analyzes the four automatic approaches (methodologies) suggested by Maryanski [100] as solutions to the problems of data conversion/translation in a completely generalized

distributed data base system environment. The methodologies are:

(1) the "brute force" approach, i.e., a unique translator for each pair of data base systems that will interface;

(2) the University of Michigan's Data Translation Project approach, i.e., the Stored Data Definition language (SDDL);

(3) the Data Specification and Conversion Language (DSCL) method proposed by G. M. Schneider; and

(4) the Advanced Research Project Agency (ARPA) Network Datacomputer Datalanguage approach.

To make the discussion more meaningful, these approaches are examined and evaluated within the context of a projected situation--the juncture of two distinct, independently developed fully distributed heterogeneous networks of networks (super networks). Although super networks per se do not exist, it is probable that they will exist in the military environment by the mid-to-late 1980's, if current trends persist. Therefore, the military environment and more particularly the military intelligence

environment provides the overall backdrop for the discussion.

1.2 Objectives.

Within the Department of Defense (DOD), the requirement to exchange automated data among and within the services has long been recognized. Networking as a method to satisfy the requirement is also well known. Within the past decade, for example, both homogeneous and loosely federated heterogeneous networks composed of geographically distributed central computer facilities have been established within the DOD to facilitate data exchange. What is apparently not equally well-known within DOD are the benefits that can accrue from the use of generalized DDBMSs and the data exchange problems that are created when separate, fully distributed heterogeneous super networks are allowed to developed in an unstructured manner. The objectives of this report are to provide: (1) managers and system designers with a common frame of reference not only for data exchange problems, but also for many of the other issues associated with interfacing fully distributed heterogeneous super networks; (2) managers and users

with easily assimilable information that can be incorporated readily into their requirements documents; and (3) systems designers with directions toward the most feasible, reliable, cost effectve, and efficient solution to the problem of interfacing super networks. Although this report focuses on the problem of data exchange between super networks in a military environment, the discussion is equally applicable to managers and system designers outside the military environment who are faced with the task of implementing either a distributed data base system or interfacing independently developed fully distributed heterogeneous networks or super networks.

1.3 Assumptions.

1. Since one of the objectives of this report is to provide easily assimilable information, "lay" language will be used throughout the report to the maximum extent possible. Where either computer science technical or common usage terms are essential, they will be explained in a manner that is understandable to mid-to-high level managers, who have a user level knowledge of computer systems, data base management

systems (DBMSs), and networks. (Users who are not familiar with network technology should consult chapters 4 through 7 of An Introduction to Distributed Data Processing by Harry Katzan, Jr. and "Computer Interconnection Structures: Taxonomy, Characteristics, and Examples" by George A. Anderson and E. Douglas Jensen, and the March 1976 special DBMS issue of the ACM (Association for Computing Machinery) Computing Surveys, edited by E. H. Sibley.

2. While this report focuses on the problem of data exchange between fully distributed heterogeneous super networks in a military environment, it is not intended as an advocation of either a network or DDBMS philosophy to the exclusion of others. Instead, it is an acknowledgement of what appears to be a clearly defined trend within the DOD and more particularly the DOD intelligence community.

3. The super networks that are described in this report are notional and therefore completely system independent, i.e., no specific hardware, software, or firmware configuration is assumed or intended.

4. The technical scope of this report is not detailed enough for implementation. The rationale for this approach is twofold: (a) the target audience is not

concerned with the details of implementation; (b) super networks currently do not exist.

5. Other aspects of super networek interface, e.g., communications paths, modems, multi-level security classification problems, communications security, data security, and data independence are discussed only as necessary to support the primary focus--data exchange.

1.4 Organization.

Six chapters comprise this report. Chapter 2 is designed to provide the reader with a basic understanding of the concept of distributed processing. Chapter 3 (Distributed Data Base Management) presents an overview of the key aspects of DDBMS technology. Specifically, it addresses three major topics--concepts and developments in DDBMS technology, distributed data base design considerations, and the key technical problems that must be overcome before a generalized DDBMS can be realized. For those readers unfamiliar with the Military Establishment, the fourth chapter is devoted to the military environment. Specifically, Chapter 4 discusses post-Vietnam developments, intelligence

shortfalls, current automated intelligence systems and development efforts, Department of Defense and North Atlantic Treaty Organization military structures, and projected automation developments (the notional super networks). Attention is focused in Chapter 5 on the four automatic data translation methodologies that were mentioned earlier. Each approach is described and briefly analyzed. The approaches are then evaluated to determine their applicability/suitability to the fully distributed heterogeneous super network environment. In addition, the final chapter summarizes significant findings and presents the author's recommendations.

CHAPTER 2

DISTRIBUTED PROCESSING

2.1 Introduction.

For the past several years, a number of organizations have been moving away from large centralized computer system facilities and toward multiple numbers of smaller dispersed sites [30,53]. This trend has been variously termed distributed processing [30,43], distributed data processing [46,75,167], and distributed information systems/networks [22,23]. This chapter seeks to provide the reader with a basic understanding of this trend to facilitate later discussion (Chapter 3) of distributed data base management systems (DDBMSs).

The recent spate of articles in widely circulated data processing magazines and other media attest to the growing, almost "faddish" popularity of distributed processing. These articles coupled with hardware and software vendors' advertisements proclaiming that distributed processing has come of age present readers with a perplexing dilemma. If these claims are valid, then distributed processing should be a well-defined field. Yet, more than a casual perusal of

the popular literature reveals that there are almost as many definitions of distributed processing (stated or implied) as there are authors who have written about it. Commenting on this situation, some authors probably shaking their heads in dismay have concluded that distributed processing is a "...concept in search of a definition." [75] The question that begs answers is why does this obvious dichotomy exist within the data processing community? The answers (reasons) are both symptomatic and systemic. First, distributed processing is still a relatively new field. Unlike the more traditional centralized approach, distributed processing is not yet based on a solid foundation of fully developed, generally accepted tenets. Second, researchers unintentionally have contributed to the basic misunderstanding by (a) addressing distributed processing only as a number of definable problem areas, and (b) focusing on the same problem areas from differing, often unstated or ill-stated perspectives. Third, the essential nature of distributed processing is often misunderstood. Distributed processing is not a monolithic concept. Therefore, instead of being one dimensional (centralization verses decentralization) as most popular authors apparently have assumed, distributed processing is really multifaceted and multidimensional. To confound understanding further, each dimension can range over a spectrum of values from

centralized to partially distributed to fully distributed and shades in between somewhat independently of the other dimensions. Fourth, there have been few attempts to view the entire concept from top-down. As distributed processing matures, more attention probably will be focused on codification of key principles and clarification of the ever burgeoning, often conflicting, and sometimes meaningless plethora of terms. Until this occurs, however, it is unlikely that a consensus will emerge on the delineation between nondistributed and distributed systems. To promote understanding of distributed processing, a working definition will be constructed in this chapter using three approaches. The first approach looks at what distributed processing allows users to do. The second approach analyzes the advantages claimed for distributed processing. The third approach is a critical appraisal of the views of four authors.

2.2 User's Requirements.

Kirkley [79] suggests that instead of trying to determine what distributed processing is, it is more productive to define what it allows the users to do. Becker partially answers the question by stating:

> Many users today view the new "distributed" concepts as a means of fitting computer resources (solutions) to their organizational needs (problems) as opposed to the previously customary fitting of the problem to the solution. [11, p. 86]

Liebowitz and Carson further elaborate as follows:

> To some, a distributed system is a collection of multiple computers or processing elements working closely together in the solution of a single problem....
>
> To other users, a distributed system is a set of intelligent terminals located at the point of use to give local organizational elements more responsive computer support....
>
> To yet another set of users, distributed systems may mean collections of geographically dispersed, independent computer centers linked together to allow sharing of software and hardware resources....
>
> To some, distributed processing means the use of small computers to off-load supporting functions from large mainframes, to extend the life of the mainframe....[82, p. 1-1]

These quotes not only mirror some of the current differing (sometimes erroneous) views of distributed processing, but they also suggest two important characteristcs of distributed processing. First, distributed processing represents more than the sum of the rather spectacular advances made in computer hardware (price-performance

revolution in microelectronics), communications (networking), and data base management technologies over the past decade. It also represents users' requirements. In fact, most authors agree that the underlying impetus for distributed processing came from users rather than hardware and software vendors. Second, any meaningful definition of distributed processing must encompass a wide range of organizational structures and more particularly management views of organizational information dynamics. [75,167]

2.3 Distributed Processing Advantages.

Another approach to the task of defining distributed processing is to examine its claimed advantages. Since many of these are meaningful only when juxtaposed with a known quantity, the advantages and disadvantages of centralized systems will be discussed first. The generally recognized advantages of centralized systems are as follows:

1. operating cost economy of scale;
2. unified control;
3. easy intrafile communications;
4. easy file access for update and retrieval;
5. consistency of integrated data bases;

6. compatibility/standardization.

The price-performance revolution in microelectronics and the subsequent advent and widespread usage of minicomputers has depreciated considerably the centralized system's long-tauted economic advantage--hardware economy of scale. Nonetheless, a relatively strong case can still be made for overall operating cost economy of scale, especially when the total requirements for centralized systems are compared to those of distributed systems, e.g., larger skilled data processing staff, maintenance of multiple hardware and software systems, etc.

The generally recognized disadvantages of centralized systems include:

1. vulnerability to failure both in the central site hardware and the communications system, if appropriate;

2. lack of configuration flexibility/modularity;

3. inability to respond quickly to changing user needs or technology (upgrades and vendor changes to include hardware and software are expensive, require detailed planning, consume resources that normally would be applied to the satisfaction of user needs, and degrade overall system performance during the implementation process);

4. communication system costs;

5. tendency toward exponential increases in overall system complexity from simple arithmetic increases in numbers and/or types of users and uses.

Disadvantages 2, 3, and 5 above may be partially offset by adding more processors to the central system, if feasible, or by "...off-loading some of the processing functions to ancillary machine facilities, i.e., performing some of the initial processing of data at the site where that data is collected, or else enhancing the central site with functionally dedicated machines." [51, p. 2]

Champine [30] asserts that the advantages and disadvantages of distributed processing systems mirror their centralized system opposites. Conceptually, his assertion is correct, but only if system design dynamics are ignored and both approaches (centralized and distributed) are assumed to be monolithic. Because distributed processing is a multidimensional concept, Champine's assertion is only partially valid. Most listings of claimed distributed processing advantages and disadvantages are based on the same implicit assumption. Therefore, all listings of advantages and disadvantages should be regarded as relative rather than absolute values, and the validity and/or

applicability of each advantage and disadvantage should be judged on the basis of individual system design goals and objectives. With this in mind, the claimed distributed processing advantages now can be considered. In general, four major advantages are claimed for distributed processing. These are extensibility, integrity, performance, and responsiveness.

Sometimes referred to as adaptability, expandability, flexibility, or modularity, extensibility "...is the degree to which system functionality and performance can be changed without changing the system design." [74, p. 29] Extensibility has two major aspects--ease of growth and ease of modification. In the growth sense, extensibility refers to the ability of a system to absorb easily incremental upgrades either in system performance or functionality at minimum costs. Extensibility in the modification sense is the ability of a system to cope with replacement of either hardware or logical function simplistically. Since centralized systems and computer networks generally exhibit only limited degrees of extensibility, distributed processing systems, if designed with extensibility as a goal, can offer substantial improvements over conventional configurations. [74]

Integrity is the measure of a system's ability to tolerate faults, errors, and failures. Integrity, which encompasses such attributes as reliability, failsoft, fault-tolerance, involves more than just diagnosis, detection, and reporting. It also includes recovery, repair, and restart. Integrity in conventional systems (centralized and network) usually is applied in a piecemeal manner at the processor level rather than as a total system approach. For example, numerous techniques have been developed for improving the integrity of uniprocessor systems. These can also be applied to processors in a multiple processor system, but only if resources such as disks are not shared. [74] In loosely federated networks or computer-communications networks [45] where the user views the network as a collection of several systems offering varying services rather than as one large system, integrity is also approached in a piecemeal manner. Each host (uniprocessor or multiprocessor) system is responsible for its own internal integrity, while the communications system is required to deal with data communications integrity (Despite some successes in the data communications area, much still remains to be done). So long as network systems remain loosely connected, the piecemeal approach to integrity is probably sufficient. The promise of

distributed processing systems is that integrity will not only be applied to the various components of the system, but also will be a prime consideration in the design of the total system structure. [74] Another aspect of integrity, which is extremely important to the tactical military environment, is system resiliency, i.e., the ability of a system to deal with abrupt loss or gain of system components or nodes. [128]

The third major advantage of distributed processing is improved system performance both in terms of faster response and higher throughput. In both uniprocessor and multiprocessor systems, there are known bounds beyond which performance in terms of response and throughput cannot be improved economically. The well-known exponential rise in software cost as maximum processor performance is approached is a classic example. Adding processors to a uniprocessor system can decrease response time; however, a price beyond the purely economic consideration of buying/leasing the additional processor(s) must be paid. This price is usually throughput degradation and/or processor inefficiency. Both response time and throughput can be improved, if a multiplicity of processors are executing concurrently. However, even greater improvements are potentially available, if multiple processors can cooperate on a single

task. Again, this is one of the promises of distributed processing. Derivatives of the overall performance advantage include resource sharing, automatic load sharing, and good response to peak loads. [74]

Responsiveness, the fourth major advantage, is a measure of human resource satisfaction. From the local management viewpoint, the fact that a computer system is situated and managed locally "...can more directly satisfy the needs of local organizational management than one located in a central facility...." [75, p. 19] Wu compared this aspect "...to the concept of 'responsibility accounting', which recognizes various decision centers within an organization and attempts to provide information relevant to each individual decision center." [167, p. 3] Distributed processing systems also are potentially more responsive to users' problems and requirements than are centralized systems. While this may be more a matter of perspective than anything else, it is still a valid consideration. One of the complaints that both users and management voice at centralized systems is that they are so tightly controlled that they often stifle initiative. A quantitative spinoff of this rather qualitative advantage is the reduced communication cost associated with distributed processing. If the majority of data needed locally is

available locally, then requests for data beyond normal requirements can be handled almost on an exception basis. This, in turn, can result in a relative reduction in the overall demand placed upon the system's communication network, which translates directly into lower communications costs.

The advantages claimed for distributed processing are impressive. Separately, they tell us much about the characteristics that users collectively deem important. But what do these advantages tell us about the overall nature of distributed processing? Enslow's answer is:

> Hardly anyone reasonably knowledgeable about the current state of the art in multiple-processor data processing systems would claim that major advances toward any significant number of these goals are possible with present technology. To fulfill such a claim, distributed data processing must be something new....[46, p. 13]

Enslow's statement is partially correct, because few of the claimed advantages (goals) are fully attainable with present technology. However, this fact alone does not support his assertion that distributed processing "...must be something new...." Like Champine, Enslow clearly falls into the trap of viewing distributed processing as a monolithic concept. This contention is supported by his later statement that "...only the combination of all of the criteria uniquely

defines distributed data processing." [46, p. 14] From Enslow's perspective, each claimed advantage must be considered part of an elaborate requirement set rather than a series of system design goals and/or tradeoffs. Enslow's view is clearly unacceptable. A more realistic appraisal is offered in part by Becker, who states:

> While called a revolution by many..., the concepts...appear as a very logical evolution of existing known technologies. [11, p. 86]

Thus far, only the claimed advantages of distributed processing have been discussed. These are indeed imposing. However, as Fry and Sibley have so aptly stated, "...distributed systems pose many new problems and exacerbate many old ones." [58, p. 33] Since discussion of the problems and disadvantages is not germane to the task of defining distributed processing and the problems and disadvantages apply equally, if not to a greater extent, to DDBMSs; their description will be deferred until Chapter 3.

2.4 Authors' Views

The first two approaches to constructing a working definition of distributed processing have provided an outline. The third approach analyzes the stated or implied

definitions offered by four authors to discern areas of commonality and disagreement. Before the various definitions can be considered, three stipulations must be imposed. First, the authors selected have disparate backgrounds and interests. Second, their opinions do not represent either a random sample or a complete survey of the entire spectrum. Third, the order in which they are discussed is not necessarily significant.

2.4.1 Katzan.

In his recent book, An Introduction to Distributed Data Processing [75], Harry Katzan, Jr. offers the following definition:

> A distributed system is one in which computers - usually minicomputers - are located in sites that are remote from a central computing facility and those computers possess the physical attributes that permit them to interact with the central facility and possibly with each other. The objective of distributed computing is to process information where it is most effective to satisfy operational, economic, or geographic conditions. [75, p. 4]

Katzan tempers his definition by describing what he considers to be the essential characteristic of distributed systems. These include:

1. local computers, which possess the characteristics of small stand-alone/self-contained processing systems;

2. local management control of remote computers to include the responsibilites of performing traditional computer installation management tasks as well as a given set of applications;

3. local computers processing data and applications for their host facilities;

4. distributed systems storing data locally either as a set of files or a data base depending on local requirements;

5. telecommunications facilities to link local processors with their hosts.

In his discussion of these characteristics, Katzan reveals not only some of the rationale he uses to distinguish centralized from distributed systems, but also some of his opinions on the operations of a distributed system. For example, he views the control and processing of applications at remote facilities to be one of the key features of a distributed processing system. Storing data locally is another key feature; however, he seems to view a local data base only as a logical subset of a central data base. (The subset must be refreshed periodically.) With

regard to telecommunications facilities, Katzan states that they do not have to be of the type that would support on-line demand-response operations. He even admits that a strong case could be made for a distributed system using other means of communication rather than telecommunications; however, he notes that "...the concept loses most of its operational flavor without telecommunications." [75, p. 5]

In summary, Katzan's definition of distributed processing calls for four elements to be dispersed--computer resources, management control of these resources, applications processing, and data. While he emphasizes that the primary system level relationship of the central (large mainframe) and remote (mini) computers is of the master-slave variety, his basic definition is still broad enough to allow for five distinct classes (or partitions as he calls them) of system architecture-- "...horizontal, vertical, geographical, functional and combinatorial." [75, p. 123] A close look at his five partitions, however, reveals that there are really only two basic types. These are horizontal (all processors cooperate at an equal level) and vertical or hierarchical (higher level processors controlling lower level ones). See Figure 2-1 for Katzan's view of the evolution from centralized to distributed processing.

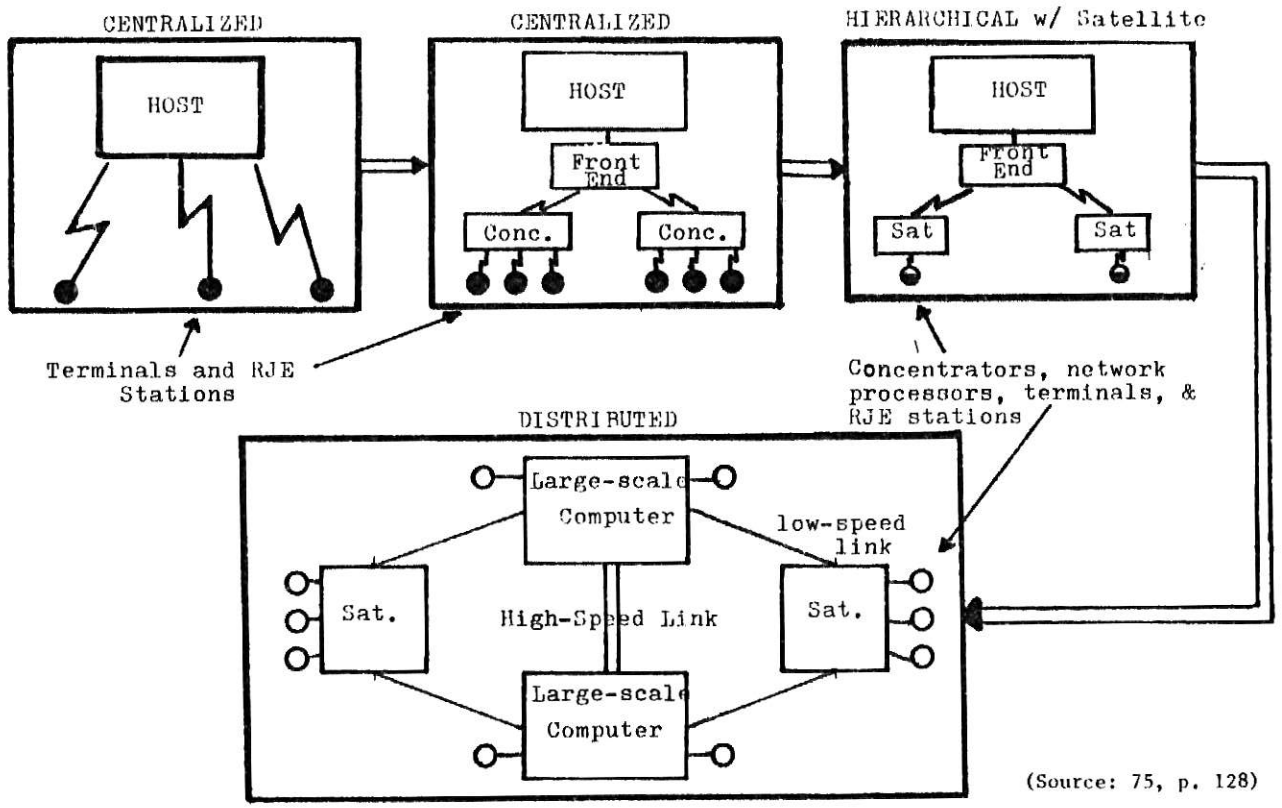


Figure 2-1: Representative pattern of evolution

2.4.2 Davenport.

A view somewhat similar to Katzan's is voiced by R. A. Davenport of the London School of Economics. Rather than define distributed processing, Davenport [40] uses three criteria to identify distributed processing systems:

> The first criterion is that the computing system possess two or more geographically dispersed processors. The processors must be application logic oriented. Applications programs or portions of programs are loaded and run on these processors.
>
> The second criterion is that the two or more dispersed processors be linked. This linkage is accomplished through telecommunications....
>
> The third criterion for distributed processing is that the network created by the two or more displaced, linked processors be confined within an organizational boundary. [40, p. 7]

Although not one of his criteria, Davenport assumes that distributed systems are primarily transaction oriented, i.e., they allow the user to interact directly with the system through either dumb or intelligent terminals. Unlike Katzan, Davenport identifies only two distinct distributed system structures--horizontal and hierarchical. The primary differences between the structures are the relationships of the processors in the network and the manner in which tasks are performed. As Davenport explains, "in the

hierarchically distributed system the processors share tasks in a structured way with each component to some degree controlled by the higher level members of the hierarchy," [while] "in the horizontally distributed system all processors cooperate at an equal level, logically, to perform a set of tasks." [40, pp. 7-8]

Like Katzan, Davenport's criteria clearly implies that four elements are distributed--computer resources, management control of these resources, applications processing, and data. Unlike Katzan's definition, Davenport's criteria and later discussion does not emphasize the hierarchical (master-slave) over the horizontal structure. Both, however, share a similar conceptual bias about processor types, i.e., minicomputers (sometimes called satellite processors) are always subordinate to larger systems, the data stored on minicomputers is almost always a subset of the central system's data base, and only large processors can cooperate as equals. Both authors also agree that examples of horizontal and hierarchical structures may be found in complex systems. Davenport's third criterion confines distributed processing to organizational boundaries. Katzan's definition does not impose such a restriction; however, his discussion clearly implies one. Even though Davenport's second criterion mandates

telecommunication linkage among dispersed processors, he does state that in the special case of a developing country that "...it would seem to be valid to have a distributed data base even when no data communications links exist, as long as there is centralized control of the data base." [40. p. 10]

There are minor differences in the views of Katzan and Davenport; however, the dissimilarites are overshadowed by the similarities. This is largely due to three factors. First, both authors have drawn heavily upon the earlier writings [22,23] of Grayce M. Booth. Second, both Katzan and Davenport are essentially reporting, albeit in a generalized manner, on the state of the art as reflected in both network and distributed processing system implementations to date--networking existing central processor configurations and using minicomputers as satellite processors to distribute resources, applications, data, and management of resources. Third and perhaps more importantly, Katzan and to a lesser extent Davenport are both viewing distributed processing from a management perspective.

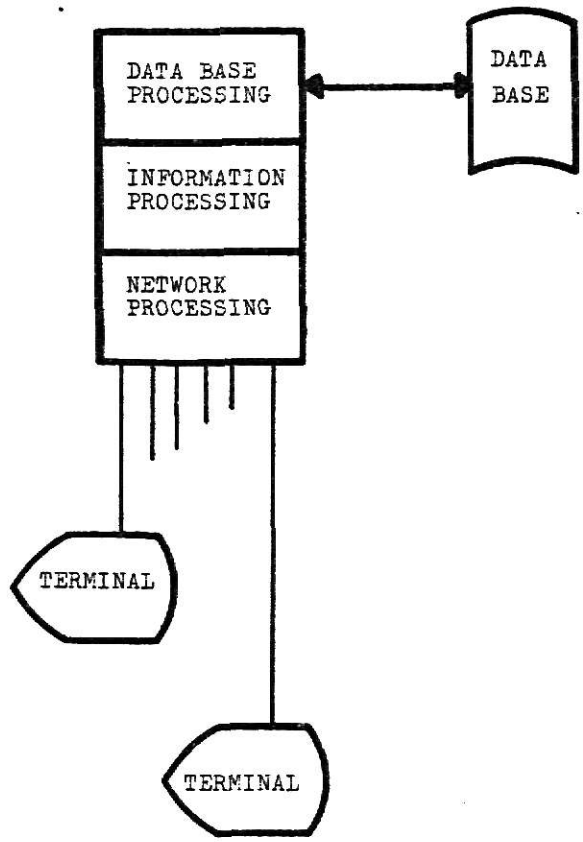
2.4.3 Becker.

The third author to be considered, Hal B. Becker, approaches distributed processing from a functional perspective. Becker [11] asserts that three functions constitute the basic building blocks of any system. These are information processing, network processing, and data base processing. His rather simplistic definition of these functions follow:

> Information processing can be defined as the manipulation (by applications programs) of information to produce the desired results. Network processing is the control of information movement between the various locations (nodes) of the network. Data base processing is the storage of quantities of information, in one or more forms, available to the network and its users. [11, p. 83]

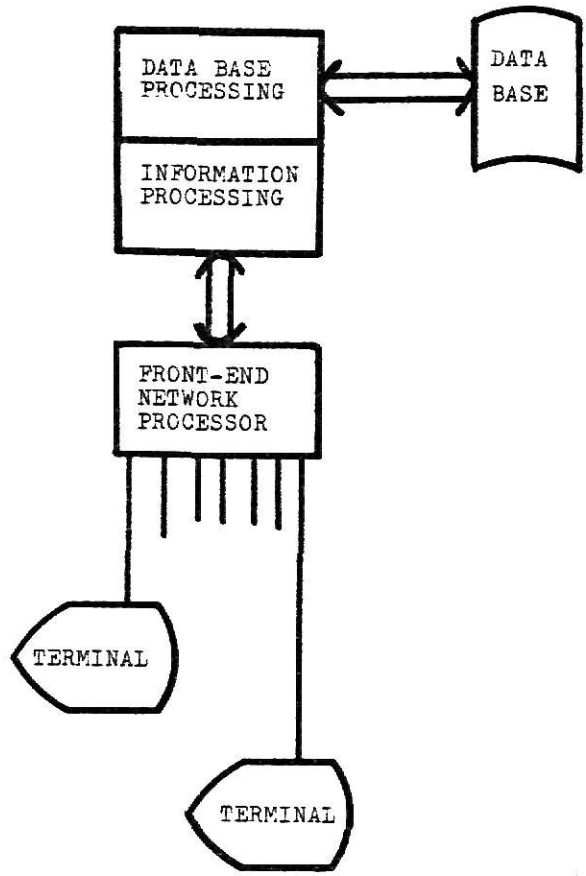
Becker's thesis is that totally distributed processing can only exist when three conditions are met. First, the three functions must be separated, i.e., the network and the data base functions must reside on separate front-end and back-end processors, which are connected to an information processor. Under this arrangement, "...each of the three...can cooperate or execute somewhat independently of the others with hardware and software optimized for its own

function." [11, p. 83] One of the problems that is solved by this functional separation, explains Becker, is the interference caused by competition for common or shared resources. The second condition is that at least two of the three kinds of separated functional processing (data Base and network processing) must be performed at more than one remote location. The final condition is that all sites must interact as co-equals. Although easily discussed; functional separation, dispersion, and peaceful coexistence among the various system processors are not easily implementable, if at all, with currently off-the-shelf hardware and software components. Becker cites three significant reasons for this situation--lack of a distributed operating system (OS), lack of DDBMSs, and lack of adequate higher level protocol language for controlling the various functional interactions. Figures 2-2 through 2-6 in sequence are representative of Becker's view of the evolution from centralized to distributed processing.



(Source: 11, p. 82)

Figure 2-2: Centralized System



(Source: 11, p. 82)

Figure 2-3: Centralized System

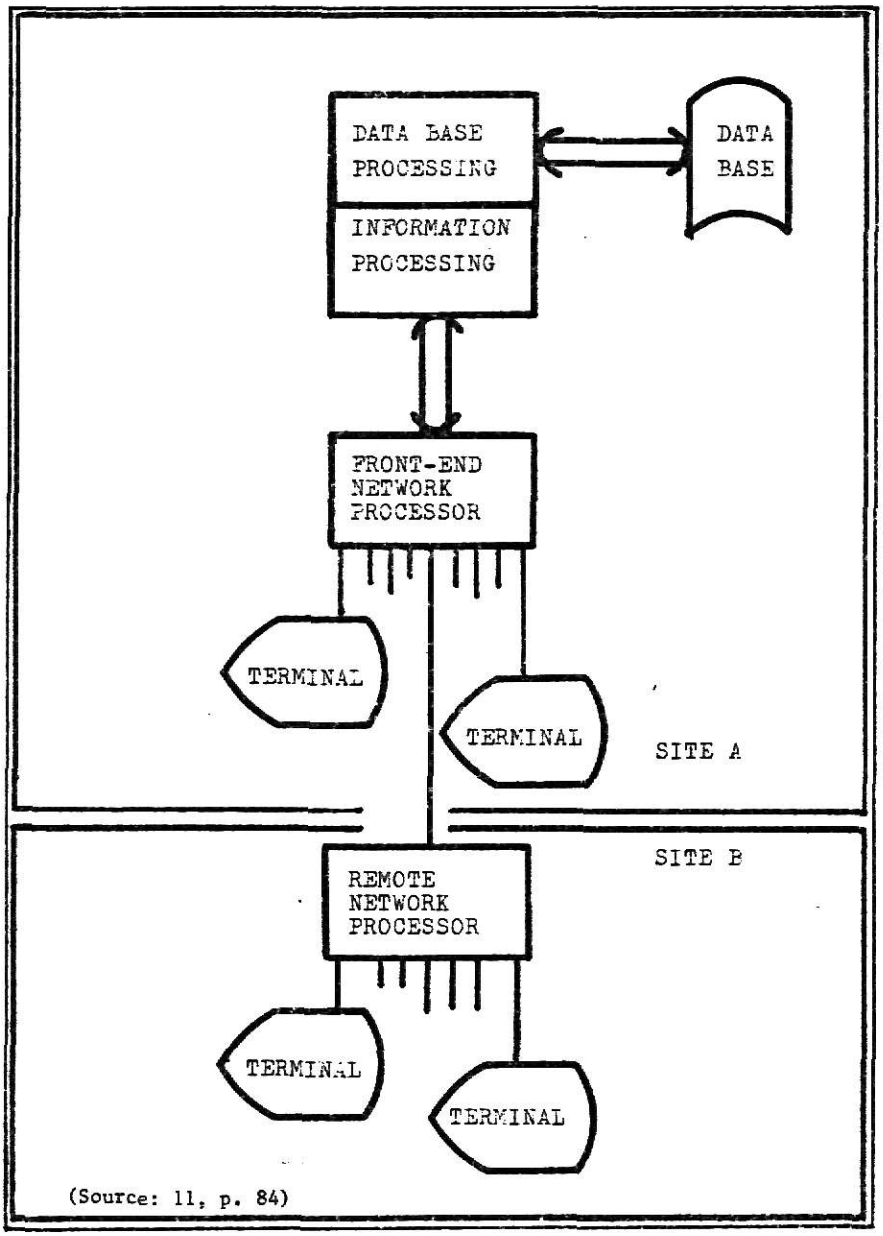


Figure 2-4: Partially Distributed system

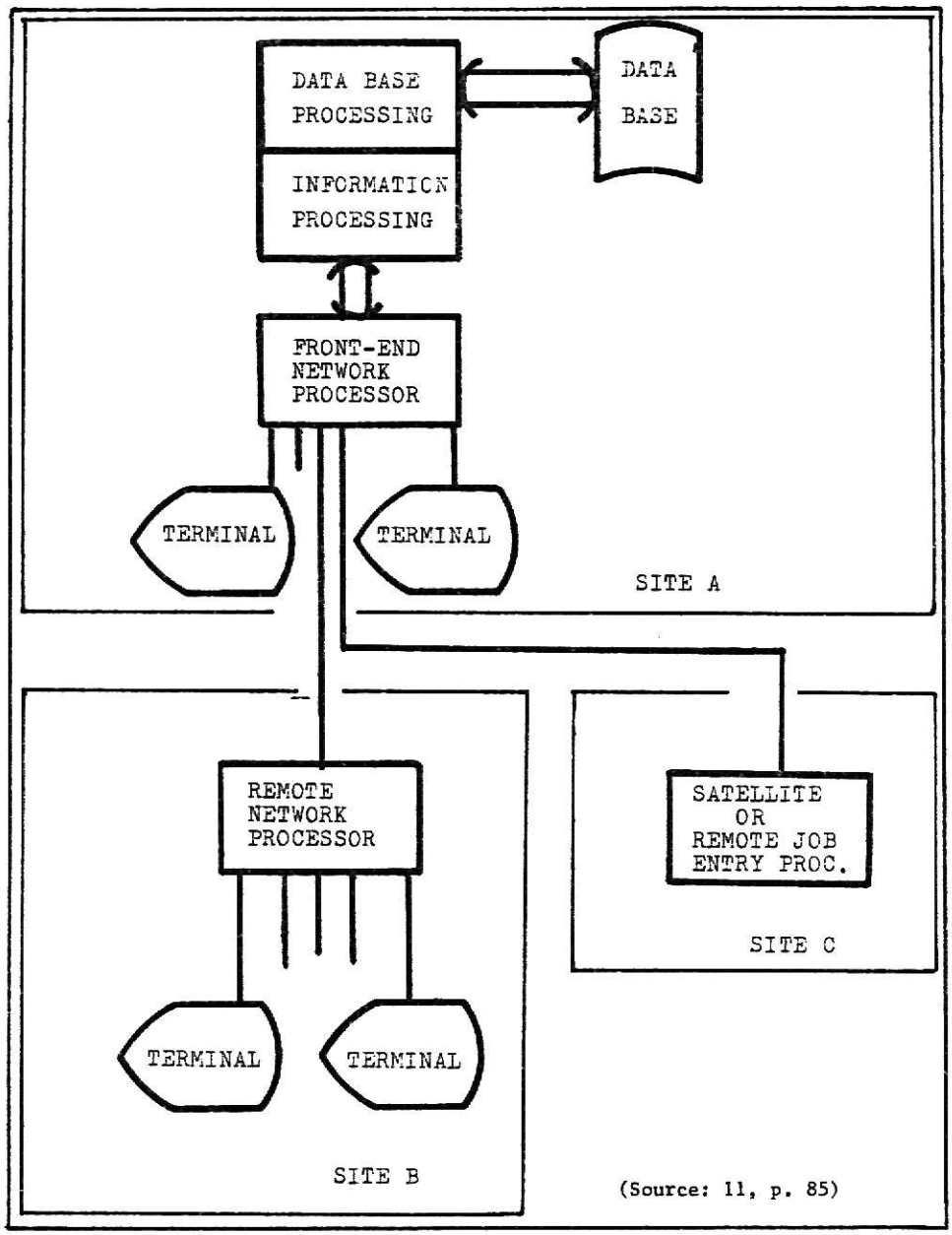
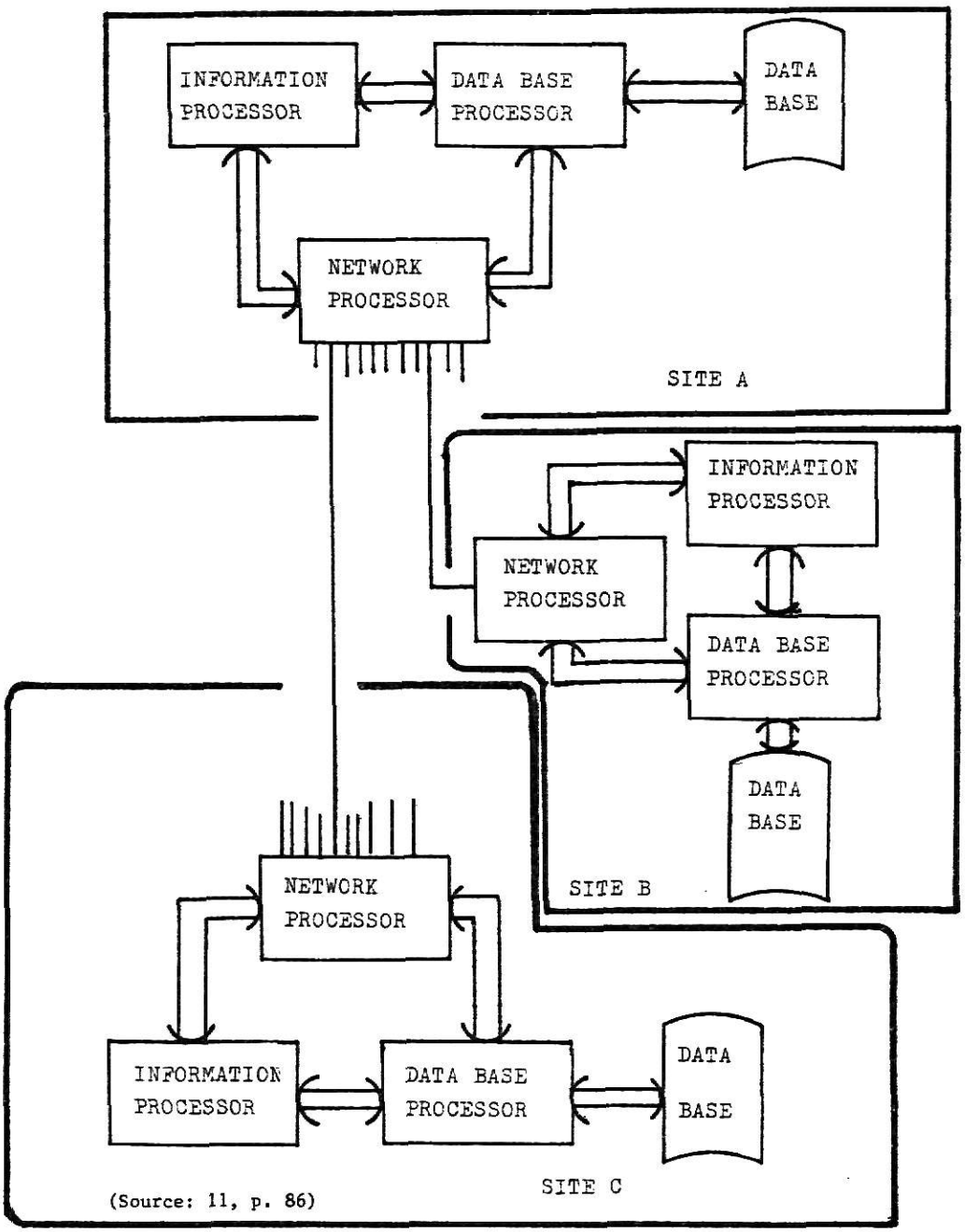


Figure 2-5: Partially Distributed System



(Source: 11, p. 86)

Figure 2-6: Fully Distributed system

The key points of Becker's definition of distributed processing are:

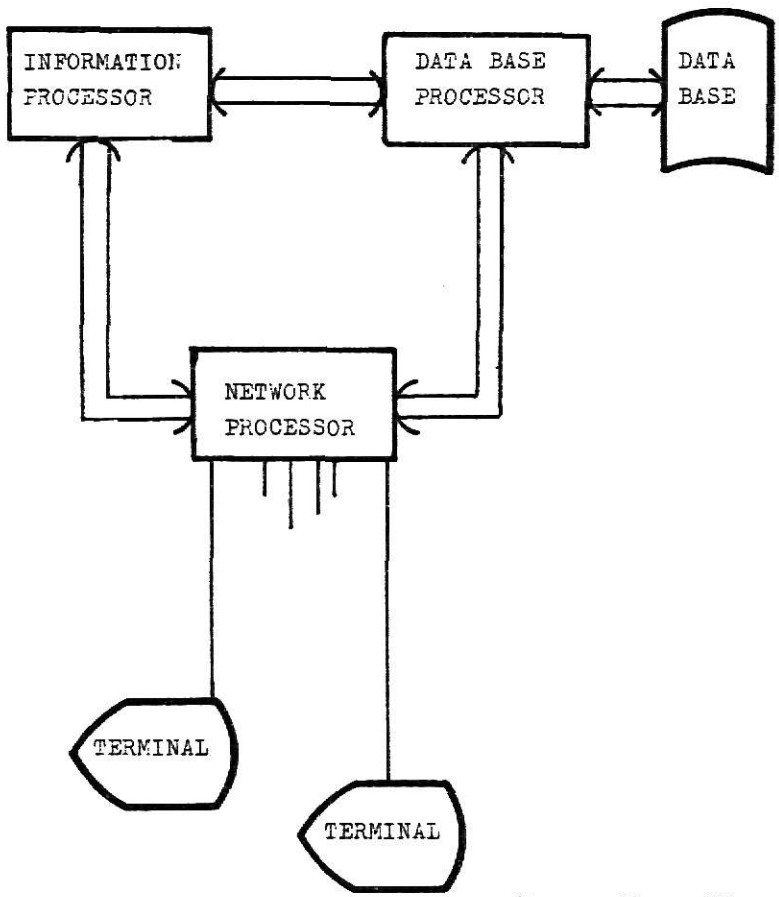
1. functional separation and dispersion of functional components (logical and physical);

2. network interconnection structrure;

3. totally distributed processing includes distributed data base management and assumes peaceful coexistence among the logical and physical components;

4. realization of totally distributed processing is still several years away due to lack of distributed OS and DDBMS and higher level protocol language.

As an aside, Becker's logic is not always sound. For example, he considers the special case of the three functions residing on separate, but connected processors at a central location (see Figure 2-7) to be centralized, "...because all of the logic (software) for all three functions executes at a single central location." [11, p. 83]



(Source: 11, p. 83)

Figure 2-7: Centralized System
(Functionally Distributed)

2.4.4 Enslow.

The last author to be considered is Philip H. Enslow, Jr. Although, as discussed earlier, his central thesis [46] that distributed processing is a new concept is much too restrictive, Enslow does some insights into the definitional problem. To Enslow, a proper definition of distributed processing must include not only the distribution of physical components, i.e., "...hardware or processing logic, data, the processing itself, and the control (such as the operating system)," [but also] "...the concepts under which the distributed components interact." [46, p. 13] His five part definition, which he considers to be a research and development definition, follows:

> * A multiplicity of general-purpose resource components, including both physical and logical resources, that can be assigned to specific tasks on a dynamic basis. Homogeneity of physical resources is not essential.
>
> * A physical distribution of these physical and logical components of the system interacting through a communications network. (A network uses a two-party cooperative protocol to control the transfers of information.)
>
> * A high-level operating system that unifies and integrates the control of the distributed components. Individual processors each have their own operating system, and these may be unique.
>
> * System transparency, permitting services to be

requested by name only. The server does not have to be identified.

* Cooperative autonomy, characterizing the operation and interaction of both physical and logical resources. [46, p. 14]

At first glance, several of the terms that Enslow uses to describe his view of distributed processing systems, e.g., high-level operating systems and cooperative autonomy, appear on the surface to be contradictory. As will be seen, this is due more to his imprecise usage of terminology rather than to any real conflict. The key feature of Enslow's concept is a high-level operating system (HOS). Because Enslow's view of a HOS differs markedly from the normal meaning associated with the term, operating system, it is perhaps more properly referred to as "...decentralized system-wide executive control." [74] To Enslow, a HOS is "...a well-defined set of policies...[that governs]...the integrated operation of the total system." [86, p.15] Unlike its large, complicated, and sometimes unwieldy namesake, a HOS may exist either as a design philosophy or as distinct blocks of code within each processor's unique operating system. Enslow attributes the following characteristics to a HOS:

1. the relationship between HOS and the operating systems of each processor cannot be hierarchical, because

this would violate the autonomous operations criteria.

2. HOS must support a larger transfer bandwidith (the amount of information that can be transferred a at a given time) between processors than that normally associated with loosely federated computer networks;

3. the environment fostered by HOS must be flexible, i.e., it must

a. allow individual processors to operate effectively when disconnected from the system,

b. support/facilitate reintegration of disconnected processors into the network without effecting in-progress operations,

c. minimize permanent bindings of loci of control or processing activities,

d. not allow creation of critical paths or components such as the physical binding of any resource to a particular component,

e. provide for processor and process interactions at a number of levels,

f. work with incomplete or even erroneous status information due to the time lags created by the autonomous operation of the various physical components.

In other words, a HOS must attain the status of system demi-god, for as Enslow notes,

> Even if autonomous multiple components are cooperating, the probability of simultaneous conflicting actions is much higher than in hierarchical systems. Also synchronizing the actions of the various controllers in the system is much more difficult, because of the presence of appreciable time-lags. Finally, the problem of deadlocks or infinite cycles withing the system is quite different from that associated with other systems. [46, p. 20]

Another important characteristic of Enslow's definition is "...a multiplicity of assignable resources...." [46, p. 14] While he uses the modifier, "general-purpose," to describe these resources in his definition, he is not really concerned with each component's actual purpose (be it general or special) as he is with the system's "...ability to be dynamically reconfigured on a short term basis with respect to those resources that provide specific services at any given time." [46, p. 14] Thus, a totally distributed system such as the type defined by Becker would still meet Enslow's requirements. This is due to the fact that while an individual resource might be considered special purpose, e.g., a front-end, the node of which it is a part would be considered general purpose from the total system perspective.

A third important characteristic of Enslow's definition is physical distribution and interconnection. While aspects of and approaches to network definition are myriad, Enslow concentrates on only one aspect--the transfer of messages. The physical transfer of messages, claims Enslow, is a prime example of cooperation between physical resources, because "such transfers follow a two-party protocol, in which the two parties must cooperate to successfully complete the transfer." [46, p. 14] (The two-party protocol allows resources either to accept or to refuse transfer of information based on that resource's knowledge of its status.) This type of transfer is much different from the gated transfer methodology normally employed in hierarchical or centralized systems "...where the master has full authority to force a slave to physically accept a message." [46, p. 14] Enslow further contends that the cooperative network interconnection concept must apply not only to physical, but also to logical resources, especially "...with respect to transfers of information such as status, requests for services, synchronization between logical resources and so on." [46, p. 14] So complete is his belief that a two-party protocol is the key factor in network interconnection that he maintains that it should apply to the transfer of information between all resources even to the interaction of two resources on a single integrated

circuit chip. From this perspective, Enslow's "...definition of a network establishes no criteria for the length of the interconnection paths of the network." [46, p. 14]

System transparency and cooperative autonomy, the other two characteristics of Enslow's definition are self-explanatory; therefore, they will not be discussed. The system that Enslow describes obviously is not available today. In fact, it may never be. However, since his description represents a possible evolutionary alternative, it cannot be ignored. By the same token, his contention that distributed processing "...must be something new..." along with the clear implication that his represents the only valid definition is certainly not proved. Despite the flaws in his main argument, Enslow does make several contributions to an overall understanding of what does and does not constitute distributed processing. One of the most signifcicant of these is his discussion of the three dimensions (decentralization in hardware, control point, and data base organization) that characterize distributed processing from a system implementation perspective. Rather than recount the details of his discussion of this aspect, the chart he uses to summarize his findings (with some minor modifications) is presented in Figure 2-8.

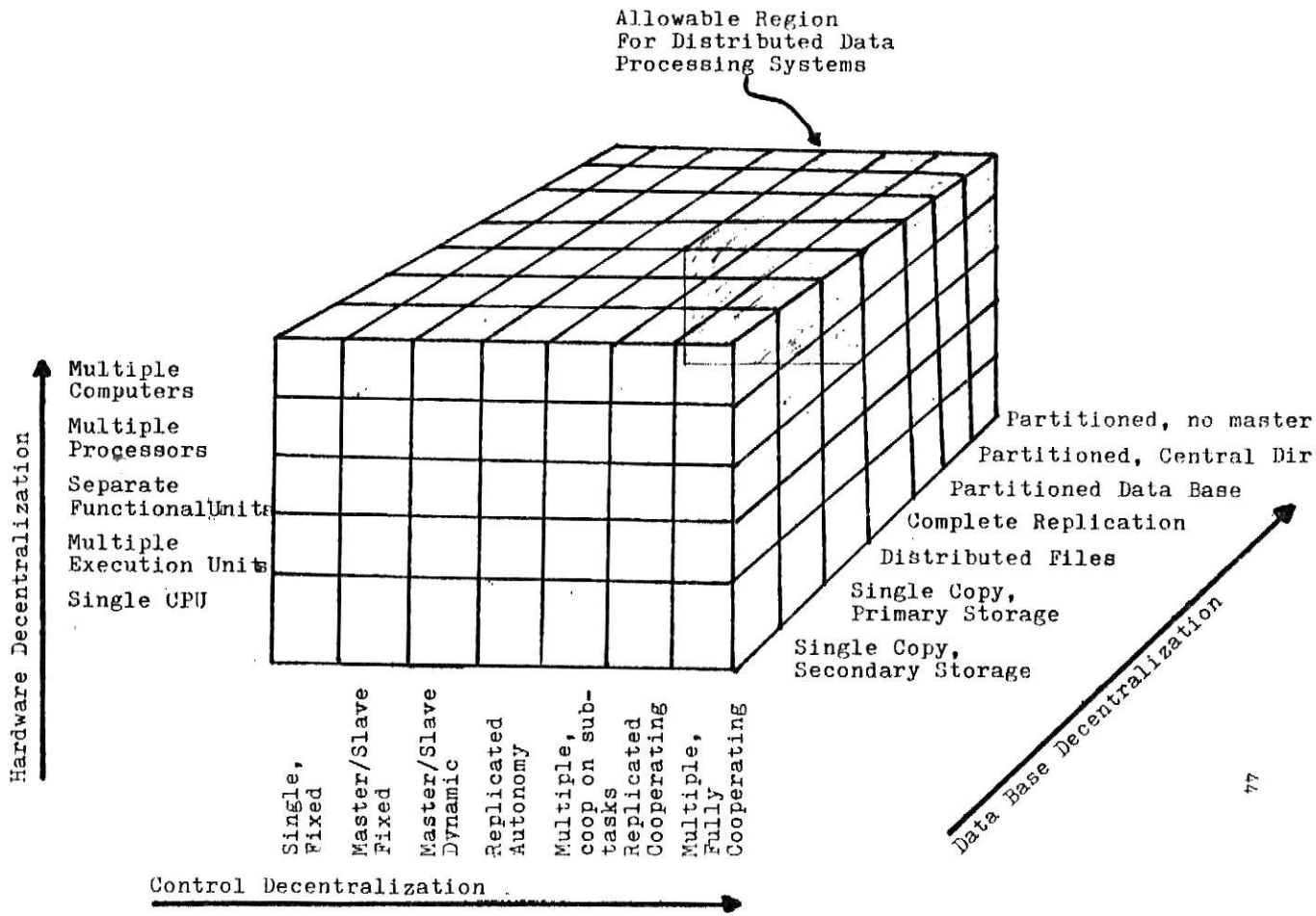


Figure 2-8: Dimensions Characterizing Distribution

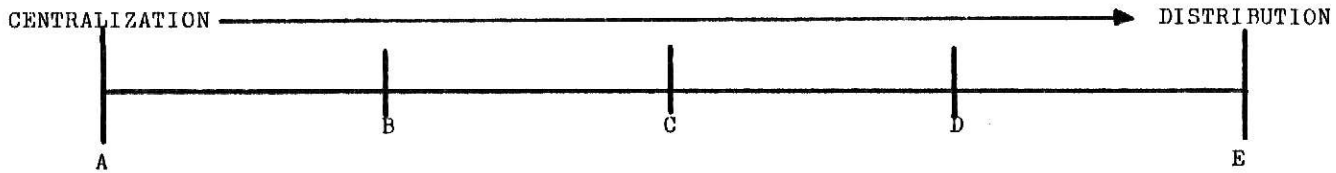
2.5 Summary and Conclusions.

Distributed processing has been examined from several perspectives. Its flexibility was discussed from the user viewpoint; its claimed advantages were analyzed and compared with those normally associated with centralized systems; and the views of four different authors were consulted. From these approaches, several facts now should be clear. First, distributed processing is still an evolving concept. Second, distributed processing systems generally consist of two or more interconnected processors, which share some common data processing function(s). These processors may be either general or special purpose or combinations. (Even intelligent processors or terminals tied to a large central system may be considered a distributed system, albeit of a very limited nature.) Third, a distributed processing system implies the existence of a network, i.e., thin-wire connection. It should be noted here that all networks do not necessarily imply the existence of a distributed system. For example, two or more processors might be linked together to form a network, but they cannot be considered a distributed system unless they share in the performance of some function. Fourth, organizations that stand to benefit most from distributed processing are those that are either geographically dispersed and/or composed of semi-autonomous sub-elements (divisions) or can benefit to some degree from

resource sharing with organizations outside normal organizational boundaries. [30,52-53,120,136,158] Fifth, systems should conform to organizational management philosophies and satisfy user needs. Sixth, distributed processing represents an evolution of existing/known technologies. What the final stage of this evolutionary process will be is unclear. Nonetheless, some of the desirable/mandatory features of fully distributed systems are beginning to emerge. These include decentralized system-wide executive control [11,46,74,118] or at least some form of system-wide discipline [156], system transparency [11,46,51,74,89-101,103-106,128], thin-wire or network-type communication systems [118], DDBMSs [11,46,51,81,89-101,103-106,128], and general purpose hardware resources [46,89-101,103-106]. (From the total system view, these resources could be nodes where both general and special purpose processors function as an entity.) Seventh, distributed processing is both a multifaceted and a multidimensional concept.

From the larger perspective, the seven facts mentioned above suggest (1) distributed processing should be viewed as part of a larger continuum whose extent ranges from uniprocessor (completely centralized) systems to fully distributed systems (Figures 2-9 and 2-9a through 2-9e

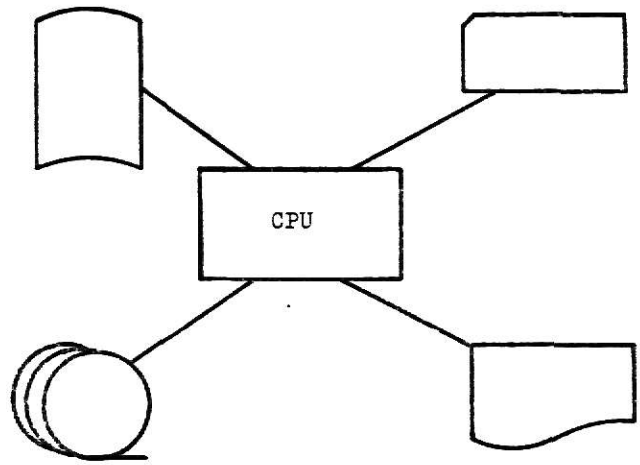
illustrate the continuum concept using five representative systems; Figures 2-10 and 2-11 present a subjective methodology for gross estimation of the percentage of distribution of each of the systems shown in Figure 2-9), and (2) any practical definition of distributed processing must be general enough to encompass a wide-ranging variety of systems. For this reason, the term, distributed processing, when used throughout the remainder of this report is defined as a network of processors which share some common data processing function(s).



(NOTE: Descriptions of Systems A Through E are found in following figures.)

Figure 2-9: Continuum Representation

SYSTEM A

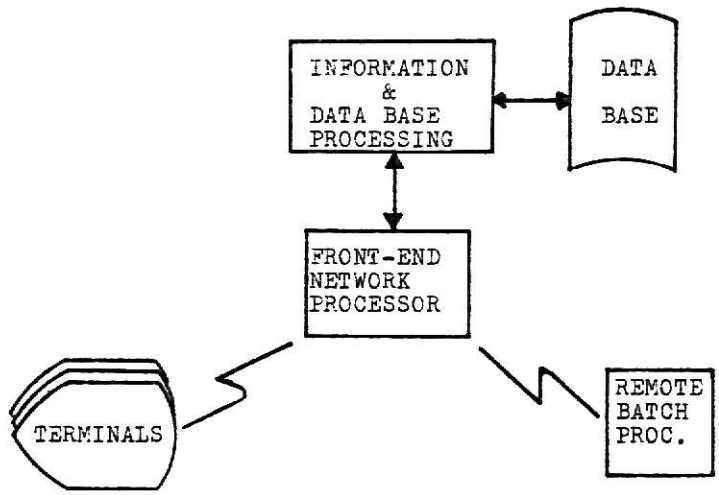


CHARACTERISTICS	TYPE	
System Components		
Hardware	C	
Control	C	
Data		
Applications	C	
Management	C	
Functions		
Information Processing	C	
Network Processing	n/a	
Data Base Processing	C	
Organizational		
Management	C	
User (Remote)		
Availability	n/a	From remote user view, distribution equates to these characteristics.
Reliability	n/a	

(NOTE: Types include Centralized, Partially Distributed and Distributed.)

Figure 2-9a: Centralized Uniprocessor System

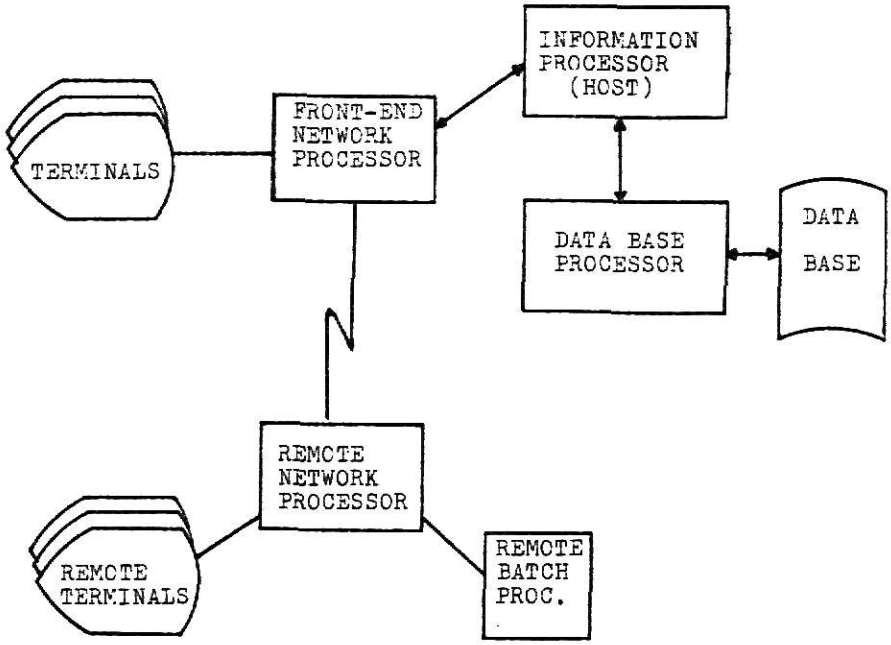
SYSTEM B



CHARACTERISTICS	TYPE
System Components	
hardware	P
control	C
data	
applications	P
management	C
Functions	
information processing	C
network processing	C
data base processing	C
Organizational	
management	C
User (Remote)	
availability	P
reliability	n/a

Figure 2-9b: Centralized w/ remote I/O & applications

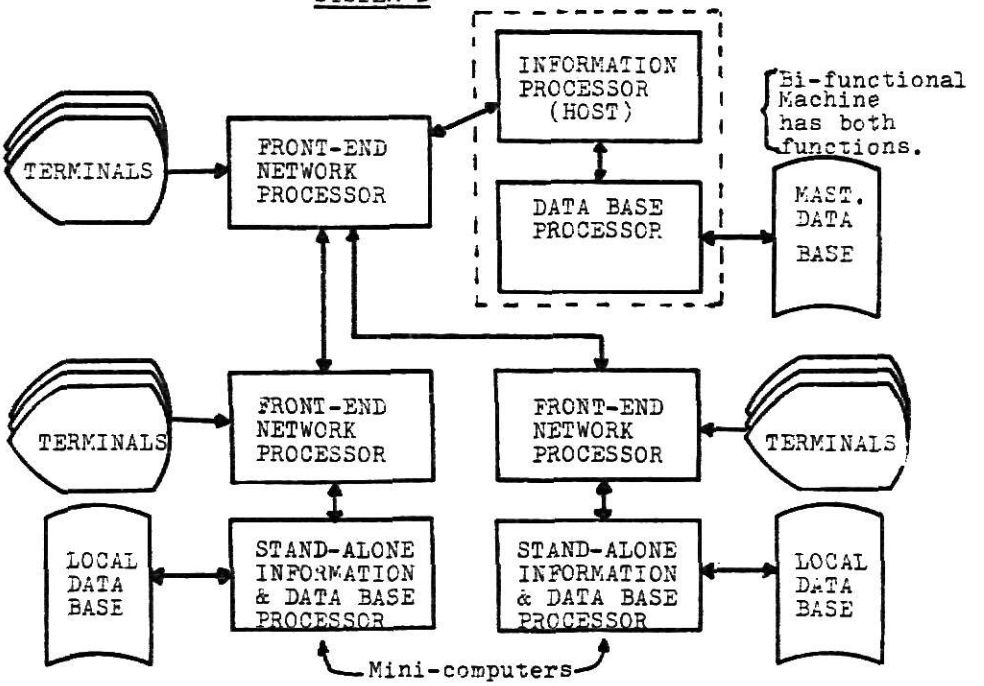
SYSTEM C



CHARACTERISTICS	TYPE
System Components	
hardware	P
control	C
data	
applications	P
management	C
Functions	
information processing	P
network processing	P
data base processing	P
Organizational	
management	C
Users (Remote)	
availability	P
reliability	n/a

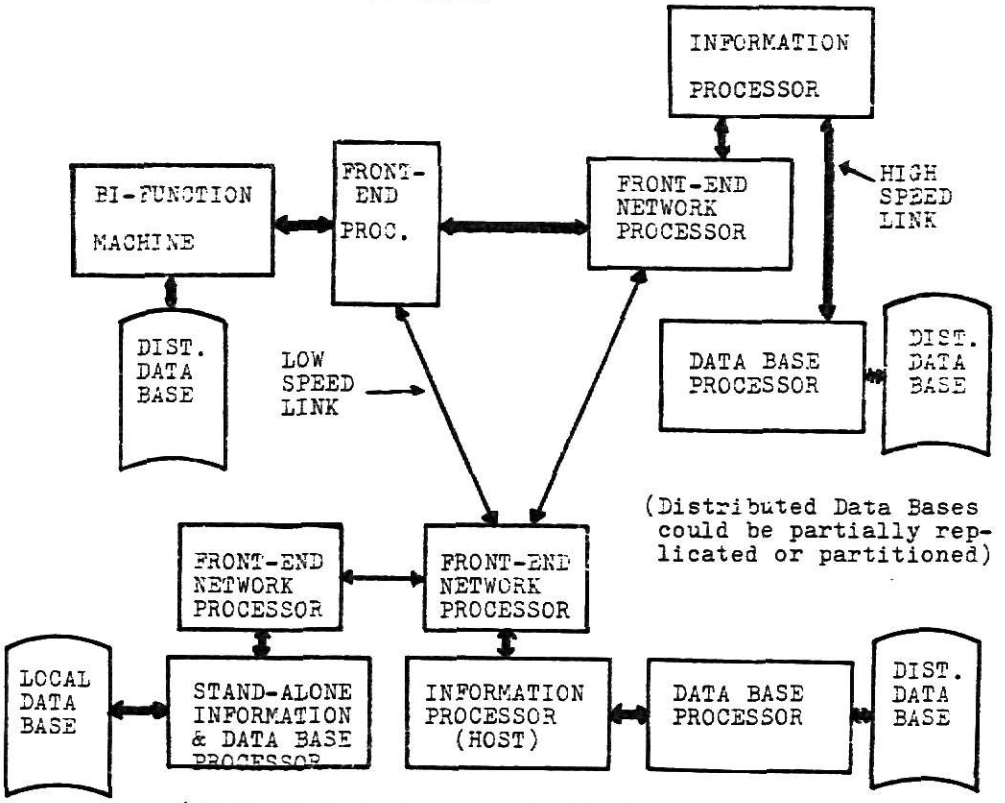
Figure 2-9c: Functionally Distributed Central Facility

SYSTEM D



CHARACTERISTICS	TYPE
System Components	
hardware	D
control	C
data	
applications	P
management	P
Functions	
information processing	P
network processing	D
data base processing	P
Organizational	
management	P
Users (Remote)	
availability	D
reliability	D

Figure 2-9d: Vertically Distributed System



(NOTE: Each system or node is assumed to have terminals attached.)

CHARACTERISTICS	TYPE
System Components	
hardware	D
control	D
data	
applications	D
management	D
Functions	
information processing	D
network processing	D
data base processing	D
Organizational	
management	D
Users (Remote)	
availability	D
reliability	D

Figure 2-9e: Fully Distributed System

A SIMPLE METHODOLOGY FOR ESTIMATING DEGREE OF DISTRIBUTION

GIVEN: -1Ø Characteristic elements (see Below).

-Each element's value ranges from Ø to 1Ø based on type, i.e., Ø for Centralized or n/a, 5 for Partially Distributed, and 1Ø for Fully Distributed.

-Other elements could be added; obviously there are degrees in between the values assigned above; however, for simplicity they will be used.

	SYSTEMS					
CHARACTERISTICS	A	B	C	D	E	
System Components						
hardware	Ø	5	5	1Ø	1Ø	
control	Ø	Ø	Ø	Ø	1Ø	
data						
applications	Ø	5	5	5	1Ø	
managemement	Ø	Ø	Ø	5	1Ø	
Functions						
information proc.	Ø	Ø	5	5	1Ø	
network processing	Ø	Ø	5	1Ø	1Ø	
data base processing	Ø	Ø	5	5	1Ø	
Organizational						
management	Ø	Ø	Ø	5	1Ø	
Users (Remote)						
availability	Ø	5	5	1Ø	1Ø	
reliability	Ø	Ø	Ø	1Ø	1Ø	
	Ø	15	30	65	100	TOTALS

(See Figure 2-11 for graphical representation of percentage of distribution for systems listed above.)

Figure 2-1C.

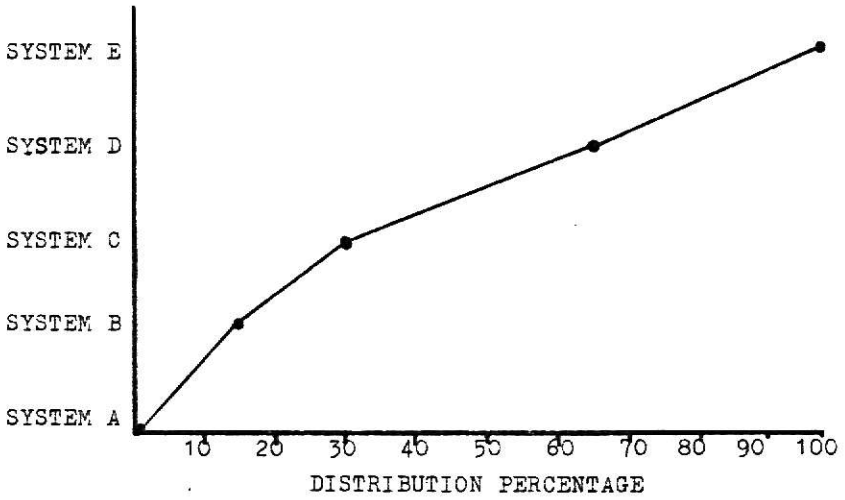


Figure 2-11: Comparison of
Five Systems

CHAPTER 3

DISTRIBUTED DATA BASE MANAGEMENT

3.1 Introduction.

Distributed data base management, one of the fastest growing areas of interest in computer science, is a manifestation of one of the major trends in society, business, and government today--decentralization and atomization. The intent of this chapter is to provide the reader with an overview of the key aspects of this developing technology. Three major topics will be covered. These are: (1) concepts and developments in DDBMS technology; (2) distributed data base design considerations; and (3) DDBMS problem areas. With regard to the last topic, both technical and organizational problems are discussed as well as the research and and development (R&D) efforts that have been targetted toward solutions to the various problems. Additionally, a comprehensive taxonomy or classification scheme for generalized DDBMSs is presented to facilitate understanding of the ramifications of the evolving DDBMS technology.

3.2 Concepts and Developments.

Like the broader field of distributed processing, definitions of DDBMS vary from author to author. The range of these definitions, however, is not quite as extensive as those for distributed processing. One very limited view is espoused by Hardgrave [63]. To him, a DDBMS only exists when there are separate data base management computers. Although his definition represents one type of DDBMS, it is, on the whole, much too restrictive. The broader definition offered by Rothnie and Goodman appears to capture most of the flavor of the distributed data base management concept and perhaps represents the main stream view of DDBMSs:

> Distributed data-base management systems...permit a collection of data which is relevant to a given organization to be managed on a network of geographically dispersed computers. [128, p. 30]

Peebles and Manning [119] probably would object to Rothnie and Goodman's usage of the term, "geographically distributed," to characterize DDBMSs, since they argue that usage of thin-wire communication linkage allows machines or processors to be located wherever needed. Booth's definition [22] is similar to that of Rothnie and Goodman;

however, she includes the proviso that a true DDBMS environment only exists when a process (program execution) at one location requires and receives access to data stored at another site. To encompass all the various combinations and permutations, a broad definition of distributed data base management will be used in this report, i.e., a DDBMS is a network of computers sharing the functions of a data base management system.

Five major types of hardware components (machines or processors) may be found in a DDBMS network. These are:

1. Front-end or network processors, which provide user external interface;

2. Hosts or large-scale multiprogrammed computers, which execute applications programs;

3. Data base machines (dedicated, special purpose data base management processors);

4. Back-end processors, which either may be general purpose computers (usually multiprogrammed minicomputers) or data base machines;

5. Bi-functional machines, which can perform the functions of both a host and a back-end.

From an organizational viewpoint, distributed data base systems have been the subject of much discussion

[5,22-23,42,63,81,86,89-106,111,114,118-119,121,123,128]. While the terminology used in these reports may differ slightly, the authors have focused primarily on two major approaches to the development of DDBMSs--the back-end approach and the generalized DDBMS approach. These will be discussed in turn.

3.2.1 Back-end Approach.

The simplest or most elementary form of DDBMS is a two machine configuration--host and back-end. Canaday et. al. [26] of Bell Laboratory first proposed and prototyped a host/back-end scheme in 1973-4. Since then, several similar systems have been proposed and/or developed [22,85,121,160-161]. (The most extensive work in this area has been done at Kansas State University.) The basic idea behind the back-end concept is to use a general purpose computer (usually a minicomputer) to handle most DBMS functions, i.e., to act as an interface between the host computer and the data base (see Figure 3-1) much the same as a front-end acts as an interface between the host and its external environment. [26]

(Source: 26)

Figure 3-1: Back-End System

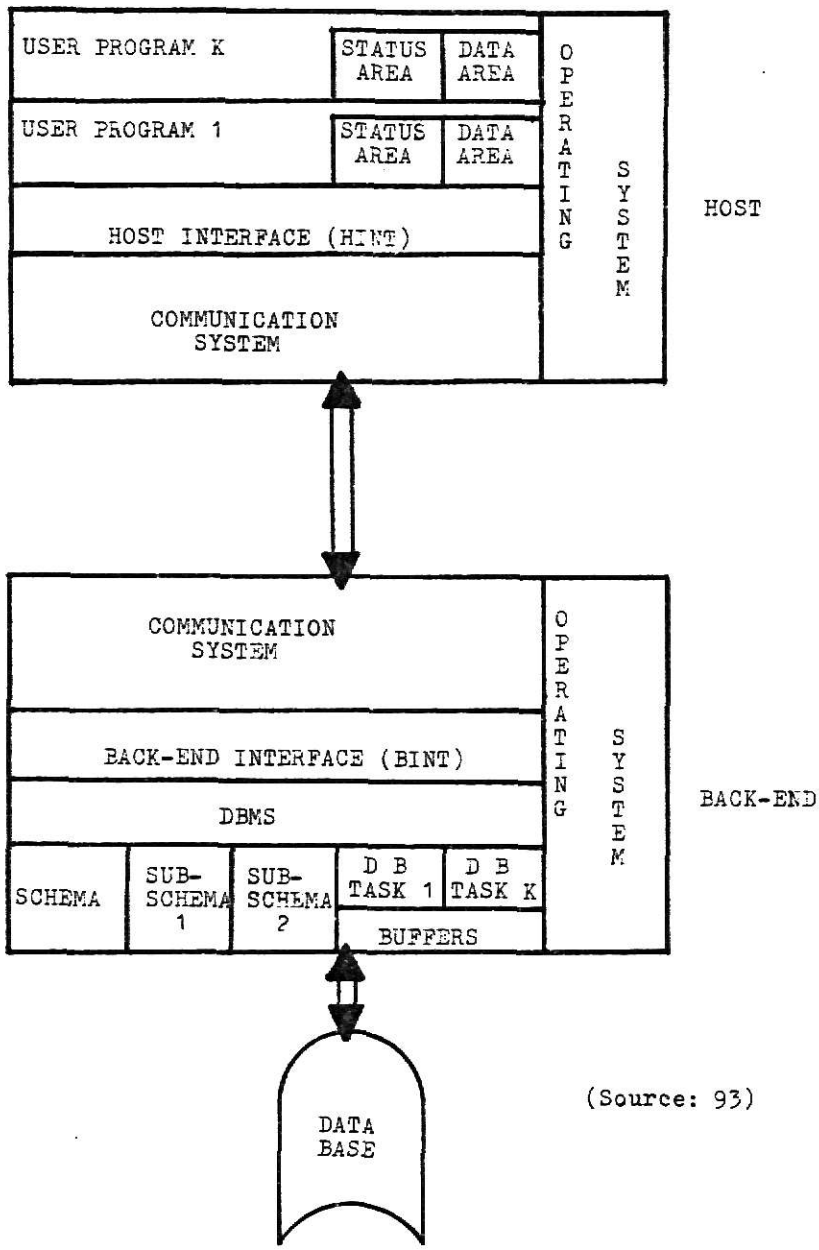
Since physical access to the data base is restricted to the back-end DBMS, this arrangement can provide a measure of protection and security to the data, because no malevolent program on the host can access or alter the data base directly. Other potential benefits include extending the life of the host computer, economy of scale and upgrade (cost of minicomputer versus cost of upgrade of host system), reduction of host operating system overhead and memory usage, enhanced error detection capabilities (host and back-end check each other's status), and increased throughput. (With regard to the latter, Maryanski and Wallentine [106] reported that simulations showed that the addition of a multiprogrammed back-end to a multiprogrammed host could provide over a 60 percent improvement in data command throughput.) Several of the advantages, e.g., reduction of host operation overhead and memory usage and economy of upgrade, are advantageous only if the capabilities released by movement of the DBMS to a back-end can be used effectively by other processes. Potential disadvantages include the costs associated with an additional machine (maintenance, system programmer support, operator training, etc.), unbalanced resources (either the host may be less than fully used and the back-end overloaded or vice versa), and increased response time overhead. [26,89] To interface a host and a back-end, additional software is needed. Specifically, interprocess

communication and interface routines must be added to the host and the back-end. These are described in detail in references 51,89,96,and 105 (see Figures 3-2 and 3-3).

The basic host/back-end configuration also can be extended to include the following architectures:

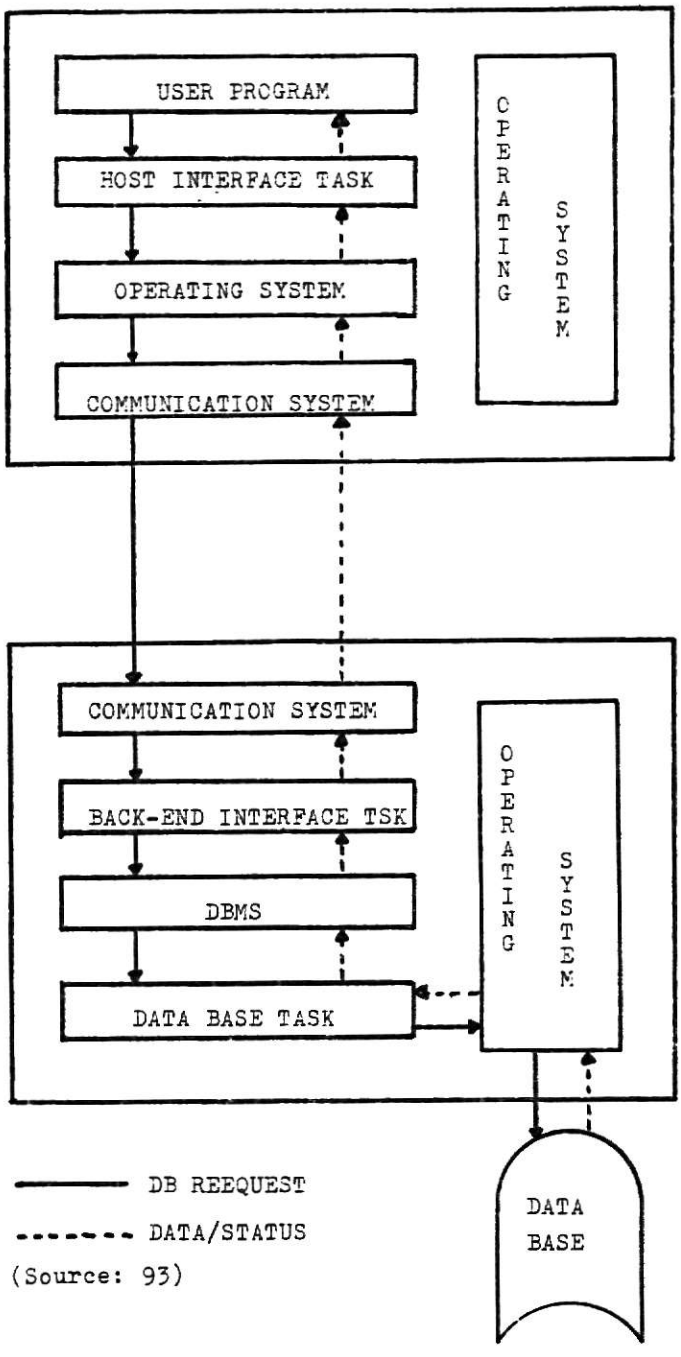
1. multiple hosts accessing a single back-end (Figure 3-4a);
2. a single host linked to multiple back-ends (Figure 3-4b);
3. multiple hosts internetted with multiple back-ends (Figure 3-5).

These structures generally have assumed homogeneity/compatibility of both back-end and DBMS resources; however, the host/back-end arrangement also has been extended to a host system internetted with heterogeneous back-end DBMS. [105] As shown in Figure 3-6, this configuration requires a threefold increase in the number and perhaps a greater increase in the complexity of the interface mechanisms.



(Source: 93)

Figure 3-2: Software Distribution Host/Back-end



(Source: 93)

Figure 3-3: Information Flow

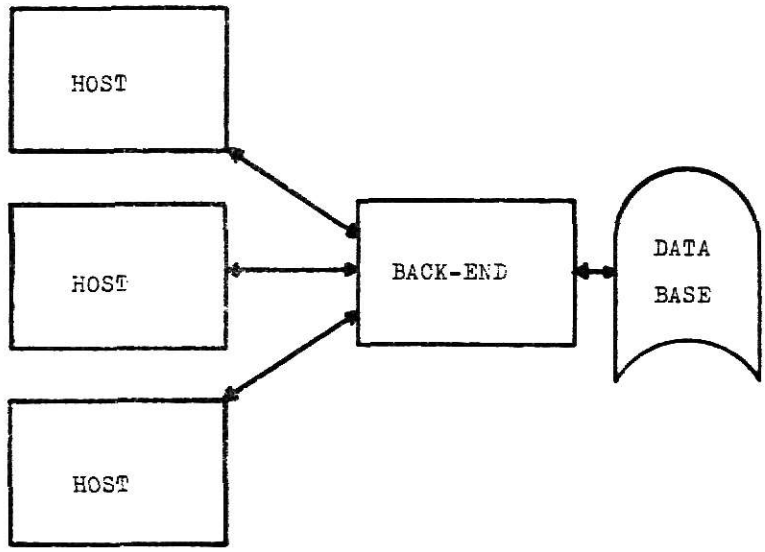


Figure 3-4a: Multiple Hosts/Single Back-end

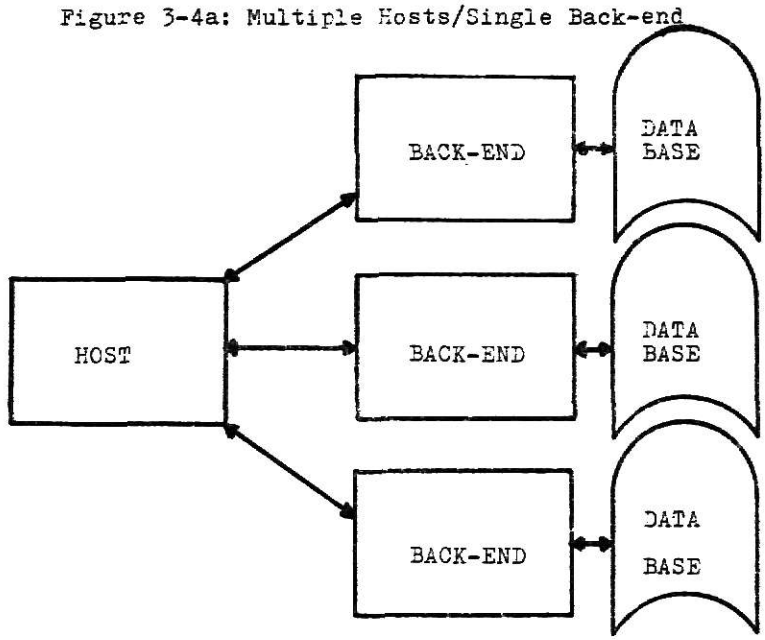


Figure 3-4b: Single Host/Multiple Back-ends

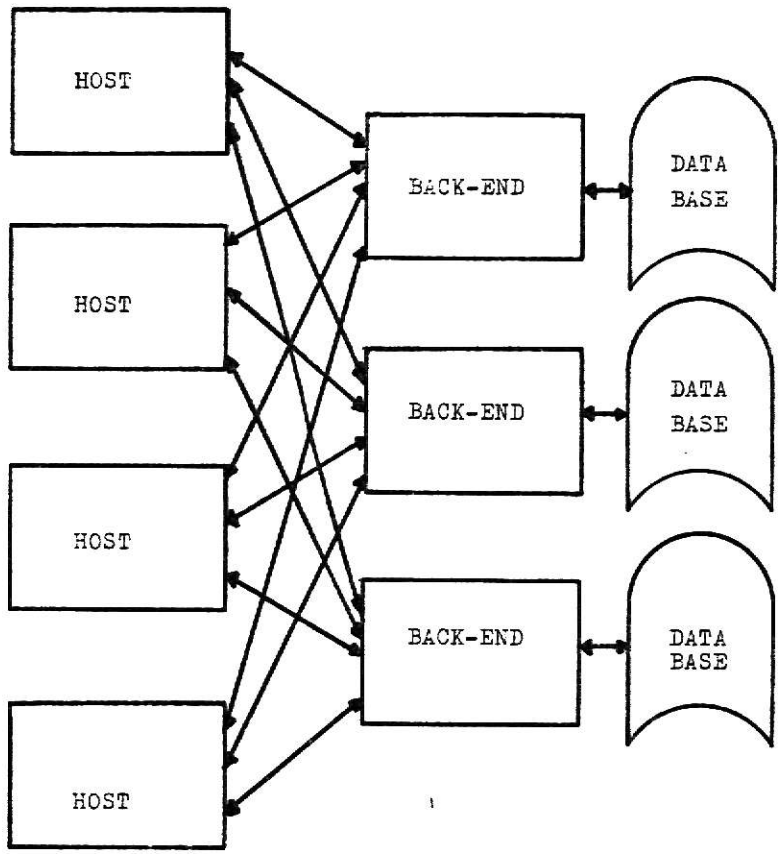
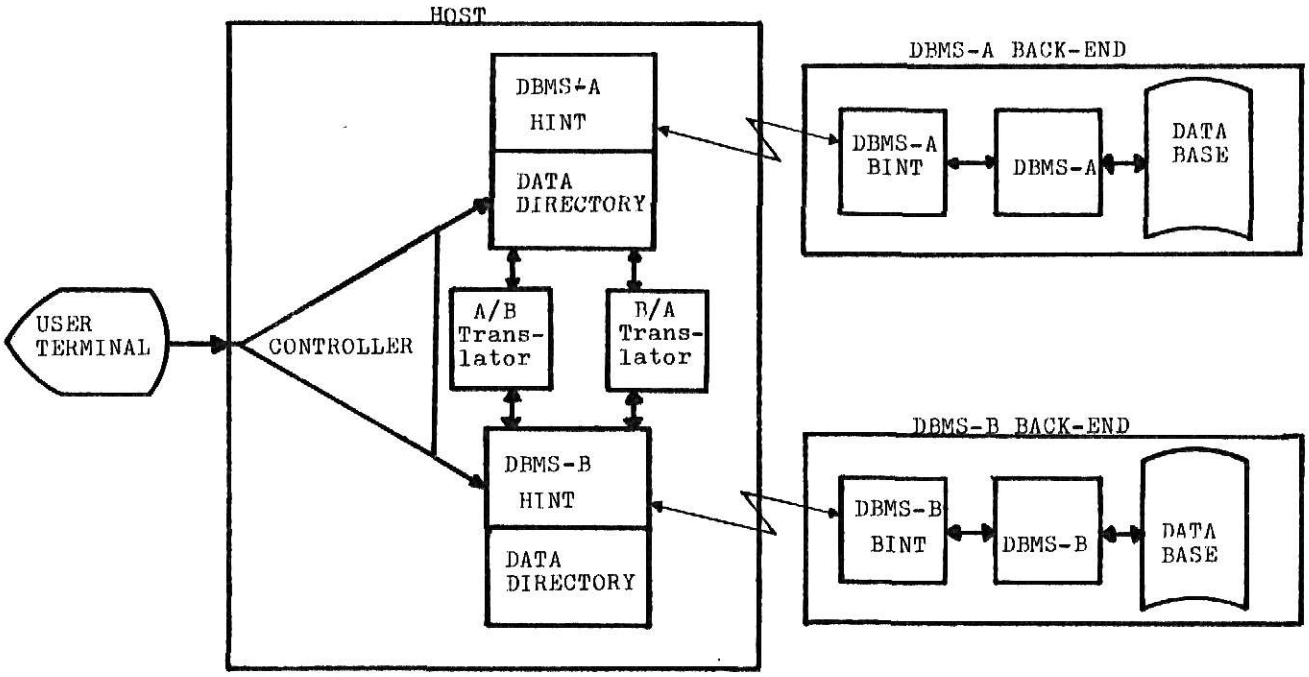


Figure 3-5: Multiple Hosts/Multiple Back-ends

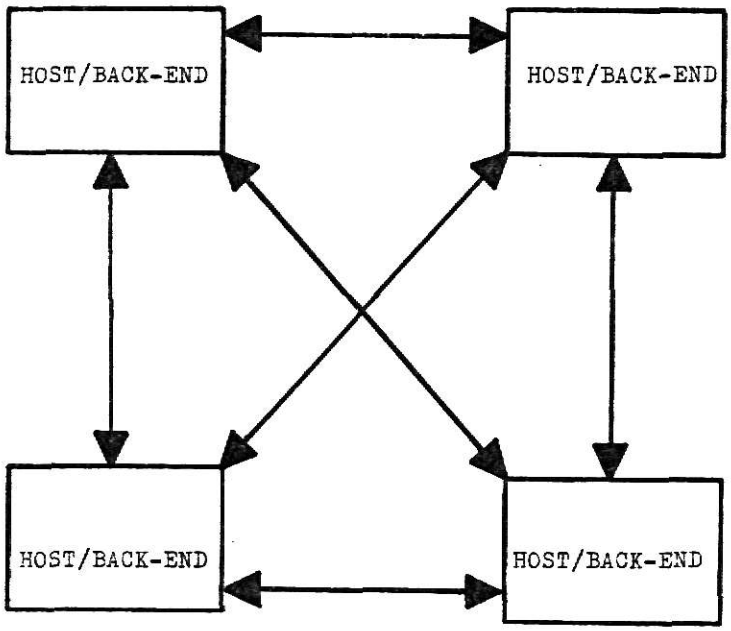


(Source: 105)

Figure 3-6: Host with Heterogeneous Back-ends

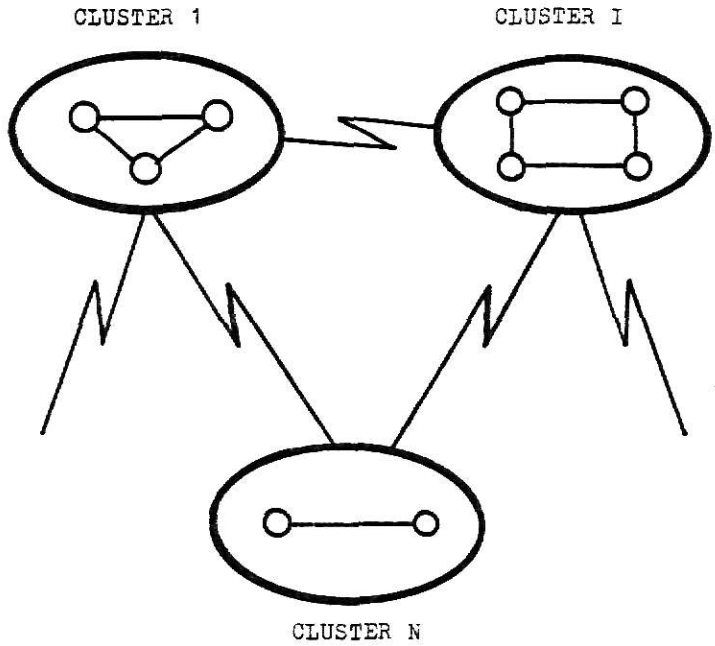
One of the reasons that emphasis has continued on back-end development is that this simple architecture can easily be extended, thus allowing for modular growth. In fact, a single host/back-end could be used as a starting point for development of a full-fledged generalized distributed data base system (see Figures 3-7 and 3-8). [161]

Since the potential exists, as stated earlier, for a general purpose, multiprogrammed back-end to be less than fully used, Maryanski, et. al. [94] have suggested that the back-end could be transformed into a bi-functional machine, which would be capable of serving both as a host and a back-end. While the security advantages afforded by the separation of the host and the back-end functions would be denigrated, the bi-functional configuration potentially would allow for more efficient utilization of system resources (see Figure 3-9 for a conceptual view of a DDBMS composed of hosts, back-ends, and bi-functional machines). To operate as a bi-functional machine, a computer must have both the host and the back-end interface routines as well as the DBMS (to include schema, subschemas, and network directory) and the interprocess communication system (Figure 3-10).



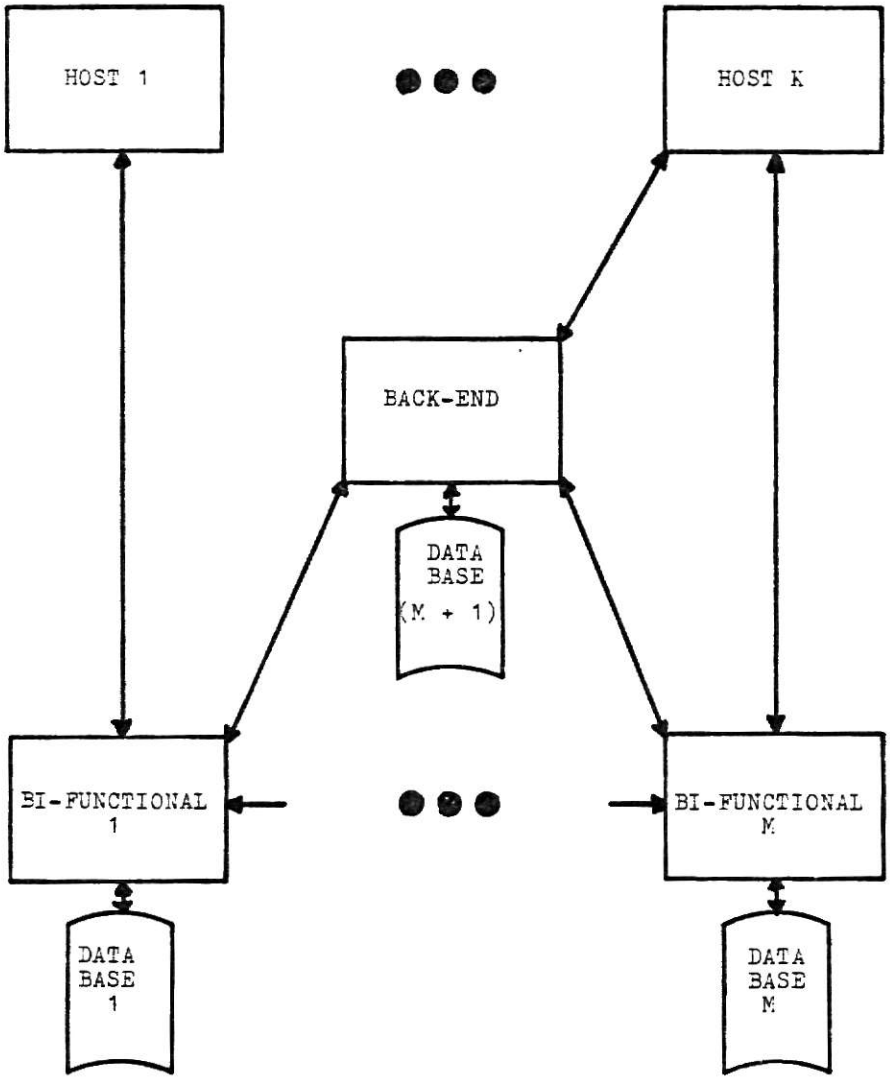
(Source: 51)

Figure 3-7: A General Distributed
Data Base System



(Source: 51)

Figure 3-8: A Geographically Distributed System



(Source: 93)

Figure 3-9: Distributed Data Base System

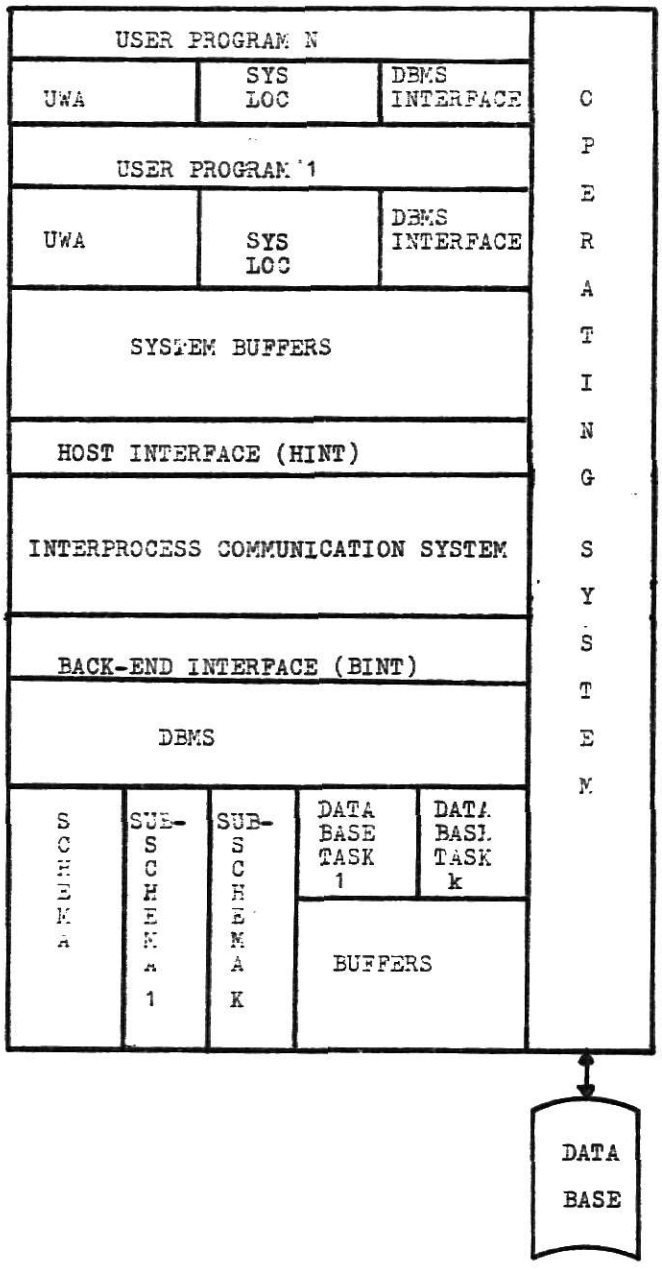


FIGURE 3-10: Bi-Functional Software Distribution

Although its origins can be traced to the mid-1960's, the development of data base machines has paralleled the prototyping of back-end systems. Data base machines, in fact, are the latest manifestation of a well-defined trend, i.e., incorporating functions that initially began as software into hardware (index registers, floating point arithmetic, and virtual memory). [24] Essentially, data base machines are specialized processors which support basic DBMS functions. Development of data base machines has been prompted primarily by the desire to improve overall data base management performance. Conventional or Von Neumann-type computer systems simply were not designed to support basic data base management functions efficiently. Instead, they were designed for numerical computations and for simple data processing and program execution. [71] Functions that might be allocated to data base machines include content associative addressing or searching and management of storage hierarchy. [24] Champine [29] suggests four ways that a data base machine might be incorporated into a system--back-end processor for a host computer, intelligent peripheral control unit, storage hierarchy, and network node. Thus far, only two special purpose associative processors have been marketed. These are the Content Addressable File Store (CAFS), which was introduced by ICL in November 1977, and STARAN, an associative system developed by Goodyear Aerospace. (The

latter is being studied jointly for data base usage by Berra, et. al. at Syracuse University and the Rome Air Development Center.) Four other data base machine architectures have been studied/prototyped by universities. These are CASSM (Context Addressable Segment Sequential Memory) developed by Lipovski, Su, et. al. at the University of Florida [149]; RAP (Rotating Associative Processor) designed by Ozkarahan, Shuster, and Smith at the University of Toronto [117]; RARES (Rotating Associative Relational Store) developed at the University of Utah by Lin, et. al. [83]; and DBC (Data Base Computer) designed under Hsiao at Ohio State [9-10]. Most of these special purpose machines use an associative memory approach, which allows data to be searched for and addressed by its contents or value rather than the conventional Von Neumann machine physical address approach. (The lack of an effective content addressable system is one of the key stumbling blocks to the fielding of a viable relational data base management system.) Although most of the associative prototypes rely on fixed head disks as their mass storage media, continuing technological developments, e.g., bubble memory and charge-coupled chips, may soon change this orientation. Because of the interest generated in such systems, it is probable that several additional associative processors will be marketed with the next two years. Detailed tutorials of the whole data base machine development effort can be found in the March 1979

issue of Computer, which contains references 19,69,71,77,144, and 150.

3.2.2 Generalized DDBMSs.

From a structural perspective, distributed data base system structures have been described variously [5,22-23,42,63,81,86,104-105,111,114,118-119,121,123,128]. Notwithstanding, the differing and often conflicting terminology used, basic system structures are essentially the same. In May 1978, Slonim, et. al. [142] proposed a standardized classification methodology. Unfortunately, this effort was both limited in its considerations (data allocation, hardware, and software) and flawed in its presentation. In this report, a new classification scheme or taxonomy is proposed to identify/describe distributed data base systems. This taxonomy keys on six factors--five independent factors and one dependent factor (see Figures 3-11 through 3-11d). The independent factors considered include system design approach, data allocation strategy, structural interconnection methodology, hardware component type, and data base components. Data conversion/translation requirements, the last factor, is functionally dependent on the preceding two independent factors--hardware component type and data base components.

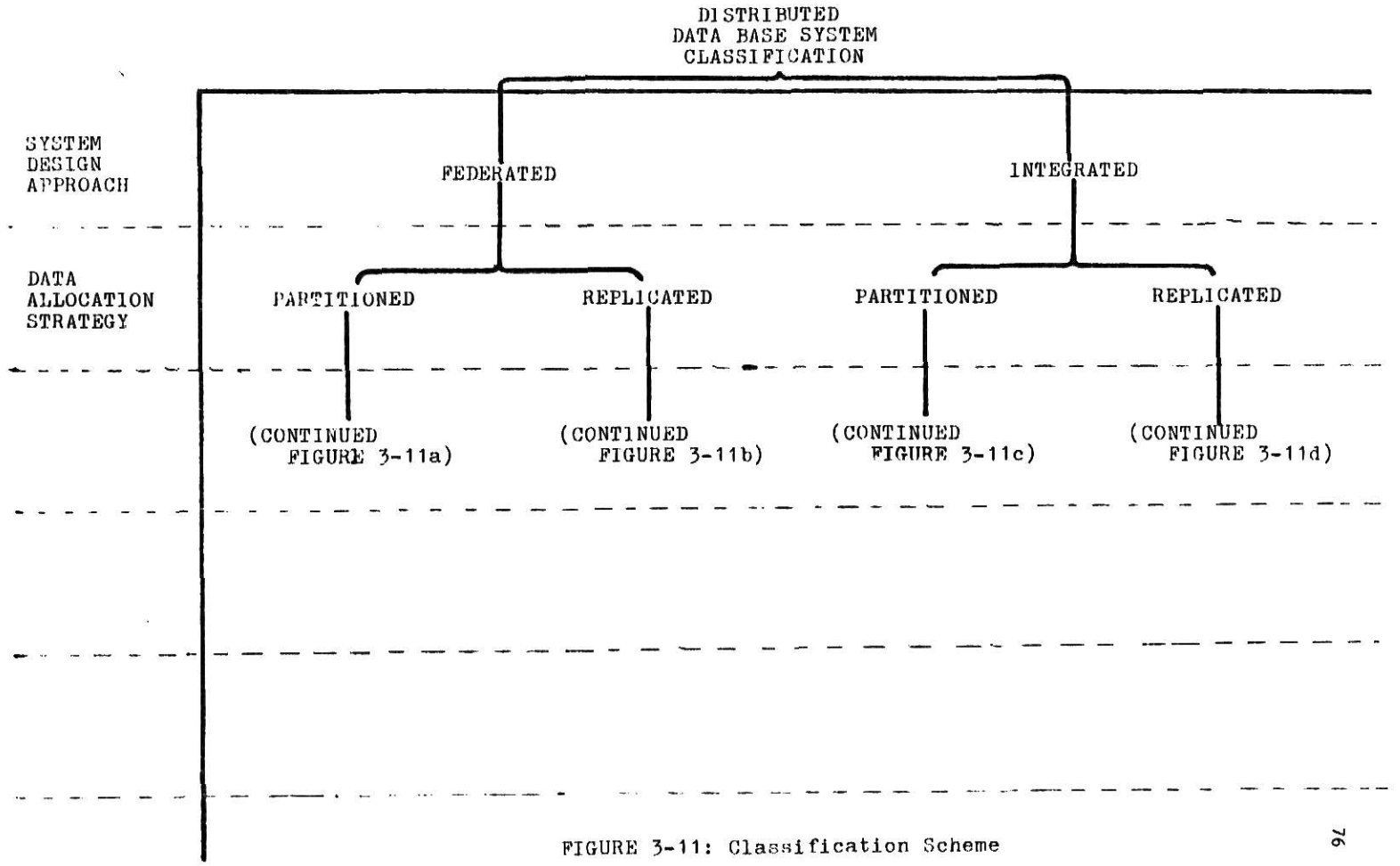


FIGURE 3-11: Classification Scheme

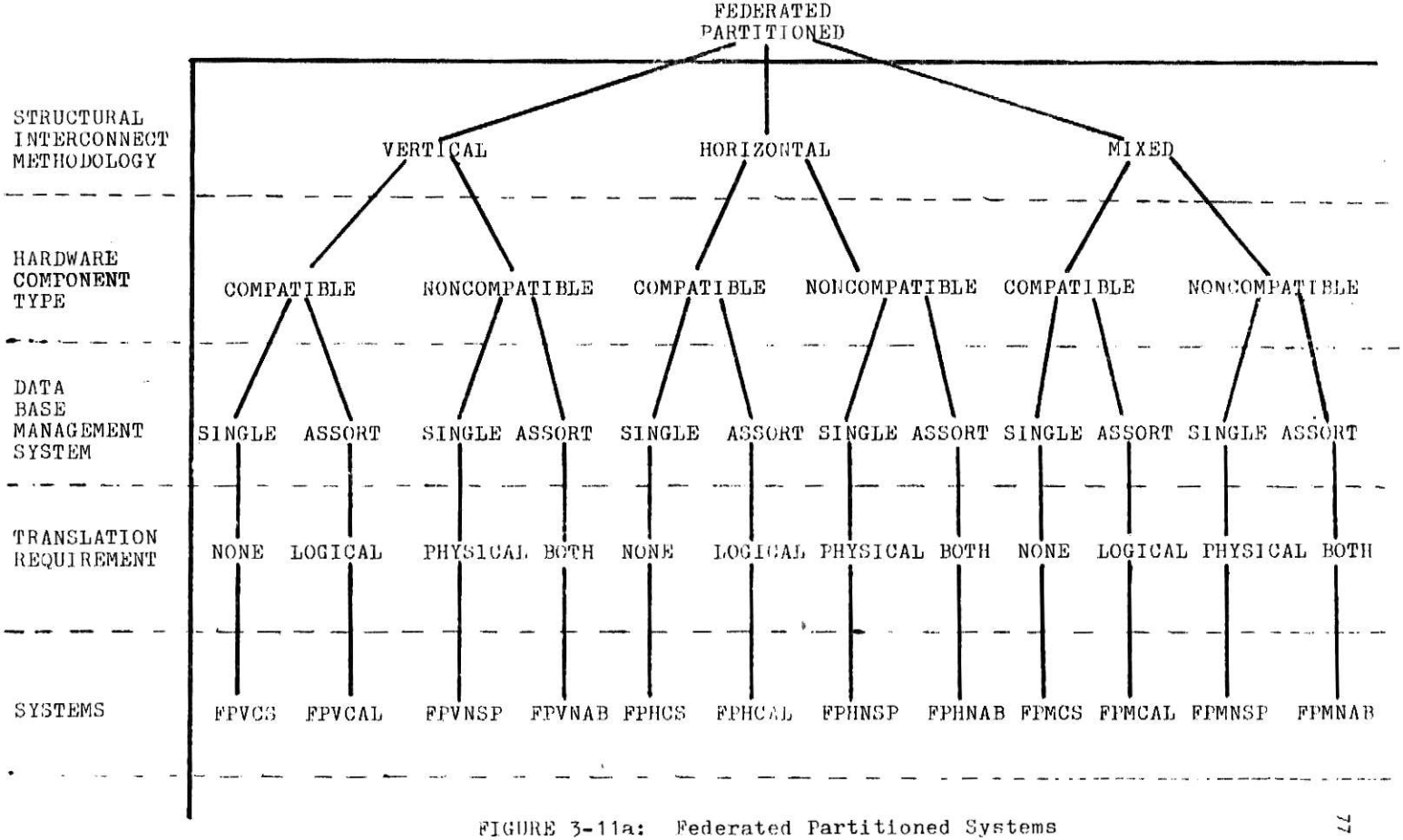


FIGURE 3-11a: Federated Partitioned Systems

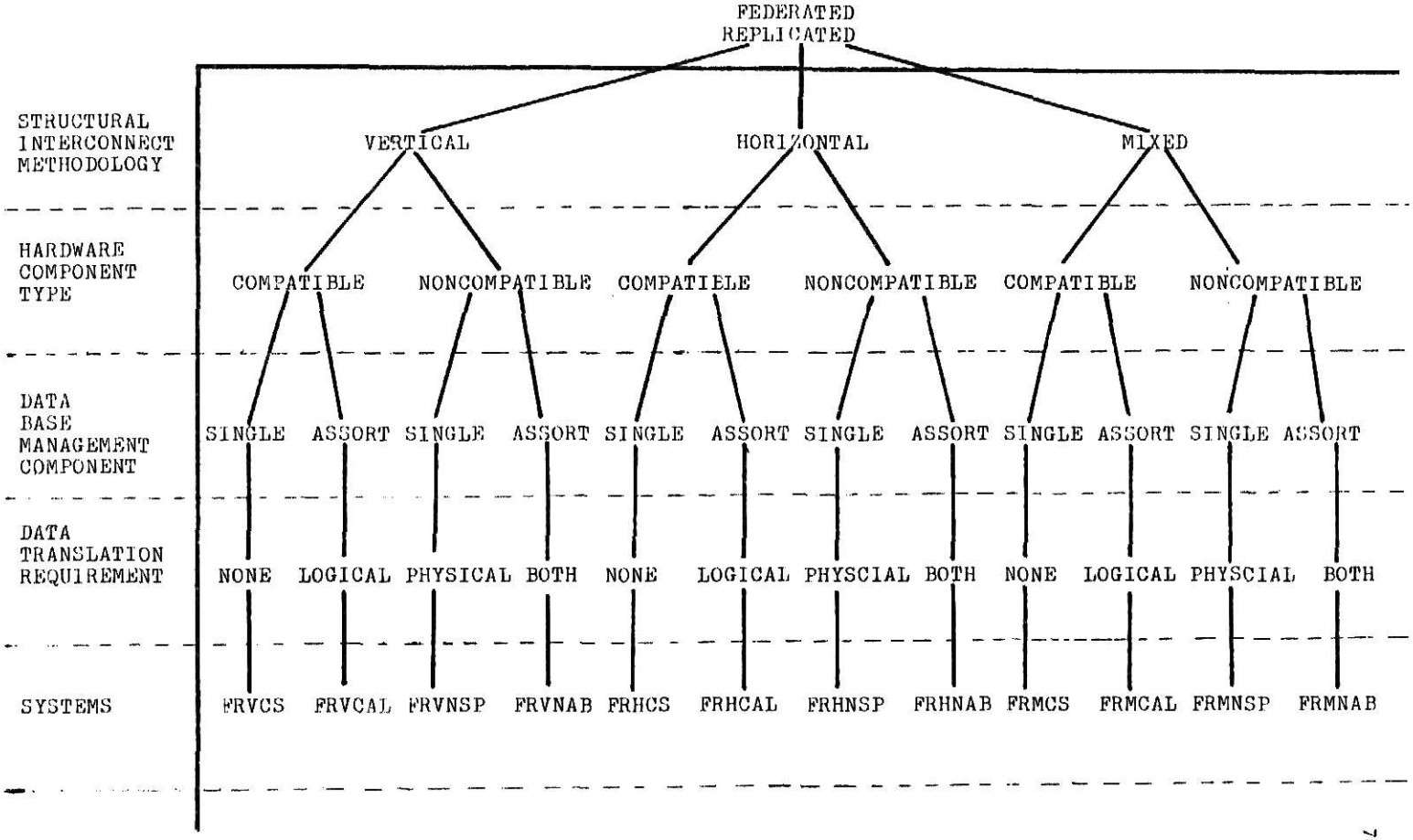


FIGURE 3-11b: Federated Replicated Systems

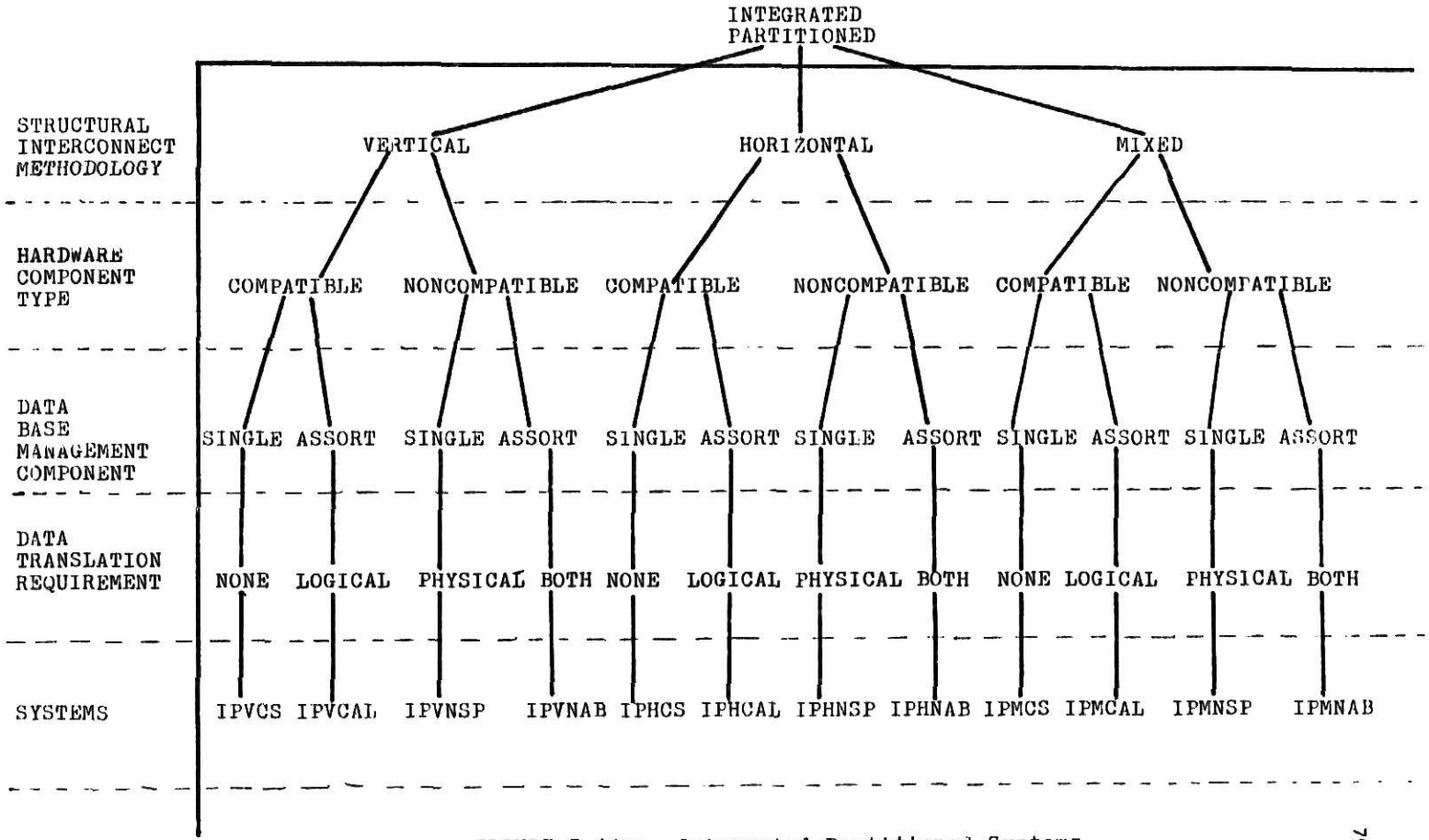


FIGURE 3-11c: Integrated Partitioned Systems

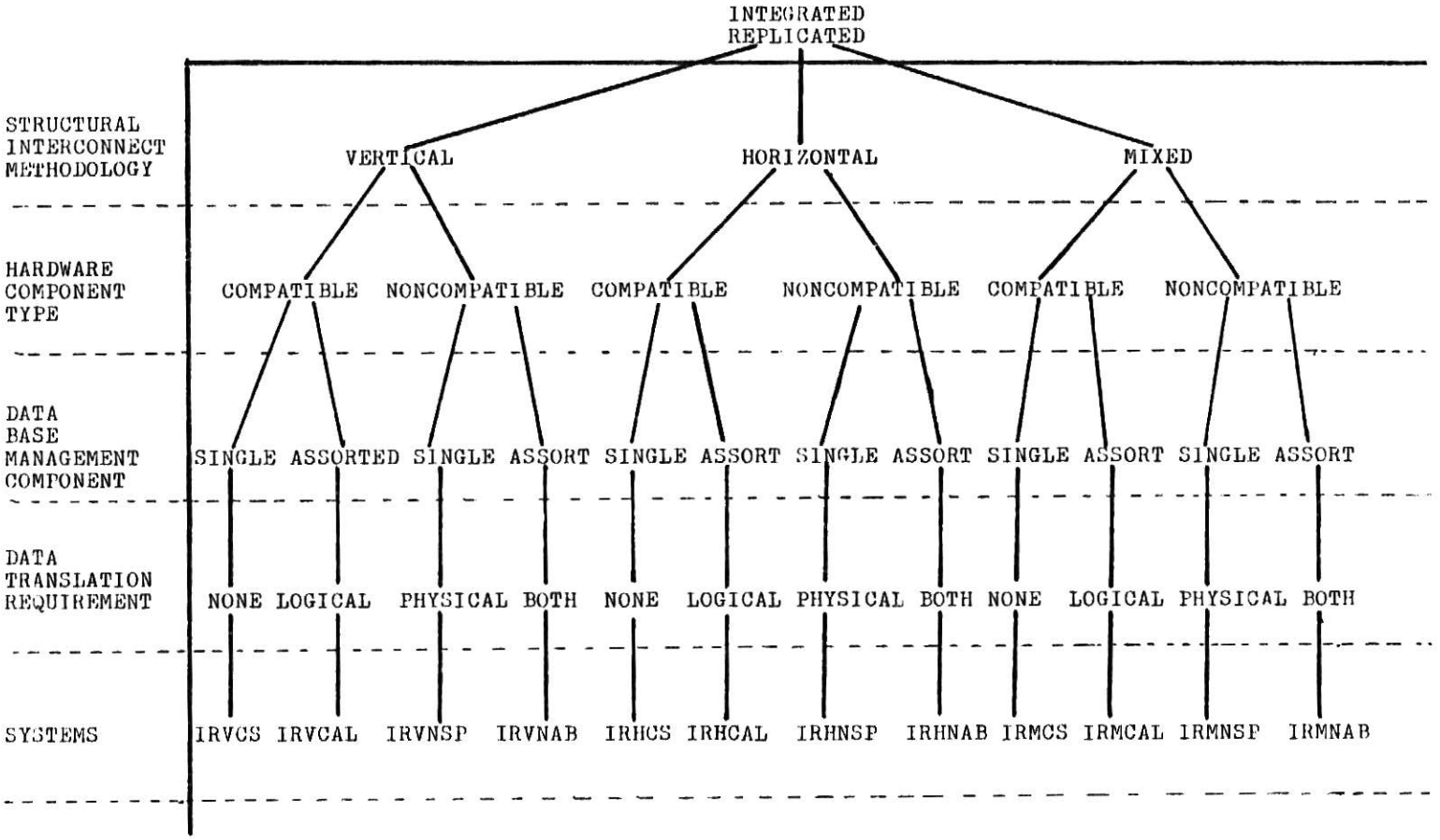


FIGURE 3-11d: Integrated Replicated Systems

There are two basic approaches to system design. These are the federated or bottom-up and the integrated or top-down. [118] A federated or compositional [81] system usually results from the merger of several existing stand-alone DBMSs. Integrated [82] or decompositional systems [81], e.g., Rothnie and Goodman's SDD-1 [129], are the result of decomposing the functions and the data base of an existing or proposed very large data base into multiple interrelated systems. Under data allocation strategy, four approaches are possible--centralized, fully replicated, partially replicated, and partitioned. In this scheme, the fully redundant/replicated strategy, where the entire data base is stored at every node, and the centralized approach, where the entire data base is stored at one central node, are not considered viable options in a generalized DDBMS environment. Therefore, they are not included as alternatives in the classification scheme. From a structural interconnection perspective. Three classifications are possible--vertical (hierarchical), horizontal, and mixed or complex [22]. In a vertical system, the interconnected processors or nodes form a hierarchy and share tasks in a structured way with each lower-level node being controlled to some degree by higher-level components. Horizontal systems, on the other hand, cooperate and co-exist logically as equals in task performance. Instances of both vertical and horizontal interconnections exist within mixed systems (Figure 3-12).

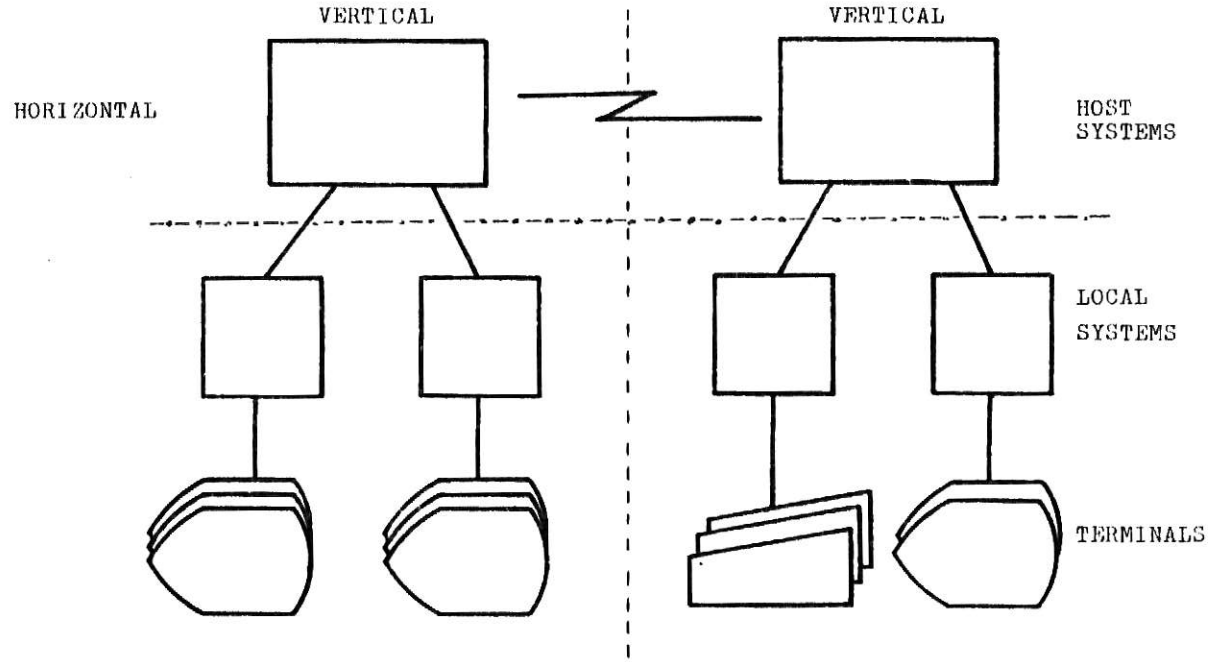


FIGURE 3-12: Mixed System Architecture

From a hardware component view, several classifications are possible. For example, individual processors might be classified either as general or special purpose. This taxonomy, however, is more concerned with internodal rather than intranodal considerations; therefore, individual processor classification is not as meaningful as internodal compatiblity and noncompatibility. For this reason, individual component classification is ignored. Instead, since this scheme assumes that all nodes whether they are composed of general or general and special purpose processors are general purpose from the total system view, only nodal compatibility and and noncompatibility is considered germane. A noncompatible or heterogeneous system as described in Chapter 2 is composed of computers having differing internal data representations, e.g., IBM 370 and CDC 6600. Compatible or homogeneous systems are formed using either identical or completely compatible computers, e.g., IBM 370 series or Honeywell 6000 series. From a data base component perspective, the data base management system components of a distributed data base system may either be single (the same DBMS at every node) or assorted (differing DBMSs throughout the system). Three classifications are possible for the sixth factor (data conversion/translation). These are logical, physical, and both logical and physical. Compatible hardware and single DBMS require no conversion/translation during interprocess communication.

All other categories require one of three types of data conversion/translation mentioned above.

Forty-eight distinct distributed data base systems can be delineated with this classification scheme. Doubtless, fewer than half of these system types will ever be implemented. For example, it is improbable that integrated systems will be implemented with noncompatible hardware and assorted DBMSs. It is also doubtful that a partitioned data allocation strategy will see widespread usage for reasons which will be discussed in Section 3.3. The most probable implementations are the federated replicated horizontal, the federated replicated mixed, and the integrated replicated horizontal architectures. Regardless, generalized DDBMSs must be designed to handle all types of systems to include the worst cases, i.e., federated replicated vertical noncompatible assorted systems, federated replicated mixed noncompatible assorted systems, and federated replicated horizontal mixed noncompatible assorted systems.

3.3 Distributed Data Base Design Considerations.

As discussed in the previous section and in references 27, 31, 81, 113, and 123, there are a number of fundamental issues that must be addressed in the design of any

distributed data base system. This section focuses on three of these issues--data distribution, DBMS component location, and application program location. Other important aspects such as security and user interface can be found in references 73, 76, 80, 116, and 152.

3.3.1 Data Allocation.

Four basic data distribution alternatives are recognized for the DDBMS environment. These include replication, partitioned, partial replication, and centralization. Although discussed as an alternative by several authors [17,44,130], replication is "...clearly impractical for data bases of useful size; realistically this approach should be viewed as a pedagogic simplification of the general 'partially redundant' approach...." [128, p. 50] Likewise, the centralized alternative is also of limited utiliy in a generalized DDBMS environment. For these reasons, both the fully redundant (replicated) and the centralized strategies are ignored here as they were in the classification scheme presented earlier. This leaves two alternatives--partitioned and partial replication.

A partitioned data base [22] exists when a logical data base is divided into unique subsets, which then are stored throughout a system. The term, "logical," is used to describe the data base, because it is unusual for a data base to be created and then be partitioned. Instead, a data base usually is designed first as a logical entity, and then it is implemented in subsets. Collectively, these interrelated subsets form a logical data base. The partitioned approach simplifies or eliminates many of the DDBMS problems that are discussed later (Section 3.4); however, the cost of this simplification must be borne by the users, because systems that do not allow some redundant data cannot easily fulfill many of the promised benefits of distributed data base systems, e.g., reliability, availability, and flexibility. The advantages of the partitioned approach are as follows:

1. Data base integrity potentially is easier to guarantee, because all data is located at unique sites. (In an ideal situation, these sites are close to the users who have a primary interest in the validity of the data.)

2. Rollback and recovery is straightforward, because failures or erroneous data can be traced to individual sites.

3. Update processing is less complicated than for

partially redundant systems, because there is only one copy of the data.

4. Security potentially is easier to maintain, because the total system is relatively static, i.e., since data migration must be strictly controlled in a partitioned data base system, it is probable that some form of centralized control/monitoring will be imposed, and centralized control usually implies better security.

The partitioned strategy also has some drawbacks. These include:

1. lack of flexibility (few sites are ever static in their usage of data, yet partitioning assumes a severely restricted data migration policy);

2. reduced reliability (The failure of one node deprives all users from access to the data stored on that node; in essence, node failure implies total system failure from the standpoint of availability, reliability, and integrity.);

3. diminished system transparency (System transparency, i.e., invisibility of data location to users, is potentially more difficult to maintain in a partitioned system, especially if thin-wire linkage is the interconnecting media and users routinely access data remote

from the node that services their requests.);

4. increased communication cost and reduced performance (As in 3 above, if users routinely access non-local data and the data to satisfy a significant number of the requests for non-local data is stored at multiple sites, then communication costs potentially are greater than in systems where some data redundancy is allowed, and overall system performance from the users' perspective is degraded.).

In sum, the partitioned approach offers many benefits, especially from a system design/implementation viewpoint. However, this approach should be pursued only if users' requirements for non-local data are minimal and reliability, flexibility, and availability are not the most important considerations.

Like the partitioned data allocation strategy, partially replicated data bases exist when subsets of a logical data base are stored on differing machines. The partially replicated approach, however, allows the subsets to overlap. In essence, partial replication is a compromise between the fully redundant and the partitioned methodologies. Its advantages are:

1. The failure of one site or even the communications

system is not necessarily catastrophic in the short term, especially if data is distributed according to usage patterns and if the user interface with the communication system is designed such that it can automatically bypass failed nodes. (With respect to the former, it is assumed that data usage patterns are monitored continuously and data is reallocated periodically.)

2. System reliability is greater than in partitioned system, because the communication system is less critical to total system operations.

3. Data availability and system transparency are easier to maintain than in partitioned systems.

4. Data locality is easier to achieve, especially for physically separated users who require access to the same data with equal or near equal frequency.

5. Changes or increases in user activity are easier to accomodate, since the partially replicated approach is inherently more flexible/extensible.

6. Potentially less multi-node access is required to satisfy normal user requests; therefore, communication system costs are generally less than in a partitioned system.

7. System responsiveness from the users' perspective potentially is better than in a partitioned environment.

Although its advantages are imposing from the users' viewpoint, the partially replicated approach exacerbates many old problems and creates some new problems for the system designer/implementer. Disadvantages include:

1. If the update to query ratio is high, communication system costs will be higher than in a partitioned system.

2. Data base integrity is more difficult to maintain due to redundant data.

3. Rollback and recovery is more complex than for partitioned data base systems.

4. Data migration, although controlled, causes additional maintenance problems, e.g., if data location is volatile, system directories become extremely critical maintenance items.

5. Management of and the strategy for updating redundant portions of the data base can be extremely complex.

3.3.2 DBMS Component Distribution.

The components of the DBMS which could be distributed include the directory, the schemas, the subschemas, and the DBMS kernel. Since the system directory can stand alone, it

will be discussed separately from the other components. The functions of a system directory are as diverse as the system it describes. Typical tasks [156] include:

1. describing the schemas, subschemas, and data as well as application programs and network communication facilities;

2. defining data attribute location(s);

3. cross-referencing of data objects and natural language descriptions of the uses/meanings of the various data objects;

4. maintaining user authorization lists for retrievals and updates to schemas.

Distribution schemes for directories can be divided into two categories--partitioned and replicated. [118] The partitoned approach allows three alternatives:

1. centralization--the entire directory is stored at one site. This approach necessitates access to one site for every update or retrieval. Obvious advantages include ease of directory update and centralized control, while the principal disadvantages include lack of flexibility, potential for catastrophic failure due to directory site or communication system failure, increased communication cost

(since every access must consult the centralized directory, and built-in response time overhead even for data resident at the node where the query originated.

2. distribution--each site has a directory containing information on the data stored at that site, thus local accesses can proceed using the local directory. Any request for non-local data must be broadcast over the communication system, since no site is cognizant of the location of data at any other site. The primary advantage of this approach is the reduction in overall access time for local queries. Disadvantages revolve around the requirement for non-local data access, which must be done via broadcast. Broadcast is more expensive from a communication cost standpoint than even the centralized approach, since all requests for non-local data must be broadcast to every node in the entire system. The potential for catastrophic failure is less than in the centralized approach; however, any approach that relies on broadcast is inherently unreliable. [119]

3. Combinations--an intermediate approach is to partition the system into several segments or groups of nodes and store complete system directories at centralized segment sites. This approach ameliorates some of the disadvantages of both the centralized and the distributed approaches; therefore, its biggest advantage is the fact that its disadvantages are not as extreme as either the centralized or the distributed approaches.

The replicated approach's three alternatives are:

1. centralized--each local site has a directory that describes the components of the data base stored at that site, while one central site has a complete directory. Advantages are greater flexibility than any of the partitioned methods and lower potential for catastrophic failure than the centralized partitioned alternative. Disadvantages include contention for a single resource for non-local queries and communication response time overhead for non-local queries/updates.

2. distributed--each site has a complete system directory stored locally. This alternative's greatest advantages are availability, reliability, and responsiveness; while its disadvantages are storage costs, update costs, and directory integrity.

3. combinations--many combinations are possible. The most general would permit an arbitrary subset of the directory to exist at each site. The basic advantage of this approach is flexibility; however, this flexibility must be paid for in other ways, e.g., updates, integrity,and consistency. This approach requires that an additional directory or directory of directories be available to facilitate directory updates. This, in turn, resurrects the

problem of where and how to store this new super directory. One spinoff advantage of the super directory is better control over directory updates.

Three basic considerations influence the choice of directory schemes. These are directory retrieval frequency (high frequency encourages redundancy), directory update frequency (high update frequency encourages centralization and non-redundancy); and reliability (encourages redundancy and distribution). [128; see also 31]

The other components of a DBMS, i.e., schemas, subschemas, and the DBMS kernel, can be partitioned and/or replicated as required by user needs and system design requirements. If one DBMS is used throughout the system, then obviously the DBMS kernel would be distributed to every site. Schemas and subschemas also would probably be available to every site, although their content might differ from site to site. In a system where host/back-end configurations are commonplace, the directory and subschemas would be available to the host, while the schema(s) and the DBMS kernel would reside on the back-end. A more detailed treatment of the interrelations of the directories, schemas, subschemas, and DBMS kernels is found in reference 156.

3.3.3 Application Program Distribution.

In attempting to define a methodology for data distribution, early efforts [27,31] assumed that program locations was independent of data location. Morgan and Levin [113] advance the theory, propose a model, and demonstrate that (1) program location and data location are dependent rather than independent, and (2) designs which consider programs to be independent of data location likely will lead to sub-optimal results from a file assignment cost basis.

3.4 DDBMS Problem Areas.

In Chapter 2, the potential advantages/benefits offered by distributed processing systems (extensibility, integrity, performance, and responsiveness) were discussed. Since distributed data base systems are really subsets of the larger domain of distributed processing, these advantages also apply to the DDBMS environment. Although not discussed in Chapter 2, distributed processing systems in general and DDBMS in particular are not without disadvantages and problem areas. Some of these disadvantages and problem

areas were alluded to earlier in this chapter. This section focuses on four major problem areas (concurrency, deadlock, rollback and recovery, and standards) and the efforts that have been directed towards overcoming these problems. (A fifth major issue, data translation, is discussed in Chapter 5.) The first three problem areas, i.e., concurrency, deadlock, and rollback and recovery, although non-trivial, are rather straightforward from a definitional perspective. Standards, the fourth area, encompass a variety of technical/management issues which range from the details of hardware interface to data naming conventions. Since the disadvantages normally associated with DDBMS correspond almost directly to DDBMS problem areas, they will be described first. The primary disadvantages of DDBMSs are fourfold:

1. Control--centralized systems are administratively oriented, while distributed systems are user oriented. [36] Distributed systems require control; however, complexity of scale greatly increases the problems confronting the organization. Instead of relying on a single individual (data base administrator) or a centralized group to administer the system, control in a distributed system must be decentralized. Unfortunately, decentralized control does not necessarily guarantee that the system will satisfy all

user's needs. To guarantee user satisfaction, users must actively participate/cooperate in the definition and maintenance of standards. The same holds true for data maintenance responsibilites. User participation and cooperation are the keys to effective DDBMS operation.

2. Security/Integrity--these issues have not been resolved completely in centralized systems. Unfortunately, distributed systems tend to magnify and exacerbate the problems of preserving data security and integrity.

3. Costs--distributed systems are more costly to administer, operate, and maintain than are centralized systems.

4. Retrieval/updates--in distributed systems, especially those which allow replication of data, retrievals and updates can become extremely complex and require a significant amount of additional software support. This is especially true for split query processing (parallel retrievals from multiple sites based on one user query) and redundant data updates.

3.4.1 Concurrency Control.

Concurrency control or process synchronization is a well-known problem both in the operating system and the DBMS environments. Unlike the operating system environment where

synchronizing mechanisms have been implemented, e.g., monitors in Concurrent PASCAL, researchers [17,18,90,127,130,146,155] are still proposing alternatives to the problem of synchronizing accesses of several processes (DBMS tasks) to shared resources (collection of data). Solutions proposed range from some form of global data base locking mechanism (two phase commit, voting, etc. [154]) to centralized control (primary site updating [146]) to an elaborate set of protocols designed SDD-1 [17,18,127,130]. The underlying assumption of the SDD-1 proposal is that global locking is too inflexible. The SDD-1 designers propose instead that the data base administrator determine by the type of transaction which of the five protocols will be used for each transaction type during the system design phase. While this approach conceivably is faster and cheaper (less communication system overhead) than the global locking approach, it is also inherently inflexible. A thorough review of all approaches reveals that concurrency control is still an open question and that some inflexibility may be necessary to insure that time-dependent errors are eliminated.

3.4.2 Deadlock.

Unlike concurrency control, most of the research in this area has been directed toward the deadlock problem as it is manifested in the operating system environment. As Maryanski [99] points out, the primary difficulty with deadlock in a distributed data base system is the lack of a central control point. Thus, the responsibility for noting either the actual or potential occurrence of a deadlock situation cannot be assigned easily. Detection and prevention are the two basic approaches available. [33] Detection is an after-the-fact determination [105], while prevention requires prior knowledge of all shared record accesses [99]. Both methods are costly; however, Maryanski [99] maintains that prevention is both less costly and potentially less detrimental to overall system performance. The algorithm he developed is based on shared record lists, which contains all shared access records for a set of tasks. The list is maintained by the run-time system. This technique works on the principle that a deadlock-prone state can be avoided, if a requesting task is given control of only a portion of its shared record list. However, "for a distributed DBMS application to operate efficiently under [this] ...prevention algorithm, it is important that the

data base be partitioned into sub-schemas." [99, p. 99] On the surface, the requirement to pre-define all subschemas does not seem to be unusual; however, this requirement also applies to all accesses to include the high-level query language variety. This means that unrestricted access to the data base cannot be allowed. Every access must be through a pre-defined subschema. This, of course, may be a very small price to pay, if deadlock prevention is a necessity. (See reference 108 for one explanation of a deadlock detection methodology.)

3.4.3 Rollback and Recovery.

Although rollback and recovery techniques are fairly well-established in the centralized environment, little research has been devoted to these problems in the DDBMS environment other than Maryanski and Fisher's study [98], which is the most comprehensive to date. The method they propose is a selective recovery algorithm, which rolls back only the failing tasks and those tasks operating with polluted or potentially polluted data. To accomplish selective rollback, Maryanski and Fisher use a "potential shared data list" that is computed from the subschemas of the application tasks. The potential shared data list is

essentially the same mechanism as that used by Maryanski in his deadlock prevention strategy. Maryanski and Fisher's selective recovery approach was designed primarily for a single host/back-end configuration. While it is doubtless applicable to more general types of distributed systems, it cannot handle redundant data. For this reason, the rollback and recovery problem, especially in a generalized DDBMS where redundancy is needed to improve system performance, is still not solved.

3.4.4 Standards.

If distributed data base systems are to become commercially and organizationally feasible, standards must be adopted on two levels--technical and organizational. Some technical standards already exist, e.g., Serial-By-Bit Data Transmission, while others are in the process of being adopted (packet switching). More standards are clearly needed in the areas of hardware interface, network operating systems, interprocess communication interface, and system design. At the organizational level, standards are needed to overcome existing hardware interface and organizational problems. Van Renssalaer [158] lists ten major problem areas that his organization continuously wrestles with in its efforts to design, operate, and maintain an information

system which matches his organization's evolving data management needs. These are:

1. Establishing a central planning and management program for company-wide information system activities so that decentralized development work could be coordinated.

2. Designing systems which could respond easily to constant geographic expansion, organizational change, and the addition of new operating units.

3. Coping with ever-increasing needs for detailed and accurate information to meet management and government reporting requirements while controlling administrative costs.

4. Designing systems which could be adapted to respond to local needs while maintaining company-wide compatibility.

5. Getting user-managers to accept responsibility for the specification and operation of their systems.

6. Convincing users in different functional areas that data is an organizational resource to be shared by all, and that individual transactions should simultaneously update the records of all functions.

7. Avoiding unnecessary duplication of effort in designing and supporting systems.

8. Developing the skills of data processing staff members to meet the needs of a growing organization, and assigning priorities to their activities.

9. Establishing, maintaining, and promoting the use of standards for hardware, software, documentation, project management, auditability, and control as a foundation for well-coordinated worldwide application systems.

10. Controlling security and privacy in an on-line, decentralized, and distributed

multi-national environment. [158, pp. 90-91]

Obviously, all of the problems listed by Van Renssalaer will not apply to every organization; however, many of them will as attested to in references 48, 80, 120, 136, and 167.

3.5 Summary.

This chapter has attempted to provide the reader with an overview of many of the key aspects and issues of DDBMS technology. Three major topics were covered. These were concepts and developments, distributed data base system design considerations, and problem areas. To assist the reader to understand some of the ramifications of this evolving technology, a comprehensive taxonomy or classification scheme was presented. The intent of the preceding short treatise was not to provide an in-depth analysis of all aspects of DDBMS development, but to increase the reader's awareness of the subject matter and to construct a conceptual framework which will be used/assumed throughout the remainder of this report.

CHAPTER 4

THE MILITARY ENVIRONMENT--PRESENT AND FUTURE

4.1 Introduction.

Although it is over 200 years removed from its beginnings, the Military Establishment in the United States today is still shaped by traditions that developed during the Colonial period. Among the most enduring of these traditions are: (1) distrust of large standing armies and professional soldiers, (2) confidence in the nation's ability to raise armies of citizen soldiers to meet national emergencies, and (3) conviction that civilian authority must maintain strict control of military forces. The realities of the post World War II era have tempered somewhat the first two of these. Nonetheless, despite the global commitments of the United States today, the total Military Establishment including reserves and the National Guard represents less than two percent of the population. This chapter focuses briefly on four major topics--post Vietnam developments, intelligence shortfalls, current automated intelligence systems and development, and projected automation developments. Included in the last topic are

descriptions of the Defense Department and North Atlantic Treaty Organization (NATO) military structures as well as the notional strategic intelligence super network, the notional tactical super network, and super network interface requirements.

4.2 Post Vietnam Developments.

Emerging from the Vietnam War, the Army found itself in the following situation:

> The US Army, Europe, was in disarray, rent asunder by its role as part of the rotation base for forces deployed to Vietnam. The Army training base in the Continental United States had concentrated almost exclusively ... on providing units and individual replacements to Vietnam. The Combat development community had concentrated on Vietnam to the exclusion of work to modernize the Army's ability to fight in other theaters. Doctrinal development was still in the mind-set of the 1950's. [145, p. 3]

During the 1970-73 timeframe, the Defense Department in concert with the three Military Departments were faced with the tasks of trying to modernize and restructure the Military Establishment consistent with US national policy. From the Army's standpoint, three basic factors influenced its efforts--lack of a well-articulated national policy,

determination to avoid the "Maginot Line" pitfall, i.e., developing doctrine and training to win the last war, and the grim realization that the Vietnam interlude had severely disrupted the normally constant force modernization process. As Army leaders looked ahead, they saw the possibility of two different types of wars--"...mechanized war - such as we might have to fight in NATO Europe - perhaps even in the Middle East; the other war - a Korea, a Vietnam, a Dominican Republic." [145, p. 2] Each type of war required differing types of forces--mechanized and light infantry. The enunciation of what has been called the Nixon Doctrine in 1972-74 not only reaffirmed US national interest in Western Europe, but also served to narrow the Military Establishment's immediate focus. In Europe, the nation's military faced its toughest challenge. While the US was involved in Vietnam, the Soviets and their Warsaw Pact satellites had quietly and steadily built up both their conventional and nuclear forces. The Soviet Union had also used this time to expand its sphere of influence. Nowhere was this more apparent than in the Middle East and North Africa. Just as the Army staff was beginning to come to grips with how to fight a future European land war, an event occurred which had a profound effect on their thinking. The October 1973 Arab-Israeli War, perhaps the most studied war

in recent history, gave the world a brief glimpse of the lethality of modern mechanized warfare. The picture it painted in diminutive was a battlefield where highly mobile forces using massive armor thrusts engaged in almost continuous day and night operations and where electronic jamming, terrain obstacles, mines, smoke, and anti-aircraft weapons were used effectively as combat multipliers. The losses on both sides were staggering. In the aftermath of this war, political and military leaders in the United States determined that the US and its allies could no longer afford to be caught unprepared as the Israeli's had been in October 1973. The lethality of the modern battlefield as demonstrated in the October War dictated that the US and its NATO allies either must win the first and last battles of the war or lose Western Europe. [145,147] Yet, the political, economic, and social climate of the early 1970's was such that the US could neither afford nor would the American public support a large peacetime Military Establishment. To fight outnumbered (3 or 4 to 1) and win, basic changes had to be made in the way US forces normally were employed, trained, and equipped. First, the Army needed new tactics and operational concepts. These were supplied in 1976, when the Army published its first comprehensive tactical doctrine manual, FM 100-5 (Operations). Second, tactics had to become the driving

force behind the development of force structure, training, and equipment. The US could no longer afford the luxury of learning how to fight after the war started. It had to be ready before the war started. [145] Although there has been some reluctance, these radical changes are now shaping the Army of today as well as the Army of the future.

Despite the developments discussed above and the fact that the US is currently spending billions of dollars to modernized both it nuclear and conventional warfare capabilities, optimistic estimates hold that a totally modernized force will not be available until the mid-1980's. Even when a totally modernized conventional force is available, it still must be able to fight outnumbered and win. The operational concept that is designed to accomplish this feat is known as the "Active Defense". The key features of this concept, which was first elucidated in FM 100-5, are:

> See deep to find the [enemy's] following echelon, move fast to concentrate forces, strike quickly to attack before the enemy can break the defense and finish the fight before the second echelon closes....[145, p. 7]

Although easily described, the active defense is not a

simple undertaking. It requires centralized planning, a mixture of centralized and decentralized control, and decentralized drill-like execution. [147] If it is to work, commanders at all levels not only have to know the capabilities of their own forces, but also the capabilities and intentions of the enemy force they face. Only with this information and a prodigious amount of training can Army leaders make the type of split second decisions that the active defense demands.

4.3 Intelligence Shortfalls.

To obtain information about enemy forces, commanders have to rely on a mixture of organic and non-organic surveillance, reconnaissance, target acquisition, and intelligence analysis resources. Applying the lessons learned in Vietnam to the projected European battlefield, the Army determined that two intelligence deficiencies had to be overcome. First, intelligence analysis and dissemination had to be improved. Intelligence analysis simply had not kept pace with the development and fielding of myriad sophisticated collection devices and systems. In Vietnam, the Army had faced the problem of locating/detecting an often illusive enemy. Target

acquisition, reconnaissance, and surveillance accordingly were emphasized. The result of this emphasis was a wide-ranging suite of sophisticated sensors of almost every size and shape from passive night observation devices to unattended ground sensors to complex air breathing and non-air breathing aerial platform mounted sensor packages. By the end of the Vietnam War, the Military Establishment's information collection capabilities had far outstripped its capacity to manage them or to analyze and use all the information provided. Yet, little was done during the war to correct the imbalance. As in World War II, charts, maps, overlays, and grease pencils were still the intelligence analyst's primary tools. When the focus shifted to Europe, it became apparent that the "stubby pencil" or stubby grease pencil approach to collection resource management and intelligence analysis was no longer adequate. Unlike Vietnam, a war in Europe will be a target-rich environment. (The authors of FM 100-5 estimated that each US battalion might have to destroy as many as 250 targets in the space of 10 minutes to blunt a main enemy attack in Europe. [145]) Since 1974, a number of major changes have been made in the organization, structure, and methodology for managing intelligence collection resources and for analyzing and disseminating intelligence. Although intelligence analysis

is still more an art than a science, the Army has recognized the need for better communication systems to disseminate intelligence and targeting information and for automated support to help analyst winnow out the wheat from the chaff. Developments in the latter area are discussed in more depth in Section 4.3.

The second major deficiency that had to be overcome was the lack of an effective interface between tactical intelligence collectors, producers, and consumers and their strategic intelligence counterparts. Throughout the Military Establishment, this is commonly referred to as the "Tactical/Strategic Interface Problem". The establishment of the Central Intelligence Agency (CIA) in 1947 marks the beginning of the first permanent strategic intelligence effort in the history of the United States. The CIA in conjunction with the National Security Agency, the State Department, the Treasury Department, the Energy Department, the Federal Bureau of Investigation, and the Defense Department (the strategic intelligence resources of the three Military Departments are included under the Defense Department) are responsible for providing strategic intelligence to the National Command Authority (the President and the National Security Council). Prior to 1973, the efforts of most of these agencies were

uncoordinated, there was much duplication of coverage and reporting, and little of the intelligence produced was ever provided directly to operational force commanders. Since then, the strategic intelligence producers (particularly the CIA and the Defense Department, which has the largest portion of the national intelligence budget) have increasingly come under fire for their excesses and failures from the Congressional arena. This has resulted in a number of changes--the national intelligence effort has been streamlined, centralized budgeting has been imposed (the Director of Central Intelligence is now responsible for formulating the budget for all intelligence agencies), a Policy Review Committee (Intelligence) of the National Security Council has been formed to establish requirements and priorities for national intelligence, restrictions on the dissemination of intelligence have been reduced, and support of tactical forces has been and is still being emphasized. [41] Despite these changes, especially the increased support to tactical commanders, and the increased emphasis within the Department of Defense (DoD) on command, control, communications, and intelligence (C3I), the practical side of the tactical/strategic intelligence interface problem is still far from being solved. Efforts toward a solution are discussed in the following section.

4.4 Current Automated Systems/Development Efforts.

In the previous section, two intelligence deficiencies were discussed--tactical/strategic intelligence interface and intelligence analysis. This section focuses on the automation efforts within DoD that are aimed toward overcoming some aspects of these deficiencies.

4.4.1 Strategic Systems.

Unlike their tactical intelligence counterparts, strategic intelligence producers have long maintained their intelligence holdings in automated files/data bases. In the late 1960's, a number of these systems were interconnected into a loosely federated network known as the Community On-Line Intelligence Networking System (COINS). Initially, this system was confined to users located in the National Capitol Region. Later, however, access to COINS was extended to the European and Pacific Commands through the Defense Intelligence Agency's On-Line System (DIAOLS). (Figure 4-1 is a graphic representation of these systems.)

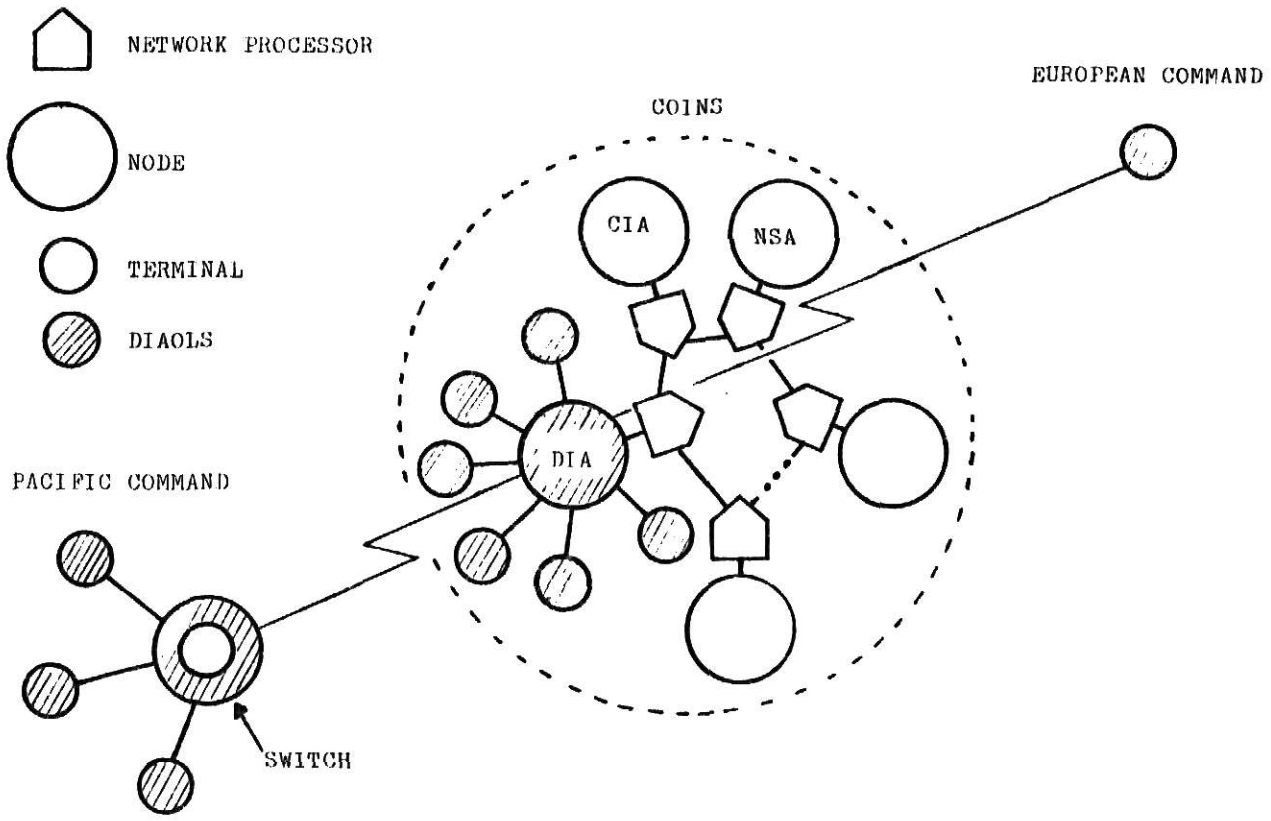


Figure 4-1: DIAOLS and COINS Systems

The drawbacks to the DIAOLS and COINS systems are fourfold. First, only certain limited intelligence information was ever placed on these systems. Second, to use either system, an intelligence analyst must know either the COINS access language or the DIAOLS and COINS access languages. Third, reponse time is exceedingly slow especially on COINS. The reason for this is that COINS, despite its name, is really a batch-oriented network. Terminal commands, in effect, spawn batch jobs. Remote users, who are assigned the lowest priority, can expect to wait anywhere for 1 to 2 hours to days for response to a query. Fourth, if a computer in the network fails for any reason, the information stored at that site is lost to the network.

In 1973, the Army, which had lagged behind the other departments in the development of networks, initiated its first experiment in distributed processing. The system, which is still not completely operational, is called ASSIST (Army Systems for Standard Intelligence Support Terminals). Although it was initiated under duress (the Assistant Chief of Staff for Management Information Systems, Department of the Army, was threatening to take away all the IBM 360 systems dedicated to intelligence support for use as base support systems), the project soon evolved. By 1974, ASSIST

was being tauted as the saviour of the Army Intelligence community. In the Functional Description [1] written for the ASSIST Conus Testbed System (ACTS), such praiseworthy phrases as "complete user transparency," "worldwide distributed data base," and "reliability/survivability" were used often. The system as envisioned by its designers (See Figure 4-2) was to provide:

1. analysts with access to local and remote intelligence files as well as interaction with remote analysts;
2. a flexible architecture to accomodate user needs through standardized hardware and modular software;
3. multilevel security;
4. reduced people costs.

To accomplish these goals, the system as shown in Figure 4-3 was designed to take advantage of the processing capabilities of a large scale Host computer and a small, yet powerful multiprogrammed minicomputer. The totality of the data base was to reside on the Host system. Each night subsets of the data base were to be transmitted to the various Intelligence Support Terminal (IST) locations after batch updates to the data bases had been received and

processed from the ISTs. A data management system (DMS) was installed on the ISTs to support user interaction with locally resident data. In the Functional Description, access to the data base whether resident on the Host or the IST was to be completely transparent to the user.

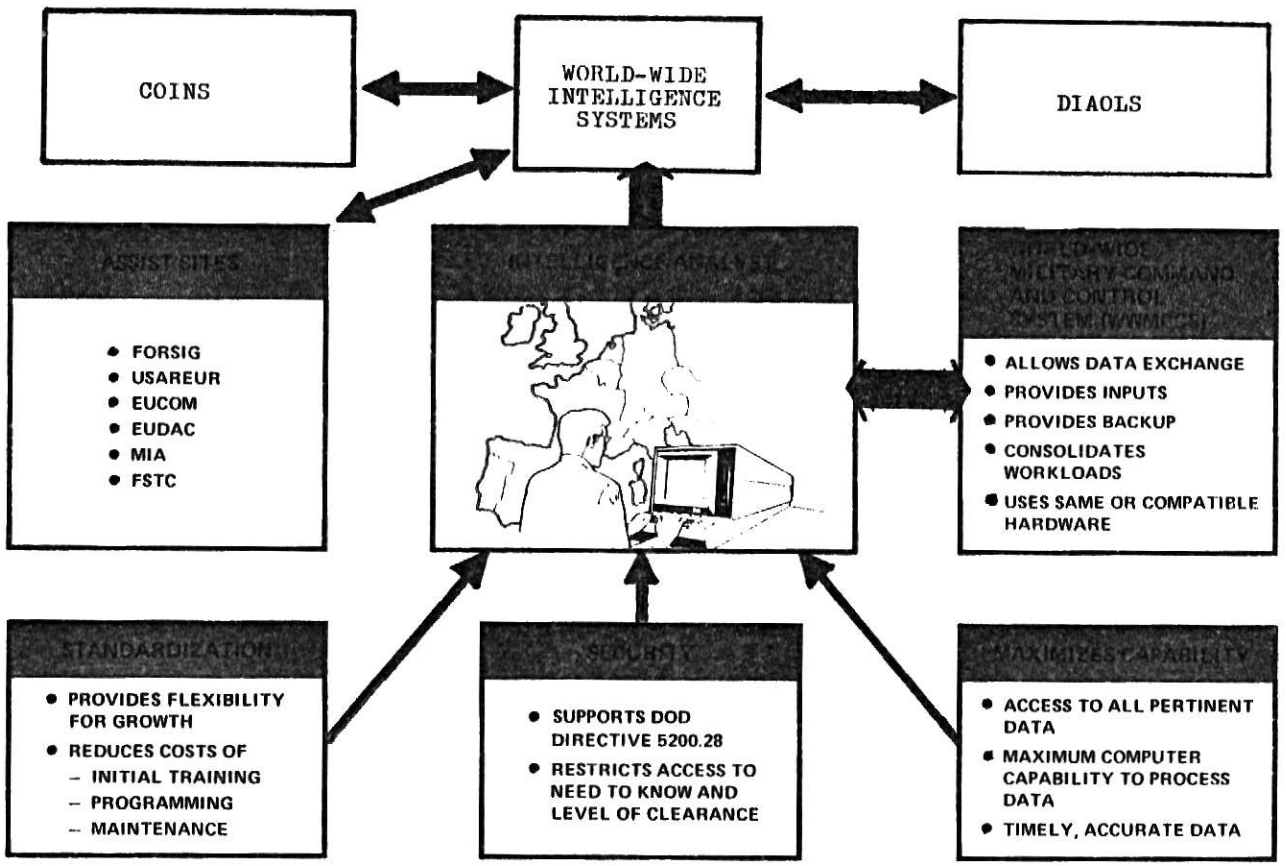


Figure 4-2: ASSIST Design

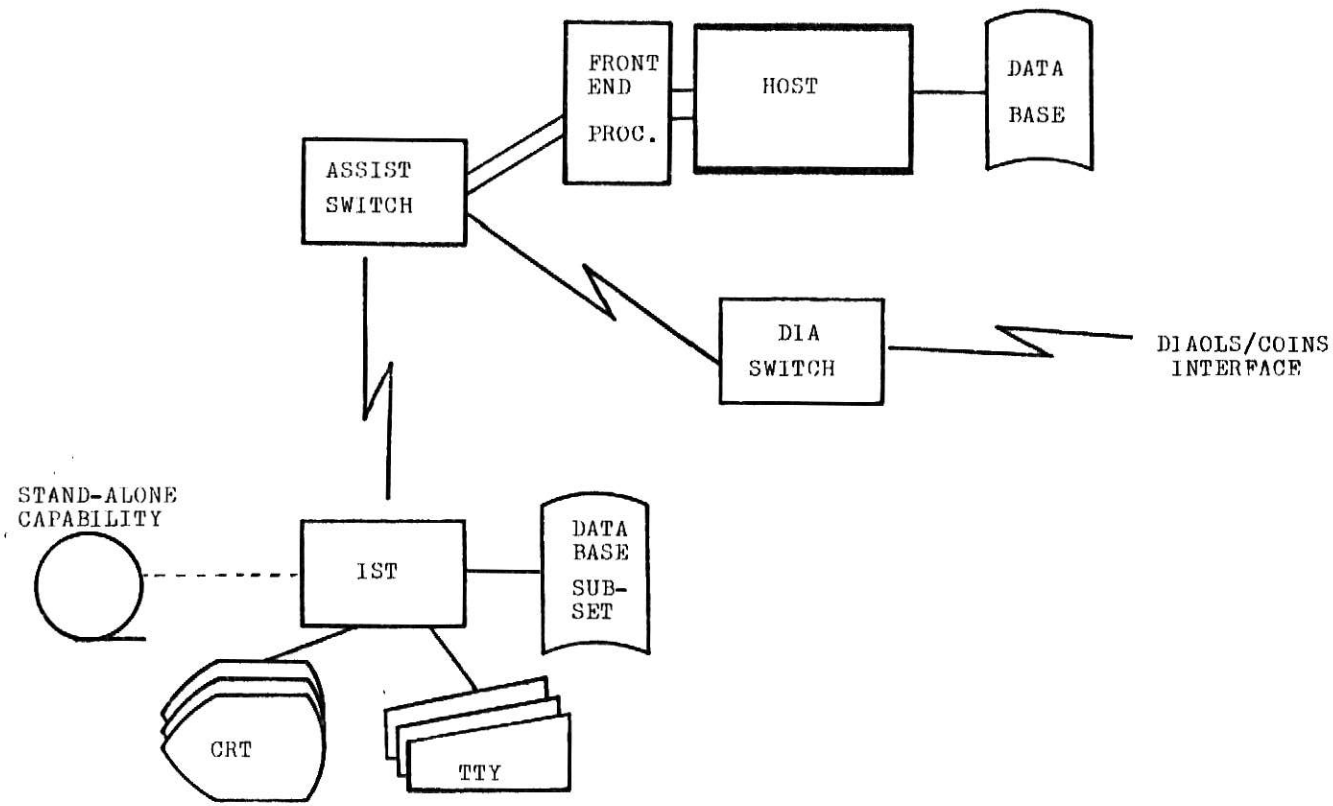


Figure 4-3: ASSIST System Configuration

As implemented, the system satisfied few, if any, of the larger system goals completely. The technology simply was not available. During the testbed system evaluation, the following major problems were found:

1. The system would not support its users easily (analysts had to know four different query languages to access data--IST DMS, Host DMS, DIAOLS, and COINS).

2. The IST DMS did not support frequent loading (small data bases sometimes took 12 hours to load).

3. The IST DMS did not support retrieval of transactions for Host batch updates.

4. The 9.6 kilobit communication link between the IST and the Host and the communications software did not support the movement of bulk data as anticipated.

Despite its many shortcoming, the ASSIST project did focus attention on the need for standardization. Additionally, the IST portion of the system worked fairly well in a stand-alone mode provided the data base was periodically compressed and updates were limited. This plus the fact that all the software was either government owned or had limited proprietary restrictions allowed to the system to be used as a preliminary baseline for the development of specifications and requirements for automated tactical

intelligence systems. (A system somewhat similar to ASSIST has been implemented in Europe. This system, which is known as EUCOM AIDES (US European Command Analyst Interactive Display and Exploitation System), is also tied to the DIAOLS and COINS systems via the DIA switch shown in Figure 4-2.)

The more or less surprise invasion of Czechoslovakia by Soviet troops in 1968 highlighted a serious deficiency in the US national intelligence system--lack of an effective indications and warning (I&W) system. This has been remedied in part by the development of a Worldwide I&W System, whose purpose is to provide warning of impending hostilities or other activities affecting US national interests. The automated system that supports the I&W effort is currently being upgraded. [41] Eventually, this system will consist of a network of remote minicomputers tied to a multi-minicomputer system located in the National Military Intelligence Center (NMIC). The NMIC system is shown in Figure 4-4.

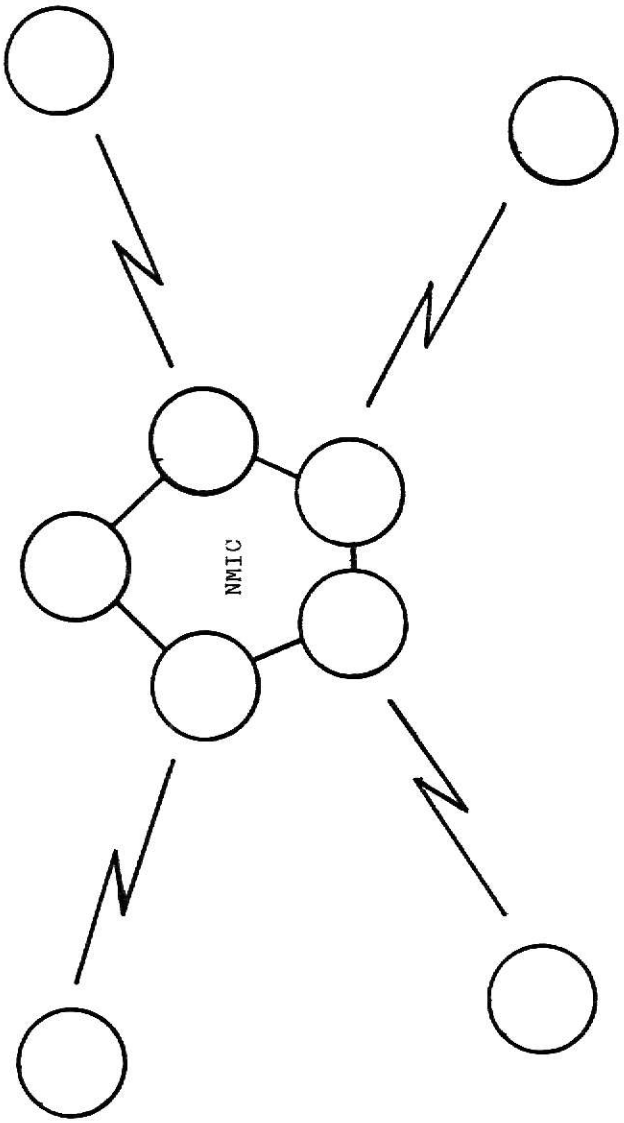
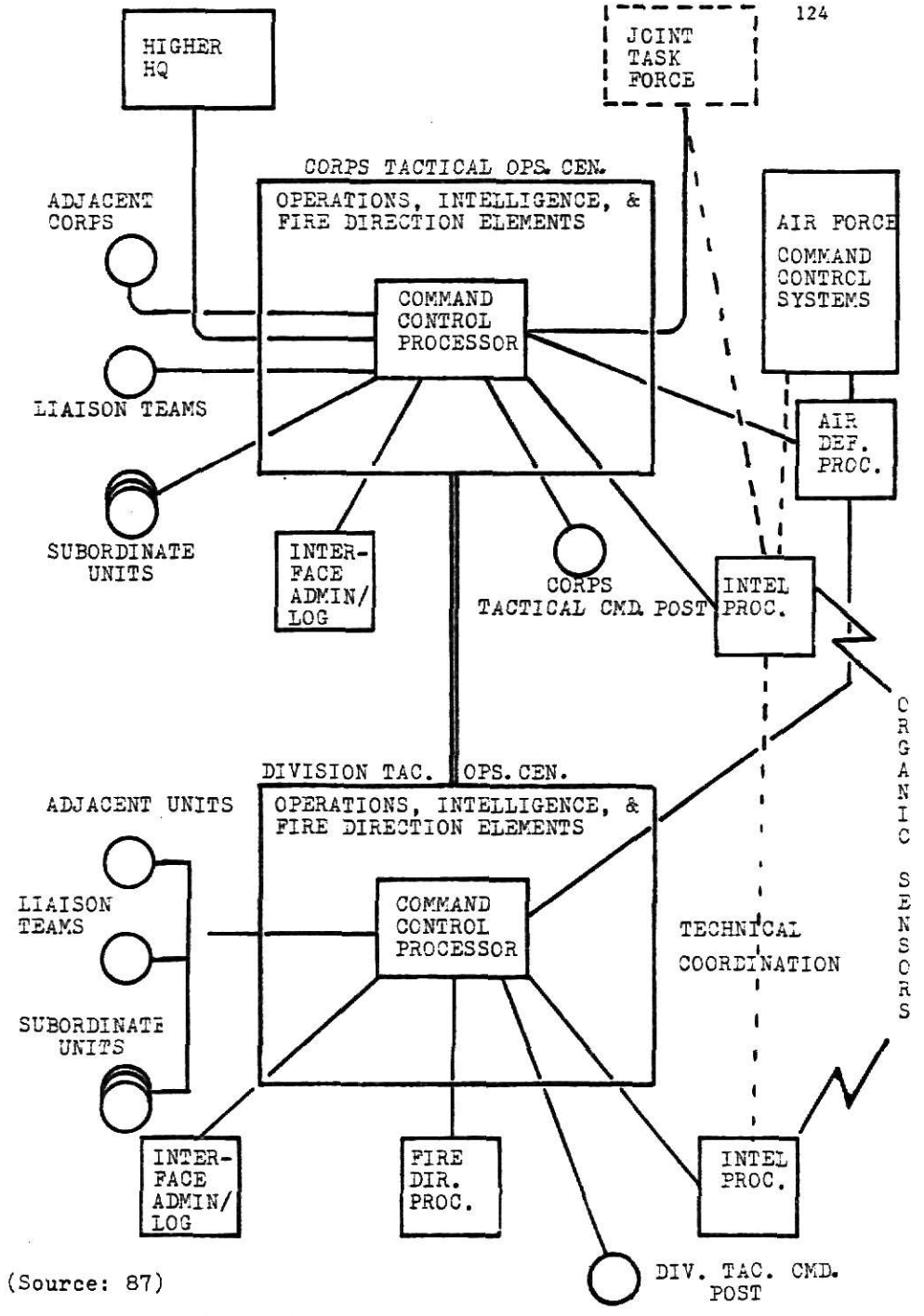


Figure 4-4: National Military Intelligence Center

4.4.2 Tactical Systems.

Until recently development of automated systems to support tactical users' requirements have been largely uncoordinated. This has resulted in the development or proposed development of over 70 separate, often incompatible systems to support various functional areas, e.g., intelligence, command and control (C2), fire direction, air defense, and administration/logistics. In 1977-78, the Combined Arms Combat Development Activity (CACDA) at Fort Leavenworth initiated a major study of the various systems. Based on the results of this study, a set of system interoperability interface criteria and an architecture for an integrated battlefield Command, Control, Communications and Intelligence (C3I) have emerged. [68,87,148] As described by Major General Mahaffey, the objectives of the latter are simply to provide "...an integrated, command, control, communications, and intelligence system capable of supporting the tactical commander in combat, crisis or peacetime. [87, p. 28] The integrated approach that Major General Mahaffey refers to is not a centralized system, but rather a fully distributed system with a system executive architecture and a data distribution (communication) system that ties the various subsystems together to form an integrated system (Figure 4-5).



(Source: 87)

Figure 4-5: Integrated C3I System

While automated developments in all the functional areas including communications are interesting, intelligence is perhaps the most diverse functional area in the proposed tactical integrated C3I system. The reason for this is that the intelligence subsystem must not only operate within the integrated C3I framework, but it must also be able to "...relate directly to the various tactical echelons and connect with other services and mulitnational sensors and sources." [87, p. 27] The requirements placed on the intelligence subsystem (human and machine) include:

1. management of collection assets;
2. analysis of all-source information for targeting and predictive intelligence;
3. dissemination of intelligence and targeting information to higher, lower, and adjacent units.

Using modeling techniques, intelligence subsystem designers have gained a number of insights into "...those functions that machines do best to help man do the job he does best." [148, p. 36] First, it has been determined that automated techniques have great potential in the collection management area. Second, machines can help to filter incoming data (separate wheat from chaff). Third, machines can correlate the filtered data with the existing data

base. At this point, however, the human filter must be applied, because no matter how effective an automated system is it does not have the intuitive capacity of the well-trained intelligence analyst. As Brigidier General (now Major General) Stubblebine pointed out in a recent article,

> The most important caution with respect to models (and automation) is to beware the tendency to treat them as the intelligence panacea, the answer to all of our problems. Automation systems, computers, models are but another tool of the soldier--like his helmet, tank, or helicopter...automation will help him do his job, but it will not do it for him. [148, p. 36]

The most optimistic estimates are that at least a rudimentary integrated C3I system will be fielded by 1985. This, of course, assumes that a tactical communications system capable of providing data communication support can also be fielded. As so aptly stated by Major General Hilsman, Commander of the US Army Signal Center and School:

> The bottom line is this: the command, control and intelligence systems...are only effective if there is a communications system to support them. [68, p. 32]

4.5 Projected Automation Developments.

In the Department of Defense Annual Report for Fiscal Year 1979 [41], Secretary of Defense Harold Brown outlined the Defense Department goals for the development of effective C3I systems (both tactical and strategic). These included:

1. C3I systems that can operate without interruption during peacetime, the transition from peace to war, and wartime in all anticipated environments.
2. C3I systems that can survive nuclear attack, help to restructure forces after an attack, and support a variety of responses to such an attack.
3. C3I systems that flexible enough to allow for modular growth.
4. C3I systems that are completely interoperable with the C3I systems employed by allies.
5. C3I systems that provide the capability to identify and control friendly forces as well as identify enemy forces.
6. C3I systems that can increase the probability of acquiring and recognizing early indicators of attack.
7. C3I systems that provide direct and timely

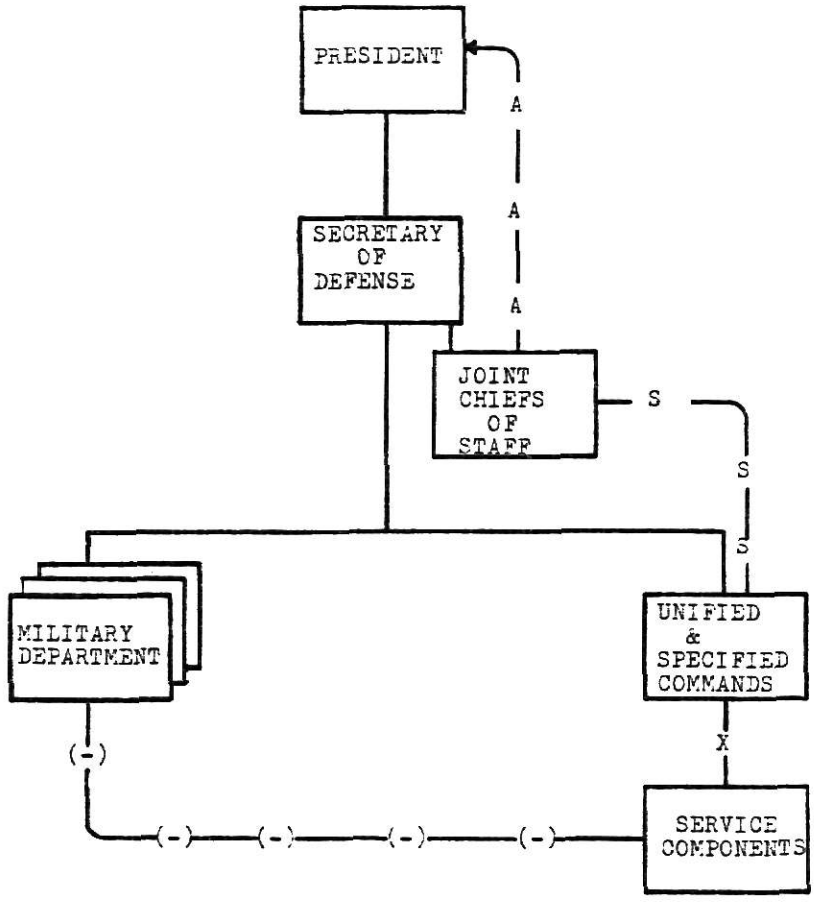
information to tactical commanders.

8. C3I systems that are rapidly deployable to support crisis management and to reconstitute destroyed facilities.

Given the goals just enumerated, it is probable that the strategically oriented intelligence systems discussed previously as well as others not discussed will be integrated to form a fully distributed heterogeneous network of networks or super network by the mid-to-late 1980's. Before discussing one possible notional super network configuration, the Defense Department and the NATO military structures will be outlined briefly to provide a perspective on how a super network system would relate to the organizations it must support.

The Secretary of Defense (SECDEF) is, by statute, the principal assistant to the President for all matters pertaining to the Department of Defense (DoD). The SECDEF has full "...direction, authority, and control...." over the DoD subject to Presidential restrictions. [154, p. 2-9] Within DoD, there are two distinct chains of command--one for operations and another for administrative matters (See Figure 4-6). For operational direction, the chain of command flows from the President to the SECDEF through the

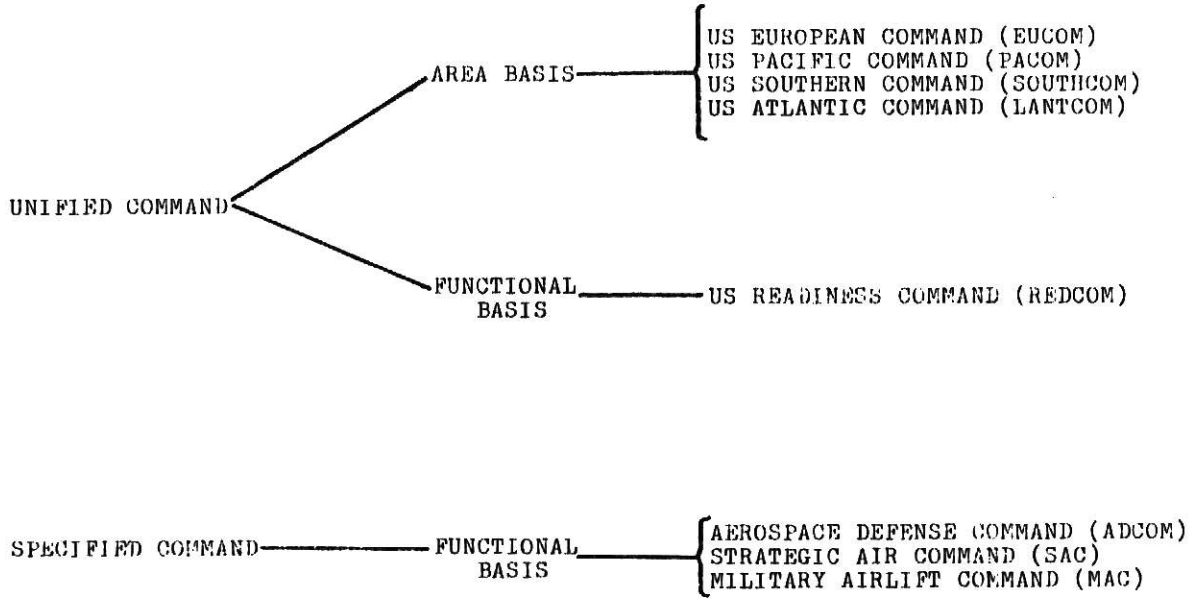
Joint Chiefs of Staff (JCS) to the operational forces. The JCS are, by statute, the principal military advisors to the President, the National Security Council, and the SECDEF. They do not command forces per se; instead, they serve as the military staff in the chain of operational command issuing orders and providing strategic and operational directions in the name of the President or the SECDEF to the commanders of Unified and Specified (U&S) Commands (Figure 4-7). Commanders of U&S Commands are responsible directly to the SECDEF and the President for mission accomplishment. (Operational command includes four elements of authority: (1) composition of subordinate forces, (2) assignment of tasks, (3) designation of objectives, and (4) authoritative direction necessary to accomplish the mission. [157])



X Operational Command
(-) Command less operational control
A Advisors
S Strategic & Operational Direction

(SOURCE: 157)

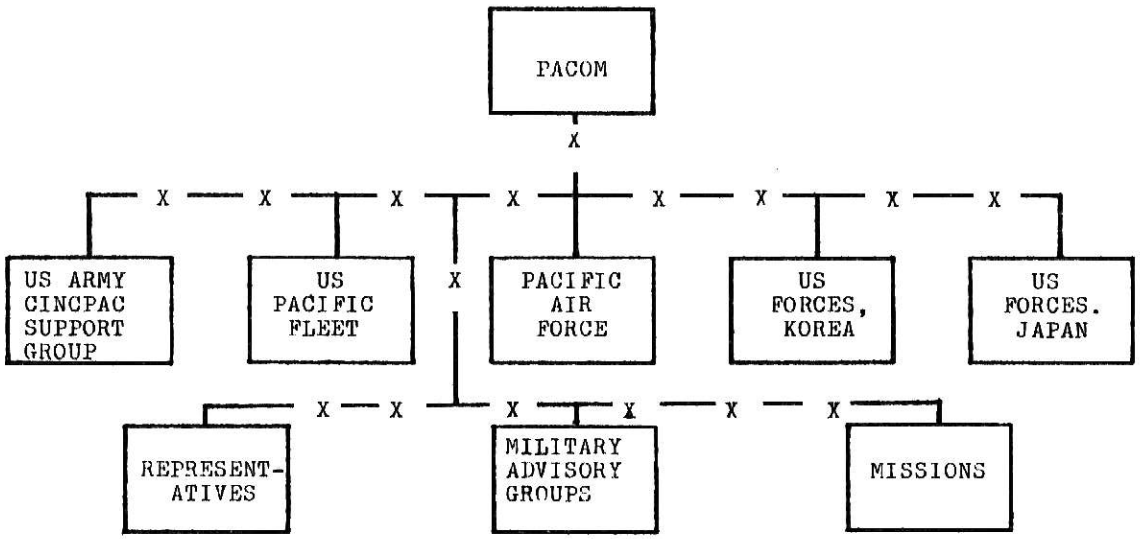
Figure 4-6: Chains of Command



(Source: 157)

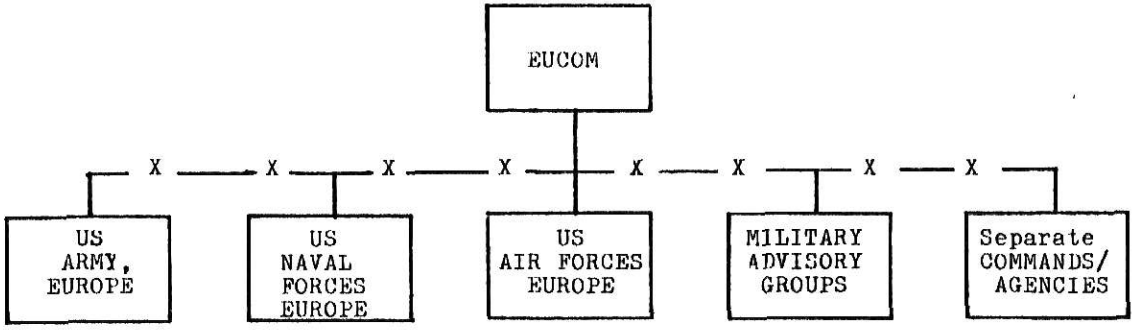
Figure 4-7: Major Active Joint Commands

Unified Commands, which are established and designated by the President, have a broad and continuing mission. A Unified Command is normally composed of significant assigned components of two or more services (see Figures 4-8 and 4-9). A Specified Command is similar to a Unified Command in many respects. It also is established and designated by the President, and the commander of a Specified Command is responsible to the SECDEF and the President for mission accomplishment. Unlike a Unified Command, a Specified Command is primarily a uni-service command, i.e., it is normally composed of forces from only one service--usually the Air Force. [157] Figures 4-10 and 4-11 show command relationships and summarize the basic characteristics of the U&S Commands as well as a third type of joint command, the Joint Task Force (JTF). Unlike the U&S Commands, the JTF is formed usually to accomplish a short duration mission having a specified limited objective.



(Source: 157)

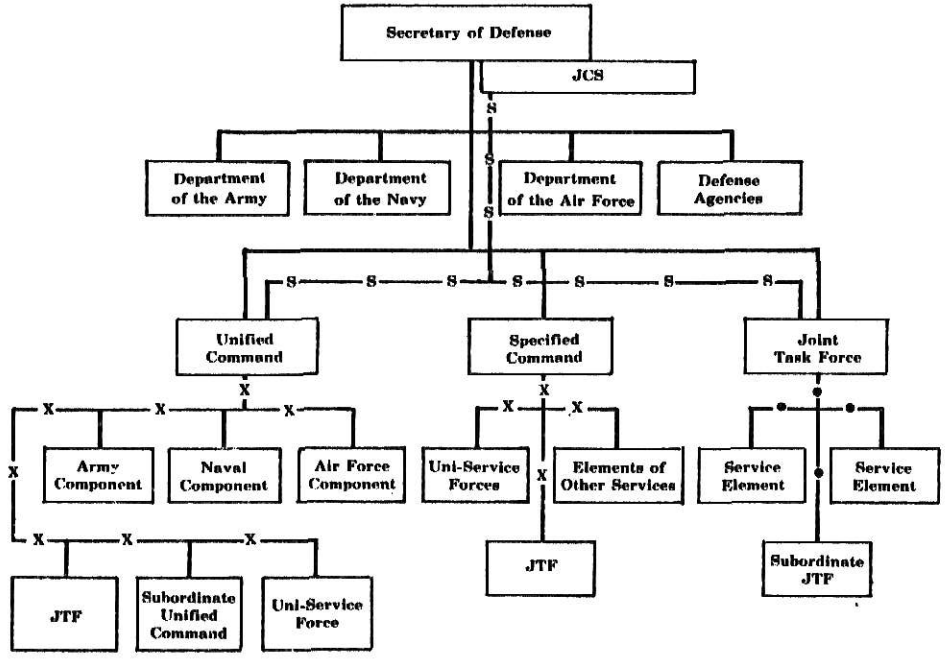
Figure 4-8: The Pacific Command



——X—— Operational control

(Source: 157)

Figure 4-9: The US European Command



(Source: 157)

Figure 4-10: Joint Forces Command Structures

	Unified command	*Specified command*	*Joint task force*
Establishing	President	President	SECDEF; commander unified command; commander specified command; commander existing JTF.
Purpose	To accomplish a broad, continuing mission by forces of two or more Services— In large-scale strategic operations; *or* In a large area requiring single responsibility for coordination of unified operations therein; *and* Requiring centralized direction over all assigned resources.	To accomplish a broad, continuing mission by a uni-Service force— To insure freedom of action without necessity for close coordination of operations which cross normal operational boundaries; *or* To provide strategic direction in an area used primarily by uni-Service forces.	*To accomplish* a specific tactical mission of limited objective and short duration by forces of two or more Services— *which* Requires close integration of tactical efforts; *or* Requires coordination of joint actions within a tactical area; *and* Does not require centralized direction of logistics.
Scope	Area or functional.	Area or functional.	Limited tactical tasks.
Force composition	Two or more Services.	Uni-Service, but may include units of other Services for specific purposes or specific time.	Two or more Services.
Staff	Joint.	Uni-Service, may include representation from other Services.	Augmented uni-Service or joint staff.
Authority	Operational command.	Operational command.	Operational control.
Logistic authority	*Directive authority.*	Directive authority.	Coordination as essential to accomplishment of *mission.*

(Source: 157)

Figure 4-11: Joint Force Characteristics

The other chain of command within DoD, the administrative chain, is responsible for all aspects of command not specifically reserved to the operational chain. This chain of command runs from the President to the SECDEF to the Secretaries of the three Military Departments. The Military Departments are responsible for providing forces for assignment to the U&S Commands (the forces so assigned are known as service components) and administrative support of their forces wherever employed. [154] The Secretary of the Army, for example, exercises administrative command through Headquarters, Department of the Army over the commands and field operating agencies listed in Figure 4-12.

MAJOR ARMY COMMANDS IN CONTINENTAL UNITED STATES (CONUS)

US ARMY FORCES COMMAND (FORSCOM)

US ARMY TRAINING AND DOCTRINE COMMAND (TRADOC)

US ARMY MATERIEL DEVELOPMENT AND READINESS COMMAND (DARCOM)

US ARMY COMMUNICATIONS COMMAND (USACC)

US ARMY HEALTH SERVICES COMMAND

US ARMY INTELLIGENCE AND SECURITY COMMAND (INSCOM)

US ARMY MILITARY DISTRICT OF WASHINGTON (MDW)

US ARMY CRIMINAL INVESTIGATION COMMAND (CIDC)

MAJOR FIELD OPERATING AGENCIES OF THE ARMY STAFF

US ARMY RECRUITING COMMAND (USAREC)

US ARMY MILITARY ENLISTMENT PROCESSING COMMAND (MEPCOM)

US ARMY COMPUTER SYSTEMS COMMAND (USACSC)

US ARMY MILITARY ACADEMY (USMA)

US ARMY WAR COLLEGE (USAWC)

US ARMY MILITARY PERSONNEL CENTER (MILPERCEN)

US ARMY RESERVE COMPONENTS PERSONNEL AND ADMINISTRATION CENTER (RCPAC)

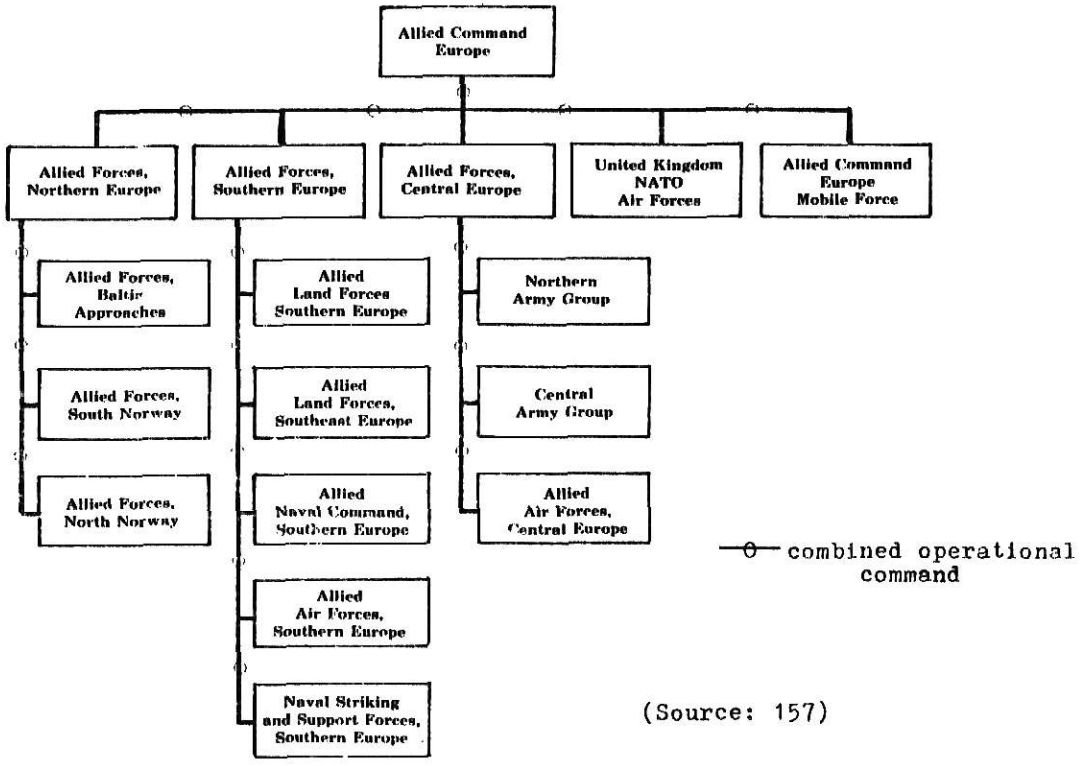
US ARMY ADJUTANT GENERAL's CENTER (TAGCEN)

(Source: 154)

Figure 4-12: Major Commands and Management Agencies

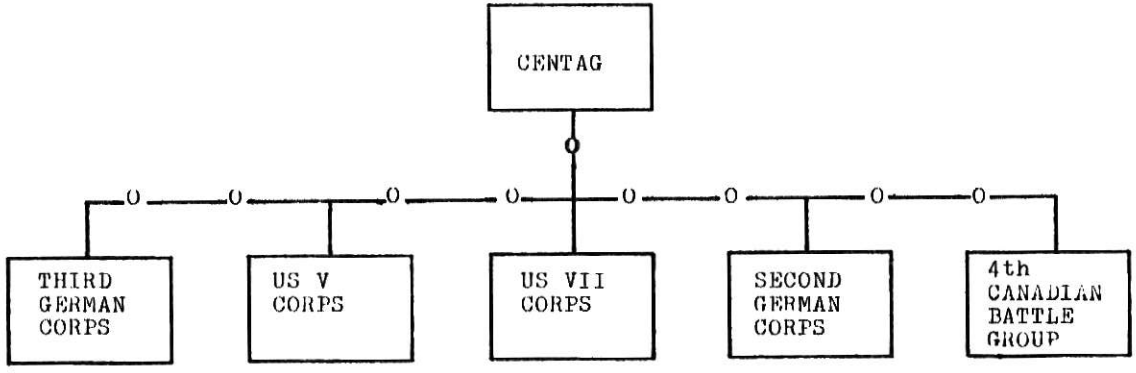
In wartime, US forces can also be expected to operate as part of a combined force, i.e., the forces of two or more nations operating together to achieve a common objective. [154] Of all the treaties and international agreements to which the US is a signatory, the North Atlantic Treaty Organization is the only one that has in being a large, active, combined military organization (Figure 4-13). Since the bulk of US forces earmarked for NATO are located in the Allied Forces, Central Europe (AFCENT) sector, only this portion of Allied Command Europe (ACE) will be described. Forces of seven nations are are subordinate to AFCENT during wartime--Belgium, Canada, Germany, Luxembourg, the Netherlands, the United Kingdom, and the United States. The Commander-in-Chief, AFCENT (CINCENT) is a German general officer. His mission is to defend the central region. There are three commands subordinate to AFCENT--Northern Army Group (NORTHAG), Central Army Group (CENTAG), and Allied Air Forces, Central Europe (AAFCE). The Commander-in-Chief, US Army Europe is also the Commander-in-Chief of CENTAG. In peacetime, CENTAG is primarily a planning headquarters, because its wartime tactical components remain under national control except for exercises, manuevers, emergencies, or war. When any of these conditions exist, CENTAG exercises combined operational control (Figure 4-14) over the following

in-theater tactical forces--the Third German Corps (three divisions), the US V Corps (two divisions and one armored cavalry regiment), the US VII Corps (two and one-third divisions and one armored cavalry regiment), the Second German Corps (four divisions), and the 4th Canadian Battle Group (Mechanized). Other tactical forces from the Continental United States (CONUS), e.g., the 1st Infantry Division (Mechanized), would be deployed to Europe just before or immediately after the initiation of hostilities. [157]



(Source: 157)

Figure 4-13: Allied Command Europe



(Source: 157)

Figure 4-14: NATO Central Army Group

4.5.1 Notional Strategic Intelligence Super Network.

At the highest level, a fully distributed super network to support strategic intelligence collection, analysis, and dissemination for the US in peacetime and for NATO in wartime might exist as shown in Figure 4-15. ((Although the strategically oriented Worldwide Military Command and Control System (WWMCCS) is shown as an interface to the notional super network, the WWMCCS functions could either be incorporated directly into the super network system or the major WWMCCS nodes, which are located at or near each of the major super network clusters, could interface directly with each cluster, e.g., EUCOM, PACOM, etc.)) The entire super network structure may be viewed as consisting of three levels (Figure 4-16). At level 2, each major cluster may consist of either a single interconnected node (SOUTHCOM), an integrated replicated vertical or mixed network/super network (EUCOM), or a federated replicated mixed network/super network (National Capitol Region). The communication system that will link the various major clusters together at this level is AUTODIN II (Automatic Digital Network). AUTODIN II, a DoD system, is a packet switched network, which is being designed to use a variety of media, e.g., fiber optics, land lines, microwave, satellite, and laser. (An article in the November 1979

issue of Popular Science titled, "Lasercom Speeds Unjammable Messages Through Space," [109] discusses the testing of a system being developed for military communication usage that consists of ground based fixed and mobile site communication facilities that employ lasers to transmit data via satellites to other ground stations throughout the world. The article states that the system is expected to be fielded in 1986.)

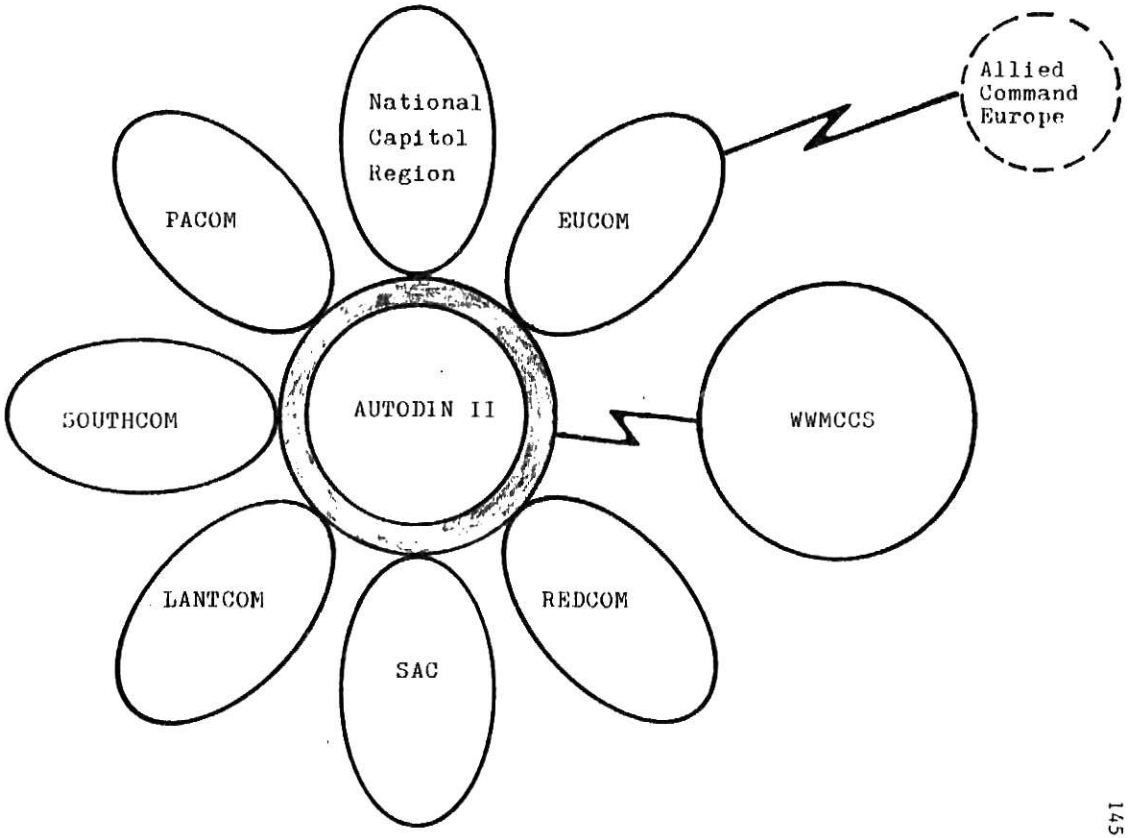


Figure 4-15: Notional Strategic Intelligence
Super Network

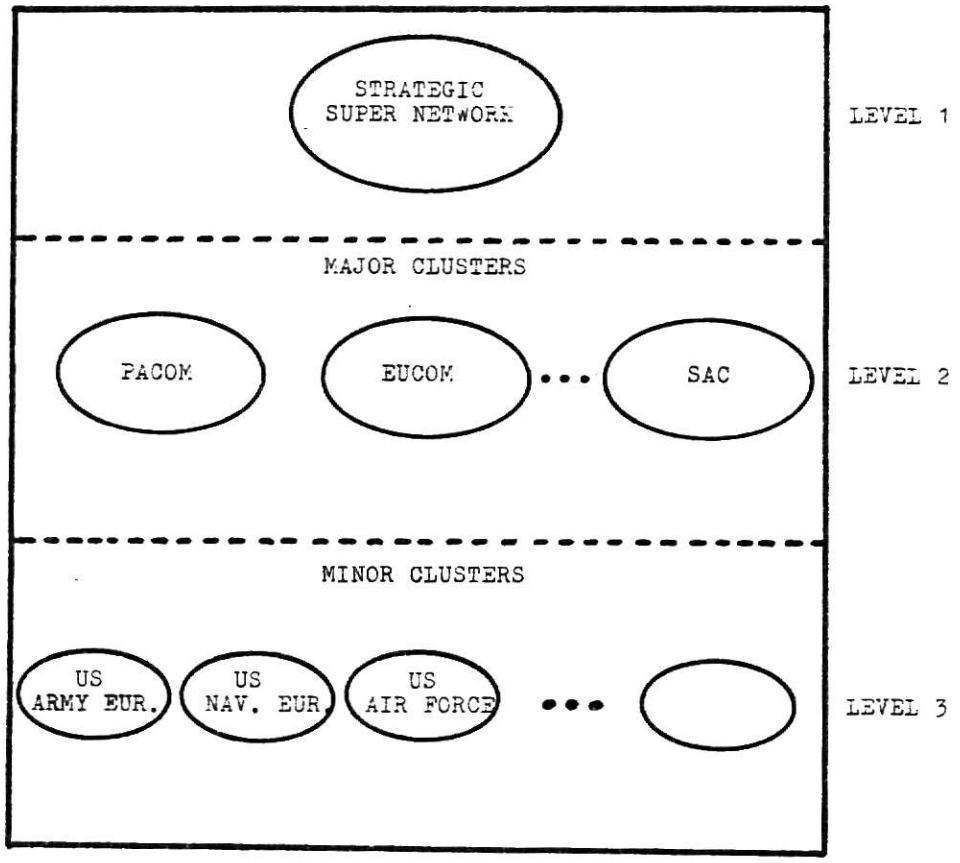
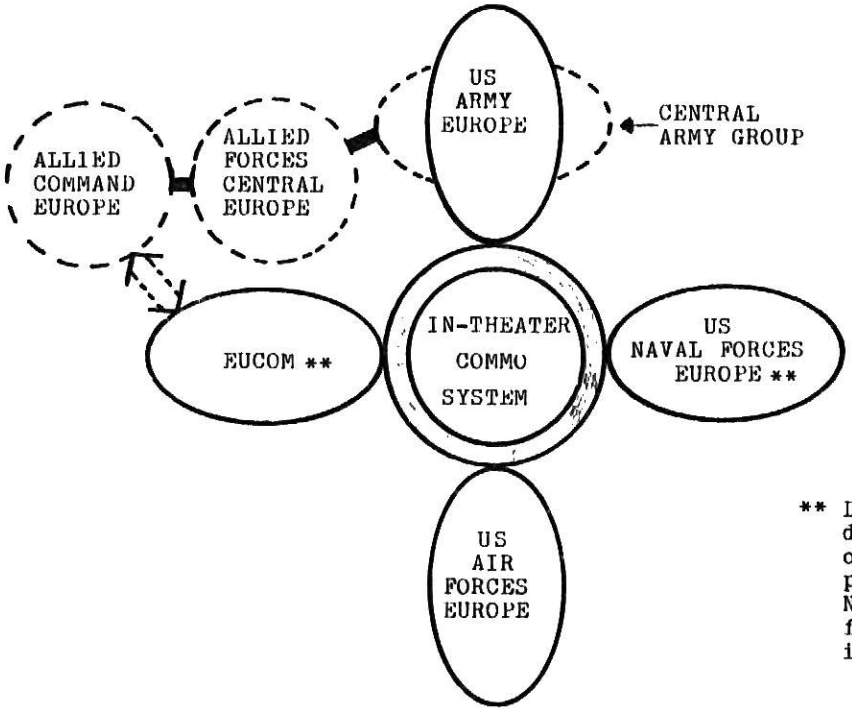


Figure 4-16: Strategic Intelligence
Super Network Levels

As shown in Figure 4-16, each major node of the super network may consist of one or more minor nodal clusters. For example, the EUCOM major cluster might be composed of four minor clusters (Figure 4-17). Since one of the major DoD C3I system goals listed earlier call for US C3I systems to be interoperable with allied C3I systems, it is probable that Allied Command Europe (ACE) would interface the US strategic intelligence system at the major nodal level as shown in Figure 4-17. If the C3I systems of both the US and its NATO allies are truly interoperable, such an interface is technically feasible. However, several problems may have to be solved before a non-US controlled system is allowed access to the entire US strategic system. Since the details of such an interface and the restrictions, if any, that might be imposed upon it are beyond the scope of this report, they will not be discussed further. The third level of the strategic intelligence super network structure (Figure 4-16) represents the juncture of the strategic and the tactical systems; therefore, discussion of this level will be deferred until Section 4.5.3 Interface Requirements.



** Due to balance of payment deficits, major portions of strategic intelligence production effort for US Naval Forces Europe and for US EUCOM are accomplished in CONUS.

Figure 4-17: European Command Major Cluster

4.5.2 Notional Tactical Super Networks.

Unlike the notional strategic intelligence super network, the tactical super networks probably will be composed of systems designed to support more than one functional area (Figure 4-5). Based on Major General Mahaffey's comments in reference 87, it is also probable that the tactical systems will employ standardized or compatible hardware and/or compatible software systems. As shown in Figure 4-18, a tactical super network probably will exist at two levels. (A third level could be added, if processors are extended down to each division's subordinate brigades.) The highest level in the system is Corps cluster. Due to its dispersed nature and its varied administrative/logistical functions, the Corps cluster most probably will consist of two separate clusters of processors (Figure 4-19). The main cluster most likely will consist of one or more command and control (C2) processors (one for the Corps Main Tactical Operations Center, one for fire direction, and one for the Corps Tactical Command Post), an intelligence processor, and an air defense processor. The other cluster probably will consist of two or more processors dedicated to administrative and logistical support functions. This cluster would be located in the Corps rear area to support Corps Support Command (COSCOM)

operations. Since each Corps normally controls from 3 to 5 divisions or division-equivalents, its system must be flexible enough to allow/support easily the addition or deletion of division systems. The second level of the notional tactical super network system (Figure 4-18) represents division level systems. At least four processors (Figure 4-20) are interconnected in each division network--a C2 processor (an additional C2 processor could be allocated for the Division Tactical Command Post), an Intelligence processor, a Fire Direction processor, and an Administrative/Logistical processor. To assure reliability and availability, the C2, Intelligence, and Admin/Log systems will probably interface with Corps systems through both dedicated functional communication links and network system communication links (Figure 4-21). (The Army as discussed in references 68 and 87 currently is in the process of defining requirements for a new tactical communication system. While the exact nature of such a system is unknown, two requirements must be supported. These are data and voice transmission.)

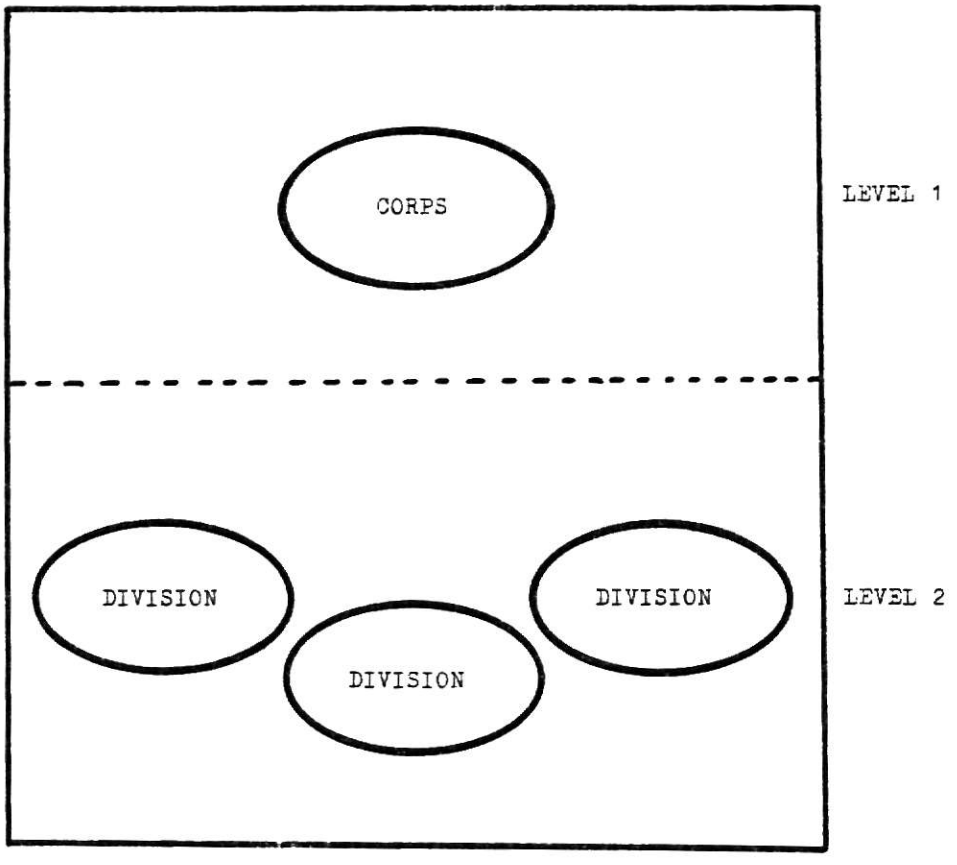


Figure 4-18: Notional Tactical
Super Network Levels

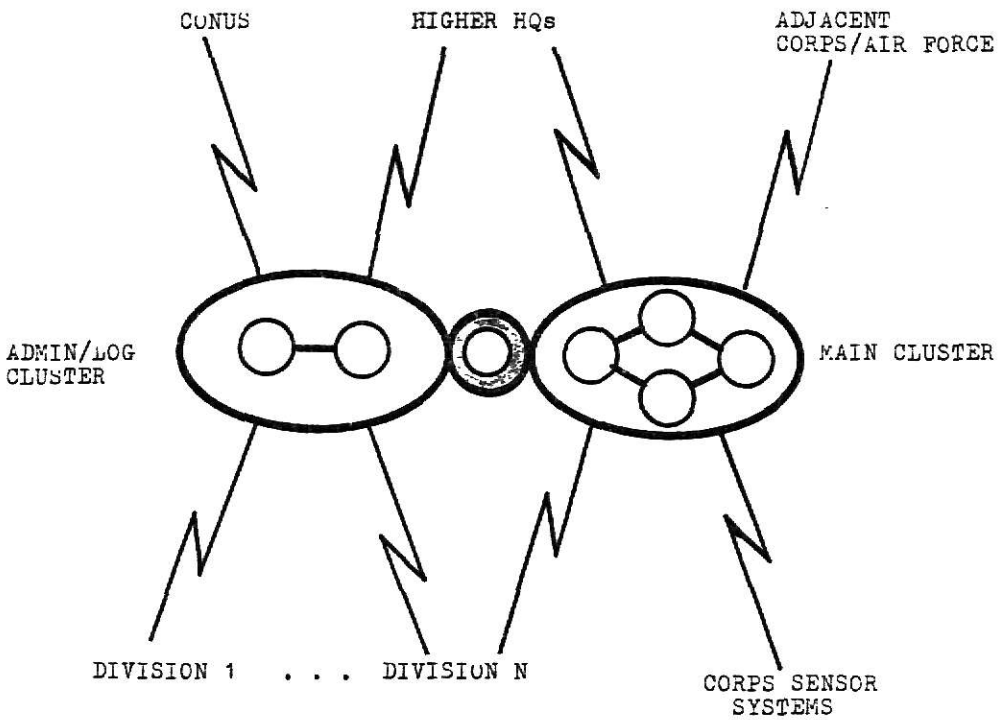


Figure 4-19: Corps Clusters

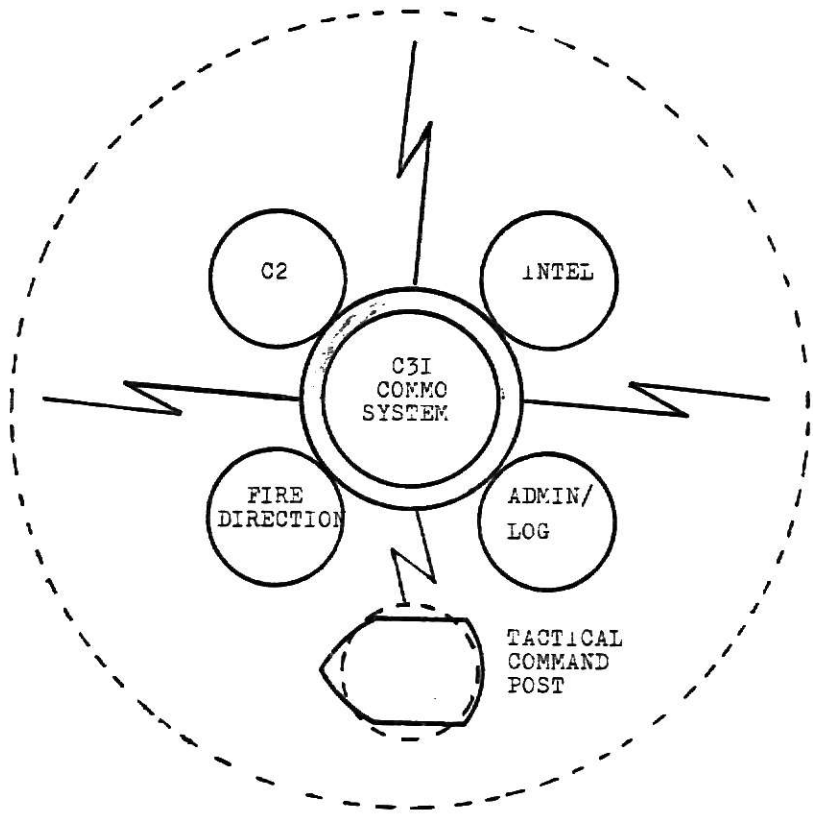


Figure 4-20: Division System

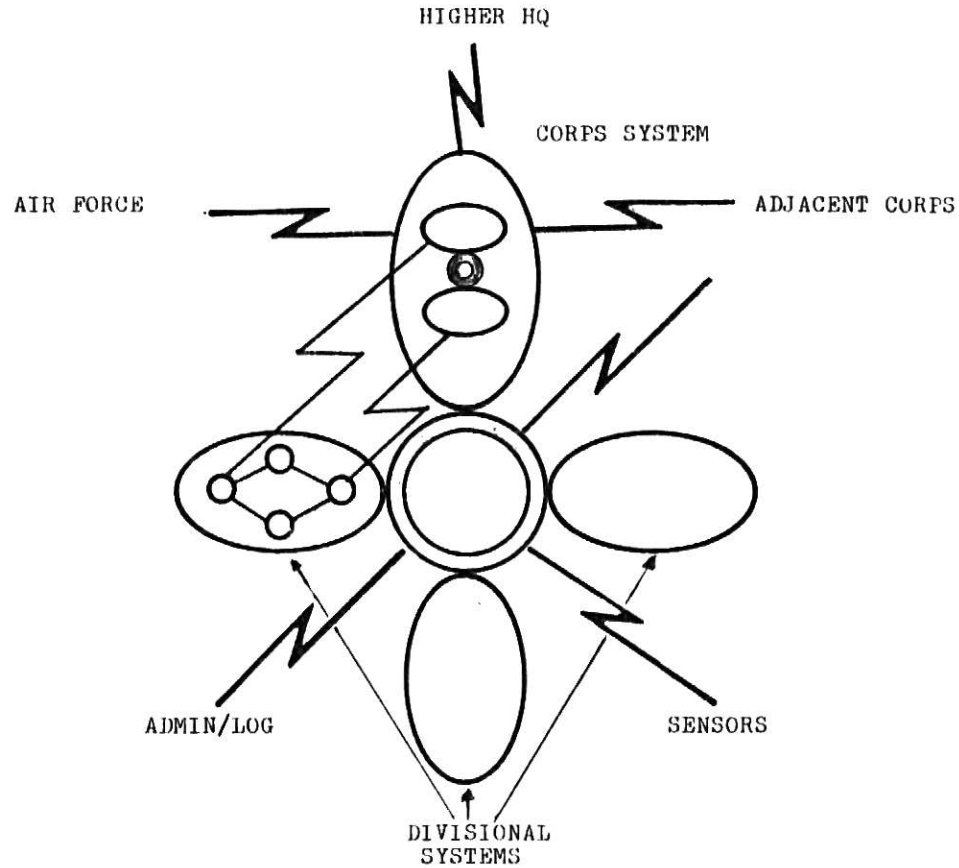


Figure 4-21: Tactical Super Network

4.5.3 Interface Requirements.

Figures 4-22 and 4-23 respectively depict the strategic and tactical super network interface problem. Assuming that both super networks have solved their intrasystem interface problems, the major obstacle to be overcome is the component command/corps interface. Since Army doctrine calls for communications to be extended from higher-to-lower command levels, each Corps system probably will have several redundant dedicated links extended to it by the component command. To handle communication interface and internetwork translation, the component command most likely will require additional processors to handle internetwork interface. Regardless of additional hardware needs, the interface translation methodology used must satisfy the following requirements:

1. It should not impose any additional burden on the tactical systems (tactical systems most probably will be limited in size and capacity due to mobility requirements).

2. It should not appreciably degrade the transmission of bulk data (national level sensor data is extremely perishable, especially in a wartime environment; therefore, the interface mechanism must not slow the flow of information).

3. It should support retrievals from either the strategic or tactical levels by both application programs and high-level language user queries.

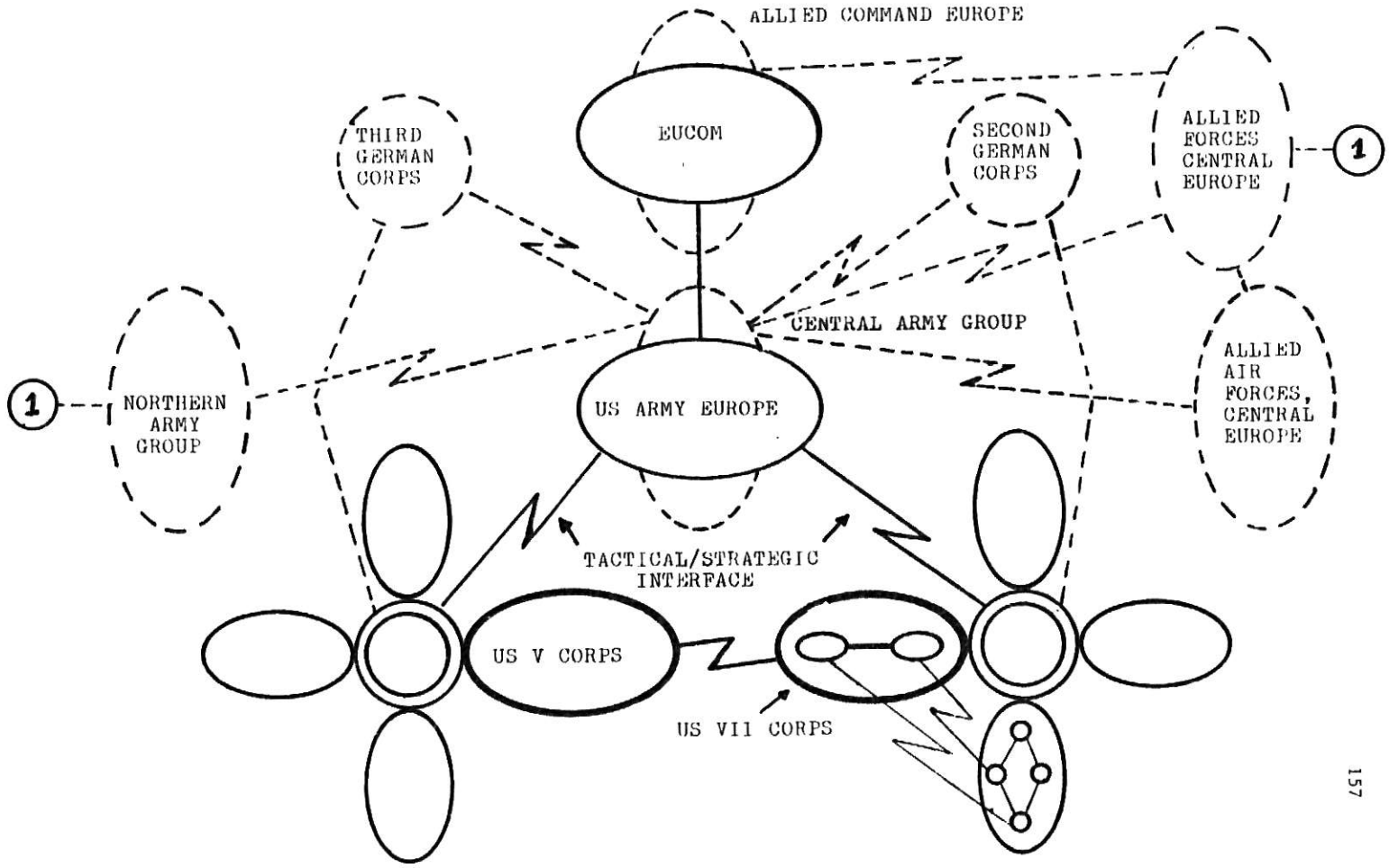


Figure 4-22: Strategic/Tactical Interface

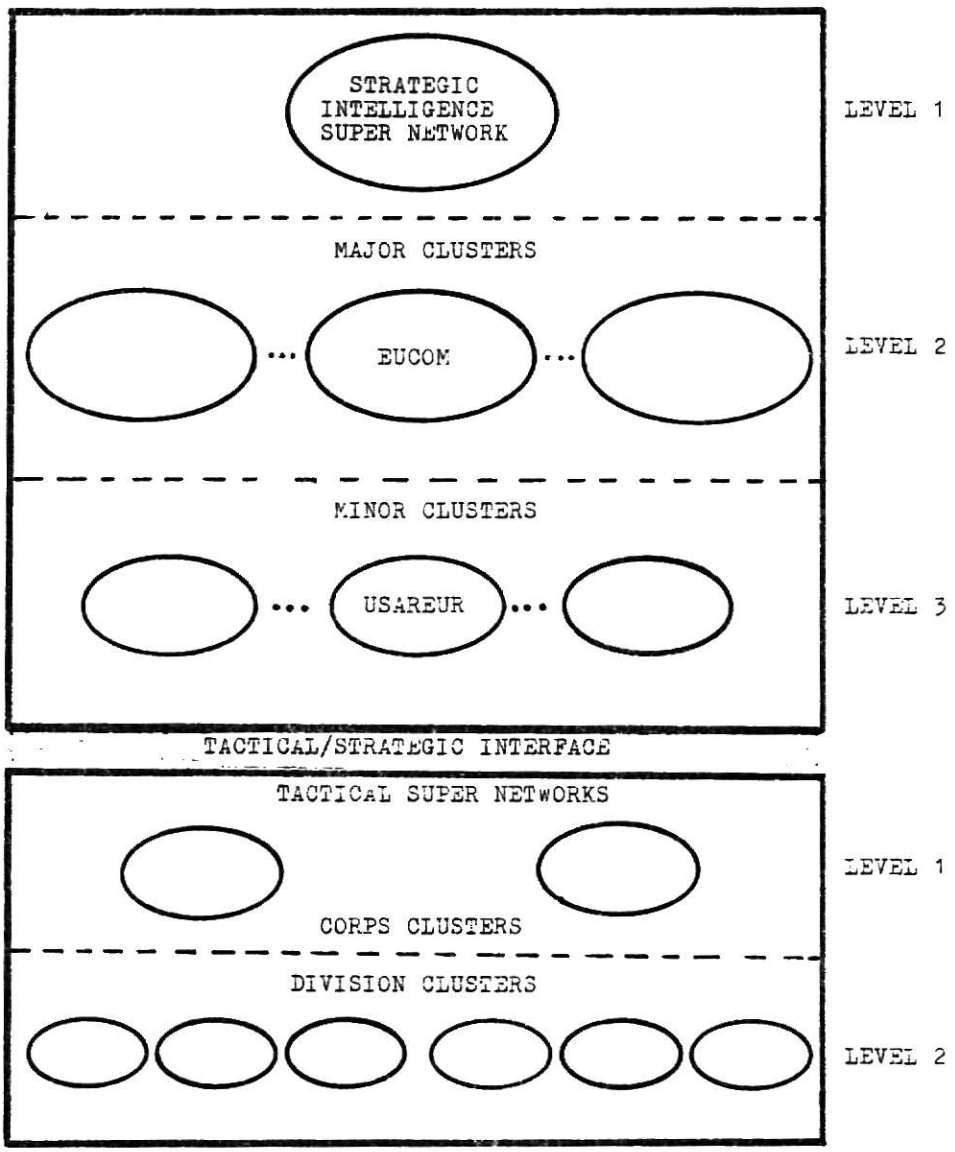


Figure 4-23: Interface

4.6 Summary.

Within the next 5-10 years, a plethora of new weapon systems, sensor systems, and support systems will be fielded in an attempt to modernize conventional US military forces. This chapter has attempted to paint a verbal picture of the realities of the military environment as it is today and probably will be tommorrow. Four major topics have been covered briefly--post Vietnam developments, intelligence shortfalls, current automated intelligence systems and development efforts, and projected automation developments. During discussion of the last topic, notional strategic and tactical super networks were described, and three interface requirements were enumerated. These requirements will be used in Chapter 6 to evaluate the translation methodologies that will be discussed in the next chapter.

CHAPTER 5

DATA TRANSLATION TECHNIQUES

5.1 Introduction.

From 1970 to 1976, a great deal of research and development (R&D) effort was expended on various methodologies for converting data from one hardware and software system to another. Unfortuately, many authors (Adiba et. al. [2], Peebles and Manning [119], Rothnie and Goodman [128], Maryanski [100], and Fry and Sibley [58]) either have viewed these efforts as entirely applicable to the generalized DDBMS translation problem or have assumed that translation problems, although non-trivial, are solvable based on the earlier work. These implicit assumptions are at best only partially valid. Because the bulk of the research effort was not directed toward solutions to the problems of data exchange among interconnected, possibly heterogeneous computer systems. Instead, most of the research was aimed at solutions to the following problems:

1. conversion of foreign files;

2. conversion of data to facilitate hardware and/or software upgrade;

3. conversion of sequential file systems into DBMS format;

4. conversion of data from one data management system to another.

A brief review of the origins (Section 5.2) of the vast majority of the R&D effort mentioned above will (1) demonstrate this contention and (2) lay the foundation for later discussion of the four methodologies suggested by Maryanski [100] as applicable to the DDBMS environment:

1. the University of Michigan Data Translation Project;

2. the Data Specification and Conversion Language proposed by G. Michael Schneider;

3. the "brute force" approach, i.e., a unique translation program for each pair of systems that will interface;

4. the Advanced Research Projects Agency (ARPA) Network Datacomputer Datalanguage.

5.2 Background.

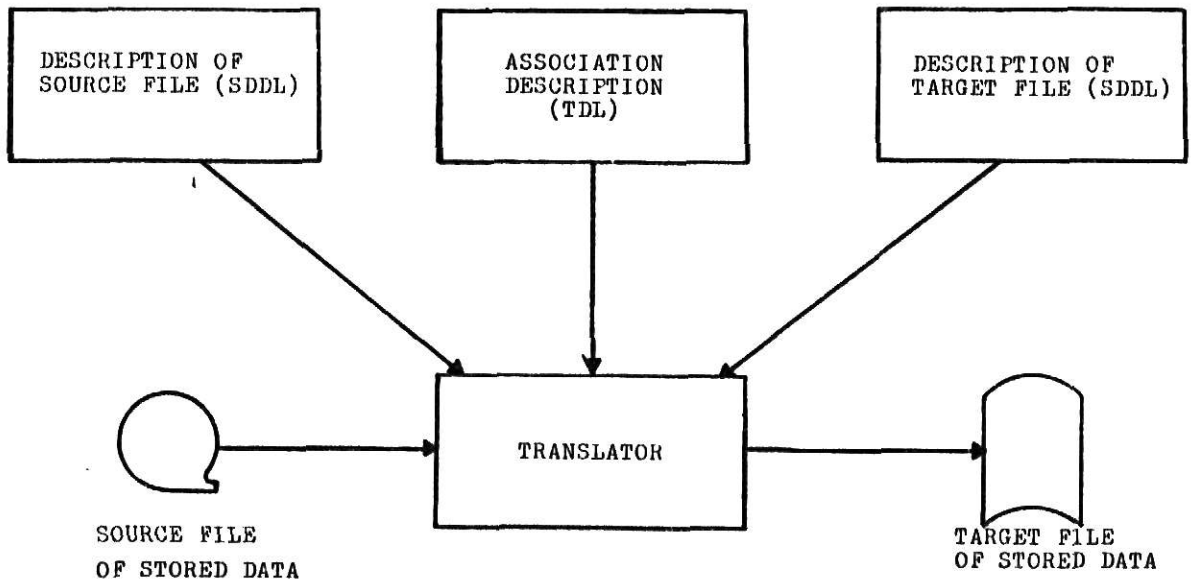
From the earliest days of computing, systems designers have had to wrestle with the problems of program and data incompatibilities whenever computer system components (hardware and/or software) were changed or upgraded. Occasionally, emulation packages and/or interface (bridge) programs were provided by or in conjunction with hardware manufacturers; however, these problems usually were solved by writing special one-time use programs to translate data from the old to the new format and by rewriting all the applications programs. In an effort to overcome some of these problems, computer manufacturers, users, and some universities and governmental agencies organized the Conference on Data System Languages (CODASYL) in 1959. CODASYL's immediate goal was to define and standardize a high-level programming language to ameliorate some of the program transferability problems. The result of this early effort was COBOL. Although CODASYL specified the language, it did not specify common data formats. As a result, the language implementers were free to specify the data formats that could be manipulated by the language. Even today, a COBOL program usually will have to be changed, especially

the Environment Division, to operate in an environment different from the one in which it was designed and written. In 1970, two different CODASYL committees (Programming Languages and Systems) established task groups to address different aspects of the data format problem. The most famous of these was the Data Base Task Group (DBTG). Working under the aegis of the Programming Languages Committee, the DBTG issued a report in April 1971 specifying among other things a Data Definition Language (DDL) to enhance the data structure capabilities of programming languages. COBOL, quite naturally, was the first language to be enhanced. While the DBTG proposal extended the logical data structures available, nothing was done in the area of standardization of data storage implementation for obvious reasons. The second task group, the Storage Structure Data Definition Language Task Group (later named the Stored Data Definition and Translation Task Group or SDDTTG), working under the CODASYL Systems Committee was established to address problems surfaced during the Systems Committee Study of generalized DBMS, i.e., the lack of data transferability or data incompatibilities. Specifically, this group was charged with designing a language for defining commonly used storage structure to facilitate data translation (conversion) [140]. At the November 1970

Association for Computing Machinery (ACM) Special Interest Committee in File Description and Translation (SICFIDET) Workshop on Data Description and Access, four papers were presented by members of the SDDTTG describing their early efforts. Fry and McGee presented companion papers, which suggested that a data and storage structure definition language could logically be divided into three separate, but interacting components--data structure definition, storage structure specification, and data-to-storage structure mapping. Young and Sibley and Taylor proposed two distinctly different approaches to the data structure mapping problem. Sibley and Taylor proposed a "non-procedural" language approach, which was later adopted; Young proposed the opposite [151]. The following year, Robert W. Taylor (University of Michigan) and Diane P. Smith (University of Pennsylvania) published dissertations which caused the SDDTTG to reorient its efforts. Taylor's dissertion, entitled "Generalized Data Base Management System Data Structures and Their Mapping to Physical Storage," dealt with the mapping of DBMS data structures to linear storage spaces; while Smith's dissertation ("An Approach to Data Description and Conversion") focused on the development of a generalized data translator system. [57] The SDDTTG's second interim report was presented at the Special Interest Group in File Description and Translation

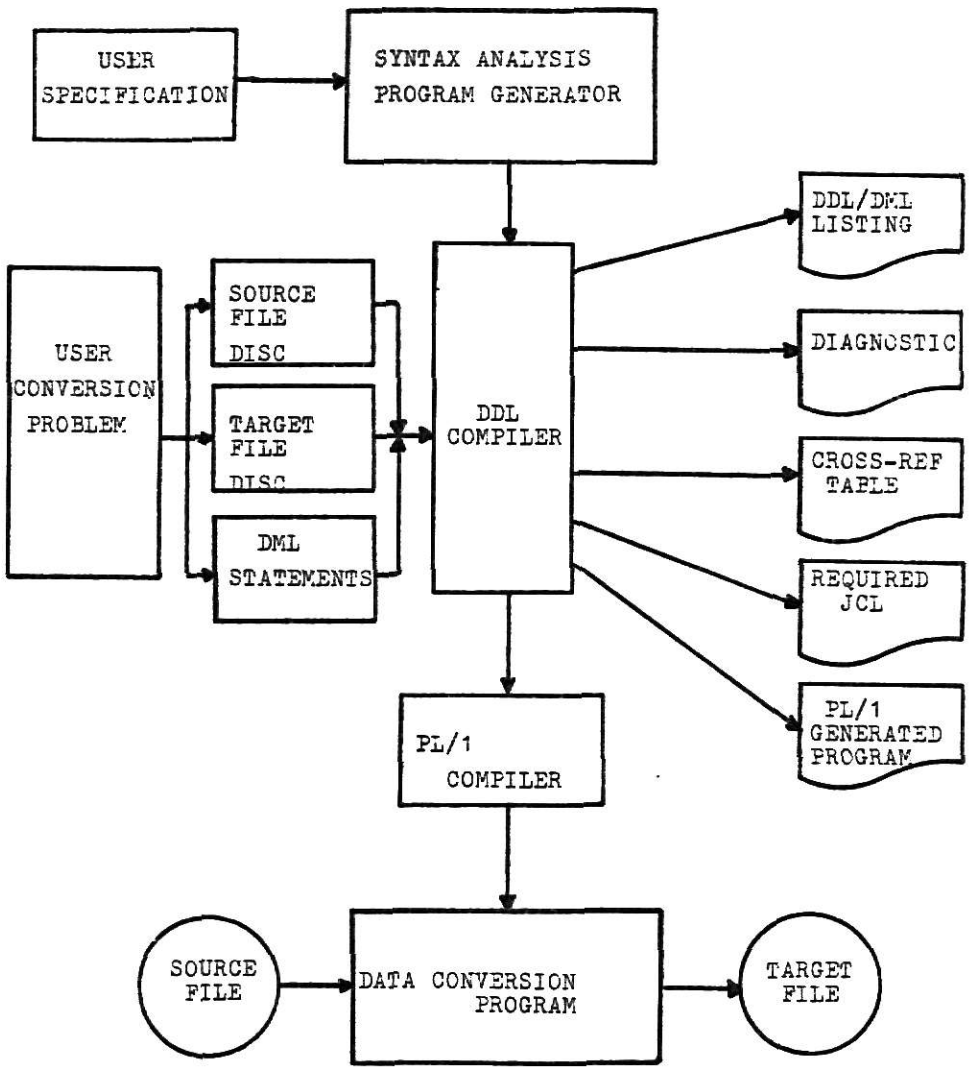
(formerly SICFIDET) Workshop in November 1972. In this report, the SDDTTG, reflecting the contributions of Smith and Taylor, acknowledged that the problems of generalized data translation and definition were much broader than earlier had been anticipated. In order "...to bridge the gap between the data formats of existing systems and the data formats required for new systems, thereby making system conversion a feasible endeavor," [57, p. 19] the group reported that two languages should be developed. The first language, a Stored-Data Definition Language (SDDL), was needed to characterize both the differing logical structures (software) and the disparate internal machine data representations of the source and target files. The second language, a Translation Definition Language (TDL), was necessary to define the mapping from source to target file data instances. [57] A conceptual view of the translation methodology proposed in the report is shown in Figure 5-1. By 1973, the SDDTTG's work was overshadowed by the limited implementation work that had begun at the University of Pennsylvania and the University of Michigan. At the former, J. A. Ramirez implemented a compiler translation system based on Smith's earlier specifications. Ramirez's system accepted as input three descriptions--the logical and physical storage structure of the data to be translated in

DDL, the data structure of the target data base in DDL, and the elements and values to be translated in Data Manipulation Language (DML). As output, it produced a special purpose translator, which would translate all sets of data that matched the source and target descriptions (See Figure 5-2). [124] Although the efforts of Smith, Ramirez and others who suggested slightly different approaches (references 7,70,84,115,124,137-139,151, and 168 represent a cross-section) were noteworthy, the most famous, and perhaps most extensive, work done in the area of data translation was carried out by the University of Michigan's Data Translation Project Group. At one time or another, this group included Birss, Frank, Fry, Merten, Sibley, and Taylor.



(Source: 57)

Figure 5-1: SDDTTG Translation Methodology



(Source: 124)

Figure 5-2: Compiler
Translation System

Before the SDDTTG's second interim report was issued, a Stored Data Description and Translation (SDDT) Working Group had been formed at the University of Michigan. The original purpose of the SDDT Group was to provide a facility that would help designers solve the problems associated with converting from one file management system to another. In 1973, Sibley and Taylor [140] published an article that clearly raised the game's ante and also probably muddied the waters for later researchers. This paper called for development of a Data Definition and Mapping Language, which Sibley and Taylor saw as "...a key element in the translation of data between computer systems, as well as in advanced data management systems and distributed data bases." [140, p. 750] The authors did not suggest that such a language would solve the translation problems inherent in a generalized DDBMS environment; however, the article must be read several times before this becomes apparent. Sibley and Taylor were, in fact, only advancing the argument (see below) that an explicit definition of data was a necessary prerequisite to/key element of the solution to the problem of interfacing distributed data base systems.

> ...with the advent of networks of computers which may have one processor retrieving (or causing retrieval of) data from a distributed data base, the processor should request data at a logical level, and retrieval will then depend on the accessing of self-defining data and storage

> mapping structures... this need implies a method of defining not only the logical data structure but a more complete data definition involving also the storage media description and mapping of data into a storage structure. [140, p. 752]

In effect, Sibley and Taylor, as the quote above shows, were simply extending the scope of the SDDT Group's earlier endeavors and providing additional justification for continued developmental work.

5.3 University of Michigan Data Translation Project.

The University of Michigan SDDT Group not only developed a generalized translation methodology, but they also designed four prototype translators and implemented three of them. Before their prototying endeavors are described, the SDDT Group's generalized translation methodology will be discussed briefly.

The methodology originally proposed by the SDDTTG and later implemented by the SDDT Group (See Figure 5-3) is a two step interpretive process. [21] (An earlier report [110] included an additional step--analysis for completeness and correctness.) During the first step, the user supplies certain specifications using two languages developed by the SDDT GRoup--the Stored-Data Definition Langauge (SDDL) and

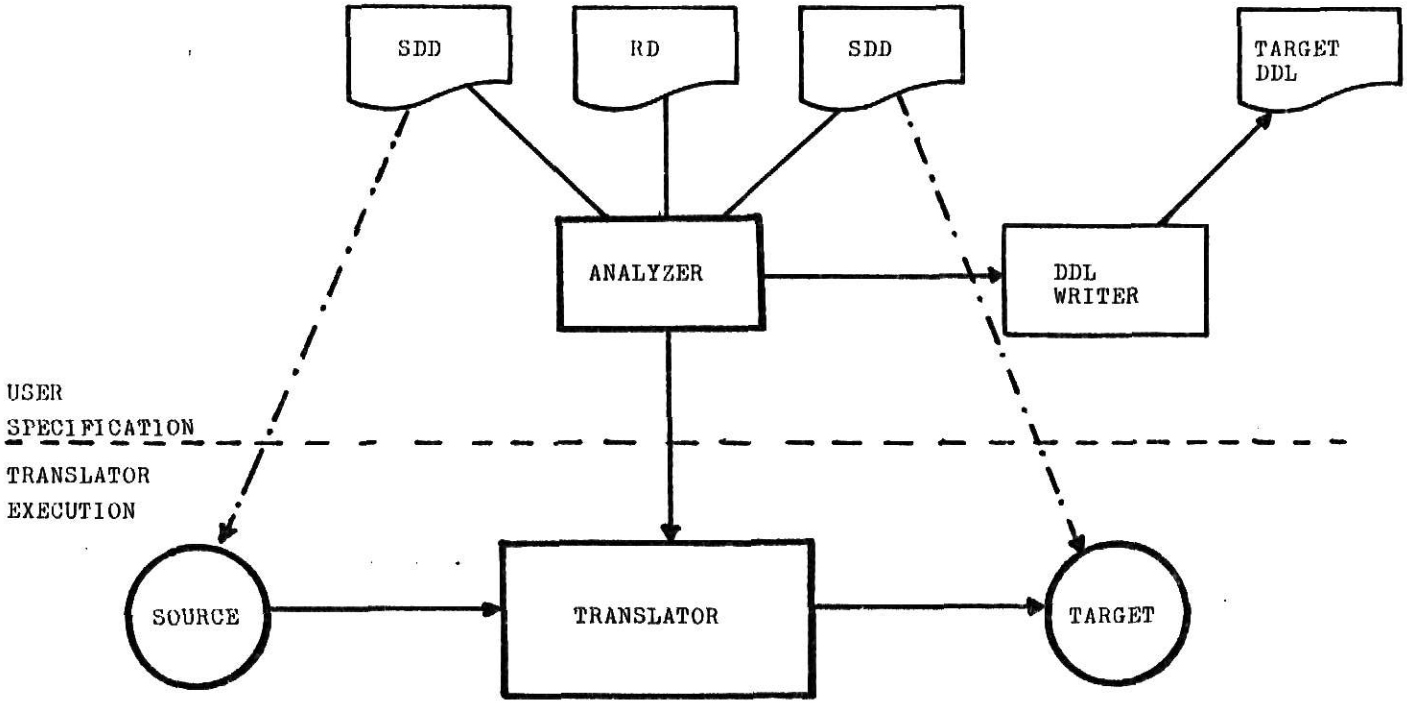
the Translator Definition Language (TDL). The SDDL, which was based largely on the language that Taylor proposed in his dissertation, is a high-level non-procedural language that closely resembles the DDL proposed by the DBTG. (The biggest difference between the DBTG's DDL and Taylor's SDDL is that the SDDL also describes the physical representation of logical data structures.) The user provides descriptions of the logical, physical, and relational aspects of both the source and the target data files/bases using the SDDL. The TDL, the second high-level language developed by the SDDT Group, is used to describe the logical transformation (mapping) of data instances from the source to the target system. After the SDDL and TDL descriptions are processed by an Analyzer to produce encoded versions, i.e., Encoded Stored-Data Descriptions (ESDD) and Encoded Restructuring Descriptions (ERD) respectively, the second step of the process begins. Driven by the ESDDs and ERDs, three modules (Reader, Restructurer, and Writer) are employed to physically transform the source data into the target data (See Figure 5-4). As described by Birss and Fry, the functions of the three modules are:

> The Reader Module, driven in part by the source ESDD performs many functions, sequentially accessing physical records, physically deblocking these records, logically identifying their components, and automatically creating an internal data base processable by the restructurer. In

dealing with complex data base structures, the Reader must keep track of access paths between these unblocked records so that the restructurer is provided with an accurate representation of the source data base.

The Restructurer accesses the internal representation of the source data base and converts it to a representation of the target data base. The conversion from source to target is a transformation of the logical structure and is directed by the ERD which contains the restructuring specifications.

The Writer, driven in part by the target ESDD, creates the target data base by constructing the target data from the Restructurer's internal data base. [21, pp. 890-891]

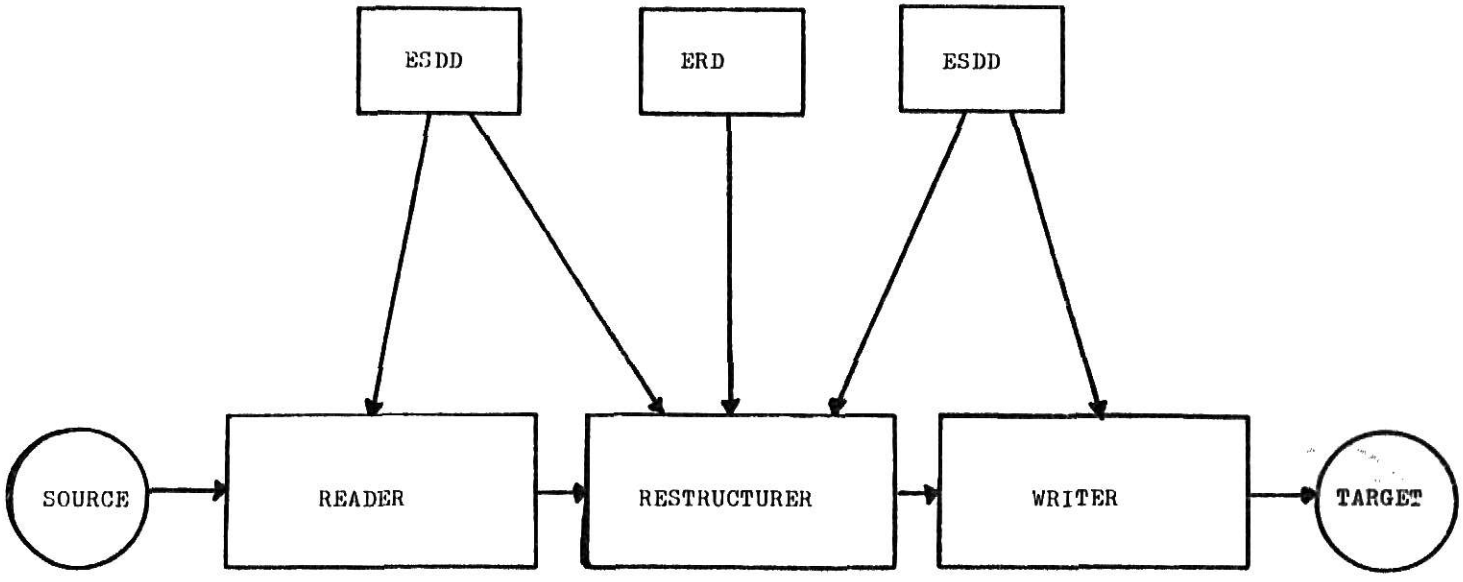


SDD - STORED-DATA DESCRIPTION

RD - RESTRUCTURING DESCRIPTION

(Source: 21)

Figure 5-3: University of Michigan Translation Approach



(Source: 21)

Figure 5-4: Translator Modules

From 1973 to 1976, the SDDT Group designed four translators--prototype, Version I, Version II, and Version IIA. All except Version II were implemented. Detailed descriptions of each of these increasingly more complex translators is beyond the scope of this paper (See Figure 5-5 for a comparison of capabilities). However, since Versions II and IIA represented attempts to translate logical DBMS structures, they will be examined.

Version II was originally designed to handle sequential, indexed sequential (Honeywell Indexed Sequential Processing or ISP), and network (Honeywell Integrated Data Store or IDS) structures. As the SDDT Group later learned, the complexity of a logical network structure resident on disks required that a number of additional components be added to the overall translator design (See Figure 5-6). For example, accessing disk rather than tape required that low-level routines for each type of data be coded to handle the more complicated input/output operations. Additionally, the group found that the Restructurer had to access the entire data base; therefore, an internal data base was designed--the Restructurer Internal Form (RIF). Since the generalized Restructurer was incapable of managing a data base, a data management system was added to (1) manage the RIF as well as the ESDD and ERD tables and (2) allow the

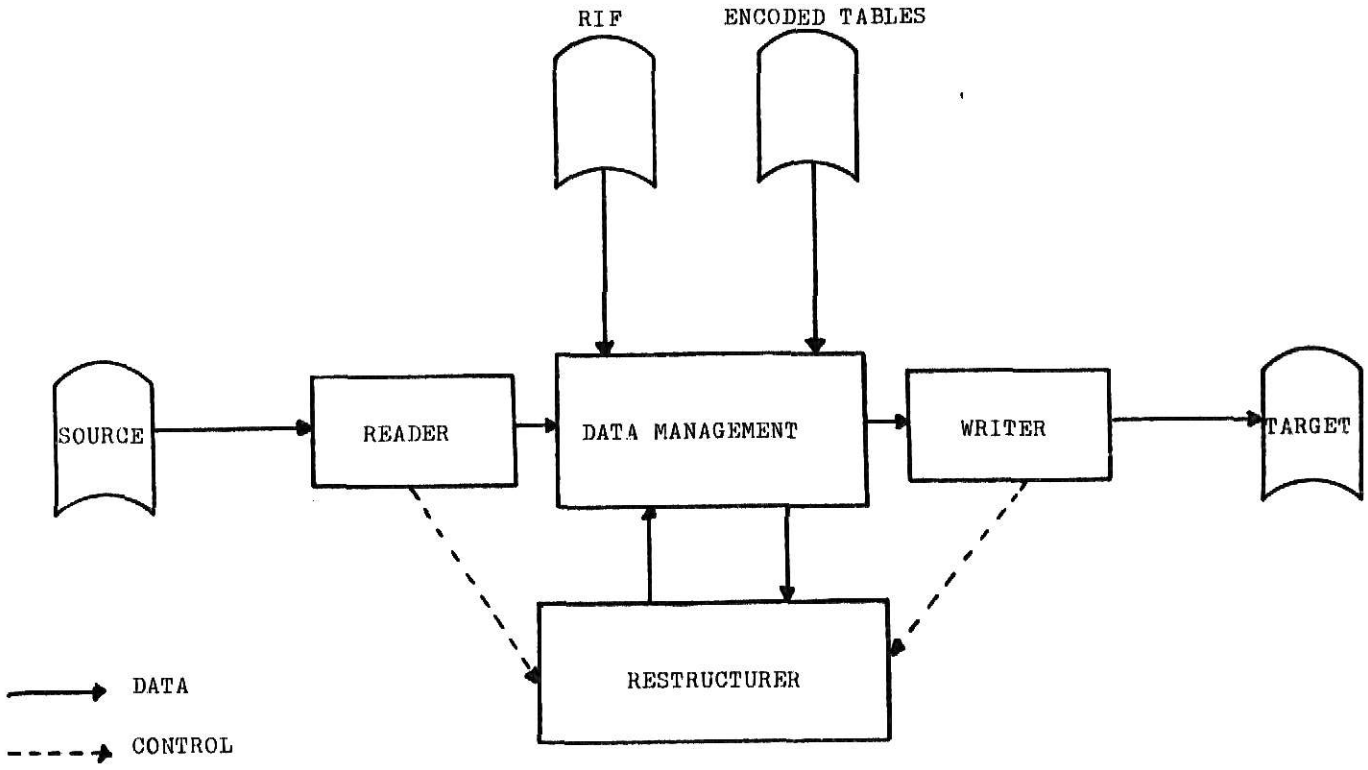
Restructurer to access the RIF. After several limitedly successful IDS data base reorganizations, the Group decided to abandon this effort and to concentrate instead on optimizing the Restructurer to handle restructuring of IDS data base organizations solely. Thus, Version IIA was born. [21]

Overall generality was sacrificed in Version IIA (See Figure 5-7). The Reader and Writer were tailored to process only IDS data base formats. An expanded version of IDS's DDL was used in lieu of the SDDL for the source and the target data base descriptions. This required that an IDS DDL specific analyzer (interpreter) be built to process the extended DDL and produce the ESDD. The Restructurer's capabilities were also expanded. In fact, as Birss and Fry reported, "the Restructurer implemented...could not only change the implementation structure of existing I-D-S [SIC] relationships, but also had the capability to create structures more powerful than I-D-S I [sic] could process." [21, p. 895] (NOTE: IDS I is an enhanced version of IDS; later Honeywell also introduced IDS II.)

The SDDT Group established the technical feasibility of transferring logical data base structures created by one DBMS into a form required by another using a generalized translation process. However, from the physical transformation standpoint, the Group was much less successful. As Birss and Fry stated in their last paper on the subject, "at this point in the research, the development of a completely general transformation modules [SIC] does not appear to be economically justifiable. [21, p. 896]

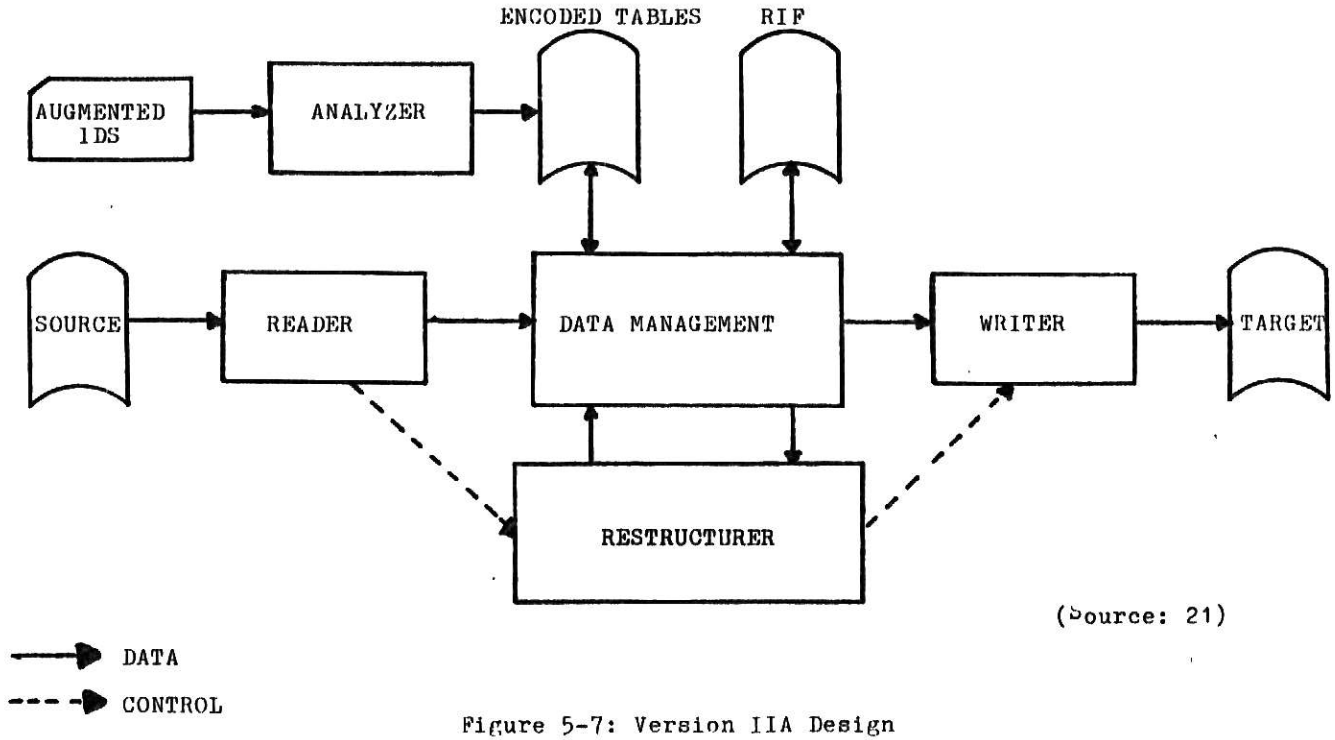
	PHYSICAL CAPABILITIES	LOGICAL RESTRUCTURING	QUANTITY OF DATA	OTHER FEATURES
PROTO	SEQ	NONE	RECORD	N/A
Ver I	SEQ	COMPRESSION	ENTRY	GENERALIZED SEQ RDF
Ver II	SEQ,ISP,IDS	HIERARCHICAL	D. BASE	
Ver IIA	IDS	NETWORK	D. BASE	DATA ADDITION INTEGRATION OF DATA BASES

Figure 5-5: Comparison of Capabilities



(Source: 21)

Figure 5-6: Version II Design



(Source: 21)

Figure 5-7: Version IIA Design

5.4 Data Specification and Conversion Language (DSCL).

Like the earlier efforts of the SDDTTG and the University of Michigan project, G. Michael Schneider's research (Ph.D. dissertation) was originally directed toward the development of a file translation capability. As the project evolved, however, Schneider expanded the language's proposed capabilities and changed its name from File Translation Language to DSCL, because it was "...a more semantically accurate description of the project goals." [133, p. 141] Unlike the other efforts, DSCL was oriented toward a solution to the problem of interfacing distinct, possibly incompatible network nodes. At the ACM SIGMOD (Special Interest Group in Management of Data) Workshop in 1975, Schneider presented a paper which not only described his initial implementation of a prototype DSCL processor (UNIVAC 1108), but also presented his goal of providing a centralized DSCL-like translation service (See Figure 5-8) that would effect "...the real-time translation and transmission of data streams between nodes of a computer network." [133, p. 139] The DSCL prototype language (a combination of non-procedural and procedural elements) only supported some physical data stream restructuring capabilities; however, Schneider envisioned several enhancements. First, the DSCL was to be expanded to support

logical data stream structures as well as physical data stream structures. Second, procedural portions of the DSCL were to be eliminated. In this regard, Schneider stated that the DSCL project would rely heavily on the earlier DDL and TDL efforts of Smith (Unversity of Pennsylvania) and Fry (University of Michigan). Third, message-exchange protocols would be developed to support "...asynchronous communications between three distinct processes - the source process generating the data stream, the DSCL process transforming the data stream, and the receiver process accepting the newly translated stream." [133, p. 147] Besides the low-level details of routing and flow control, the message exchange protocol, according to Schneider, must provide conventions for handling the following:

> 1) Initiating a translation process and identifying the DSCL Control Program
> 2) Identifying via a network-wide address the source and the receiver processes
> 3) The handshaking procedure between DSCL and the source and receiver processes
> 4) The procedure for requesting input blocks from the source process and passing on newly translated output blocks to the receiver process
> 5) Error recovery procedures in the event of abnormal termination [133, p. 147]

Whether these enhancements were ever made is unknown, because no additional material has ever been issued by Schneider or any of his colleagues. It is probable that his

efforts were overtaken by events, e.g., development of the ARPA Network Datacomputer.

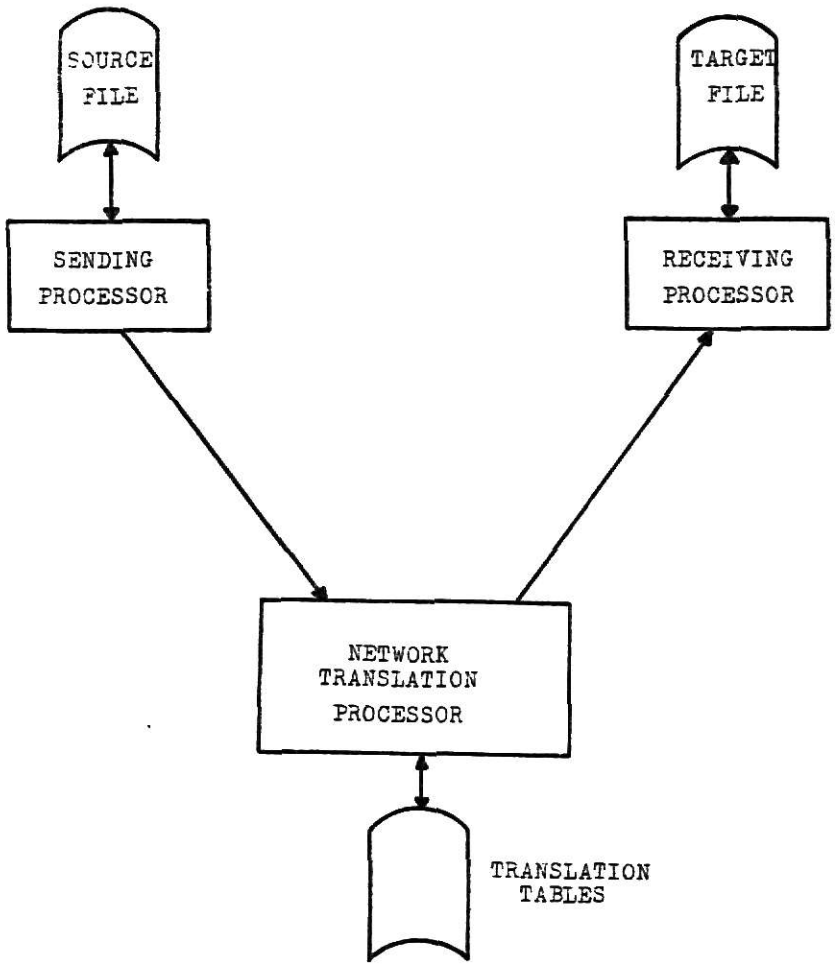


Figure 5-8: DSCL Translation Service

5.5 The "Brute Force" Approach.

Although Maryanski, et. al. [105] discuss the translation problems of incompatible back-end DBMSs and Maryanski [100] suggests that the "brute force" approach, i.e., constructing a unique translator for each pair of data base systems, is a viable alternative, he does not provide answers to a number of questions. For example, where does each network node in a generalized DDBMS store 2(K-1) translation programs; how do these translators fit into the overall DDBMS architecture; or which component(s) is (are) responsible for the translation process? Unger, et. al. [156] suggest one solution, i.e., to split the translation functions between a Network Interface (NINT) mechanism and the network communication system. As described in their paper, the NINT is a facility "...which allows any of the components comprising a generalized distributed DBMS to access the network connecting the components." [156, p. 4] The NINT, which consists of two modules (Command Interpreter and Command Handler), is essentially equivalent to the Global Data Manager (GDM) suggested by Rothnie and Goodman for SDD-1. [129] The basic functions of the NINT are threefold:

1. processing user requests to determine the

operations that need to be performed;

2. determining data storage location by accessing a system directory;

3. deciding where each operation should be performed, i.e., which component, and how the user request is divided into subcommands. (The selection algorithm must consider two basic factors--access efficiency and communication system costs.)

In order to accomplish these functions, especially in a heterogeneous or generalized DDBMS environment, the NINT or GDM must have access to translation routines. In the system proposed by Unger et. al. [156], the Command Handler (CH) is responsible for accessing these routines in the course of constructing the subcommands. Since the NINT only supports command/subcommand translation, the communication system must be responsible for all other aspects of intercomponent translation. How the communication system handles its portion of the translation process is not addressed; therefore, it still must be considered an open question.

5.6 Datacomputer Datalanguage.

Although described as a network data utility [88], the ARPA Network Datacomputer is actually a large scale back-end computer (See Figure 5-9). As such, its primary purpose is to provide data sharing services to a large number of users with heterogeneous machines. To accomplish this, translation/conversion problems had to be solved. The Datacomputer designers developed a two-level approach to the problem of data translation. First, since the data bases had to be shareable by a wide variety of users with different hardware configurations; the Datacomputer was designed to perform both physical data representation conversion and logical data restructuring. Second, a Datalanguage was designed to allow self-contained requests to be transmitted in toto to the Datacomputer, where the specified tasks are executed and the requisite data returned to the user in a user pre-specified format. (The Datalanguage does not support a high-level query language.) At first glance, the Datalanguage appears to be little more than a standardized set of data manipulation language (DML) primitives, which users incorporate into their application programs. However, the Datalanguage also "...includes

facilities for data description, for database creation and maintenance,...and for access to a wide variety of auxiliary facilities and services." [88, p. 392] Since the Datalanguage is a high-level non-procedural language, detailed data descriptions have to be available not only for the Datacomputer, but also for user sites. These are stored in machine readable form in a special Datacomputer directory. Whenever a Datacomputer task is executed, two descriptions are retrieved from this directory--a file description and a port description. (The file description describes the data as it is logically stored on the Datacomputer; the port description describes the standardized data format required by the communication system.) The Datacomputer uses these descriptions to map the data between the two representations. This process may entail "...conversion from one elementary data type to another... [as well as]...pruning and reordering of branches in a hierarchical data structure." [88, p. 392] Although all the facilities described were not available initially, the Datacomputer and the Datalanguage have been expanded incrementally. Today, the Datacomputer, a functioning system resource available to nodes in the ARPA Network, is the only system with a working multi-nodal translation capability.

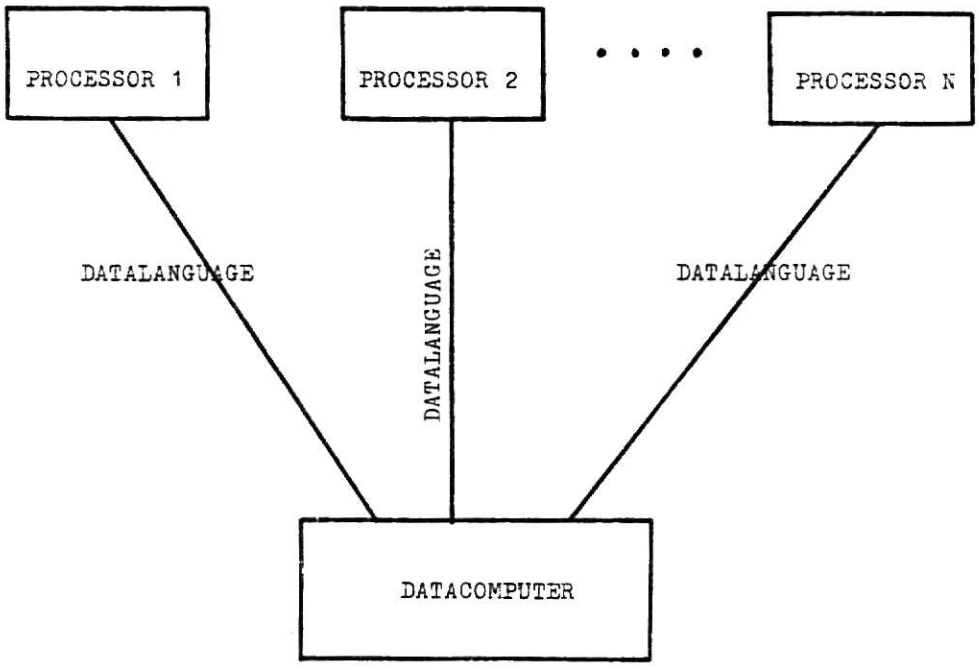


Figure 5-9: Datacomputer

5.7 Summary.

Four data translation approaches have been discussed in this chapter: (1) the University of Michigan Data Translation Project; (2) G. Michael Schneider's Data Specification and Conversion Language; (3) the "brute force" or NINT-to-NINT approach; and (4) the ARPA Network Datacomputer Datalanguage. As shown, these approaches collectively represent only a small percentage of the total amount of R&D effort that has been expended on data translation since 1970 (See references 7,70,84,115,124,137-139,151, and 168 among others).

In the final chapter, the four approaches discussed in this chapter will be analyzed and evaluated against the criteria listed in Chapter 4.

CHAPTER 6

EVALUATION AND CONCLUSIONS

6.1 Introduction.

As shown in Chapter 5, a great deal of research and development (R&D) effort has been expended in the data conversion/translation area since 1970. Unfortunately, much of this R&D effort was not aimed directly toward solutions to the data conversion/translation problems of the generalized DDBMS environment, but rather toward more basic problems, e.g., conversion of foreign files, conversion of data to facilitate hardware and/or software upgrades, conversion of sequential files into DBMS formats, and conversion of data from one data management system to another. In this chapter, each of the data conversion/translation methodologies discussed in the previous chapter will evaluated against the criteria established in Section 4.5 of Chapter 4. To set the stage for this discussion, a brief sketch of the varied environments in which both the strategic intelligence and the tactical super networks must operate is presented first.

6.2 Environmental Considerations.

Hopefully, another global war will never occur; however, the Military Establishment to include both its human and materiel resources must be prepared for such an eventuality. During peacetime, interface between the super networks is not nearly so critical as either during wartime or the transition from peace to war. In a peacetime environment, the exchange of intelligence/information (data) between the super network systems will be more heavily weighted toward the strategic-to-tactical rather than the converse. The primary reason for this is the fact that strategic intelligence considerations are the driving force of the national intelligence effort during peacetime. Accordingly, strategic sensors, which have far greater ranges than their tactical counterparts, normally provide the bulk of the information collected. Additionally, tactical sensor usage is more closely monitored and centrally controlled than during wartime for two reasons. First, the early indications of preparations for hostilities will most likely occur beyond the range of tactical sensors. Second, tactical systems are employed sparingly during

peacetime in order to prolong their life expectancy and assure their availability for their basic wartime missions. In sum, strategic systems during peacetime are the primary collectors, while tactical sensors are cast in a supporting/secondary role. During the transition from peace to war and during wartime, tactical sensors will predominate and data should flow almost equally both ways with perhaps the tactical-to-strategic side dominating slightly.

6.3 *Analysis of Techniques*.

Before analyzing the data conversion/translation techniques discussed in Chapter 5, the three criteria to be used in the evaluation will be reviewed briefly as an aid to the reader. For a data conversion/translation methodology to be considered viable for military super network interface, it must:

1. not impose an additional burden on tactical systems;
2. not appreciably degrade bulk data exchange flow;
3. support both application program and high-level query language retrievals.

Using these criteria, it should be apparent that the University of Michigan's generalized translation methodology is not a satisfactory alternative for the following reasons. First, this approach, if used, would clearly impose an additional, if not intolerable, burden on every processor in both super networks. Second, the University of Michigan abandoned the generalized translation methodology, because of its complexity and the slowness of the translation processor. Third, this technique was designed for complete file/data base to data base translation and not for selective retrievals.

Schneider's DSCL approach is not feasible, because it was never developed beyond the specification stage as far as logical data structure conversion/translation is concerned. Additionally, while Schneider's centralized translation processor approach would not impose any additional burden on tactical systems, it has a built-in potential for catastrophic failure, i.e., if the translation processor is down for any reason, there would be no interface between the super networks. Increasing the number of translation processors would enhance overall reliability; yet both the cost and the vulnerability of several fixed site translation

processors might be too high in a wartime environment.

While it is more simplistic than either of the two previous techniques, the "brute force" approach would create an intolerable burden for all processors in both networks. For example, if there were only 30 different hardware/software confiugurations in both systems, each node would have to store at least 58 different translation programs to satisfy all conceivable requirements. The brute force also does not provide for a high-level query facility.

The Datacomputer approach, i.e., one large back-end system or network data utility, like Schneider's DSCL is simply not feasible in a military environment for many of the same reasons. Yet the Datalanguage developed for use on the Datacomputer has potential. For example, its standardized set of Data Manipulation Language (DML) primitives and its standardized transaction format represents perhaps the most feasible aspects of the four techniques considered.

6.4 Conclusions/Recommendations.

While much effort has been expended, only selected portions of the previous research are even remotely applicable to the generalized DDBMS environment. Nonetheless, from the larger perspective, each effort has contributed to general understanding and collectively, they suggest directions for future research. The Network Interface (NINT) mechanism suggested by Unger, et. al. or the Global Data Module (GDM) or manager suggested by Rothnie and Goodman would appear to provide an excellent framework from which to approach the conversion/translation problem. For example, to create the type of environment needed by military users and implied by the benefits claimed for DDBMSs, a high-level query language facility could be implemented across both super networks using the NINT as a base (See Figure 6-1 for a conceptual representation of such a facility). In addition, the NINT or GDM in conjunction with a network processor could handle the conversion/translation requirements. In order to simplify this aspect, it would appear that the Datalanguage approach of a system-wide DML or at least standardized DML type

primitives would simplify the NINT subcommand translation problem as well as portions of the intercomponent communication problem. Additionally, the Datalanguage's standardized transaction format suggests that a common or standardized data interchange format or protocol needs to be developed. (This view is supported by Birss and Fry's conclusions in reference 21.) If a system-wide format were adopted, it would eliminate much of the translation overhead for each individual component and tend to simplify several aspects of the conversion/translation problem as it relates to DDBMSs in general and the super networks in particular. Although not required for on-line conversion, a generalized DDBMS in a military environment should have mechanisms that would permit either an entire data base or portions thereof to be moved and installed easily on another system. These tools could be used by data base administrators at Corps level, for example, to construct a data base for a newly arrived replacement division or to reconfigure/reconstruct a data base on a system that failed or was down for one reason or another. The University of Michigan s SDDL or an expanded Data Description Language seem to be logical starting points for the development of such facilities.

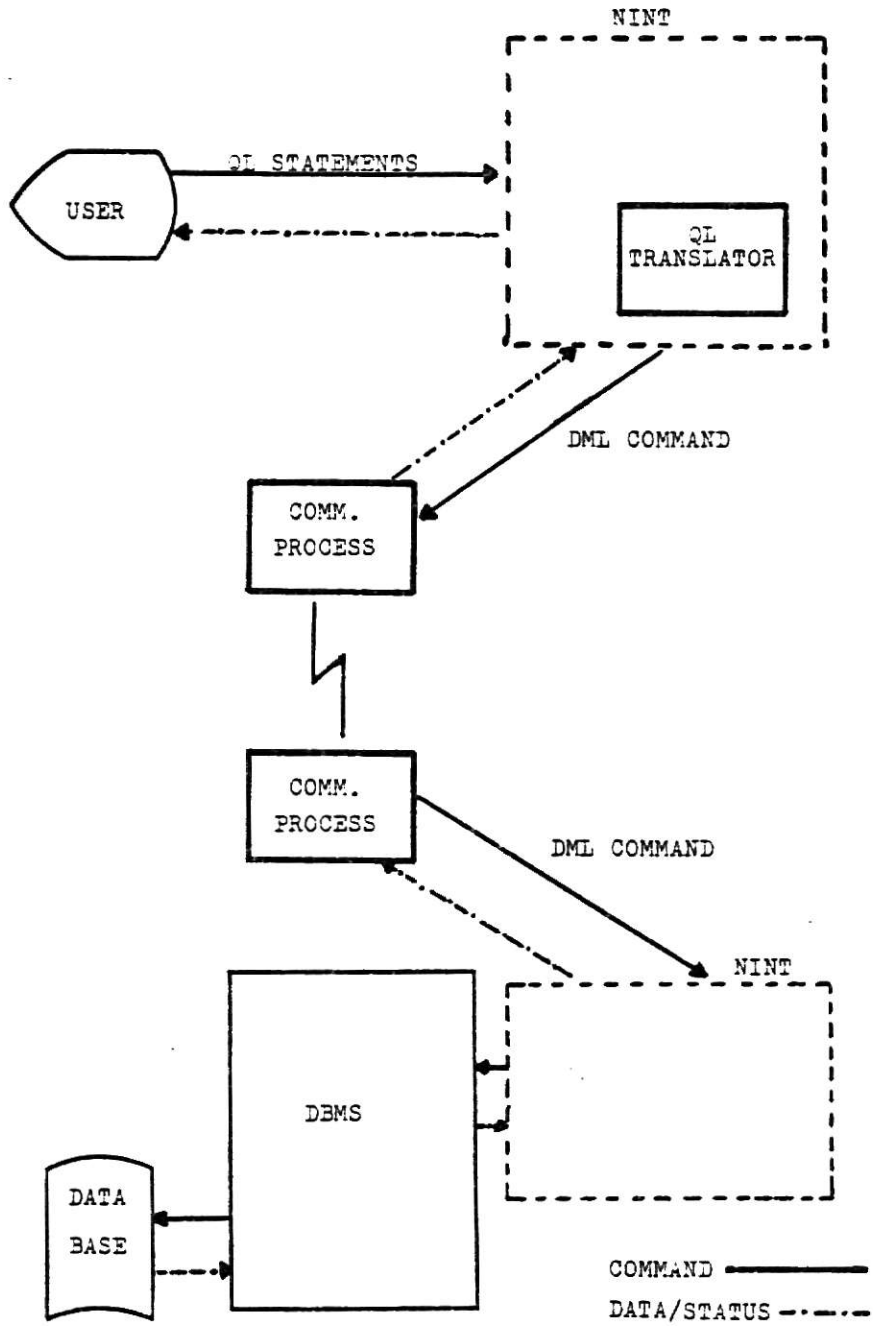


Figure 6-1: Query Language Process

The translation problems posed by the interface of two super networks are far from being solved. Additional effort must be expended in the area of conversion/translation before such systems can be interfaced. While directions for additional research have been suggested, these are by no means complete. Before any further research is undertaken, however, a complete top-down system design should be accomplished. The model and protocols suggested by White [164] could serve as a starting point for the development of a conceptual view of the entire intercomponent processing problem (Figure 6-2). It is probable that a top-down study would show that greater efforts are needed in the area of standardization. Traditionally, the Military Establishment has been reluctant to change. However, if the C3I goals outlined by the SECDEF are to be achieved, interdepartmental politics must be set aside in favor of greater efforts toward standardization. For unless the current approach of developing special purpose systems is altered, an overall C3I systems development plan is established, and standardization efforts increased, the Defense Department may well have to face the problem of interfacing a number of incompatible systems at both the strategic and tactical levels with all the problems that this sort of endeavor entails (See reference 38).

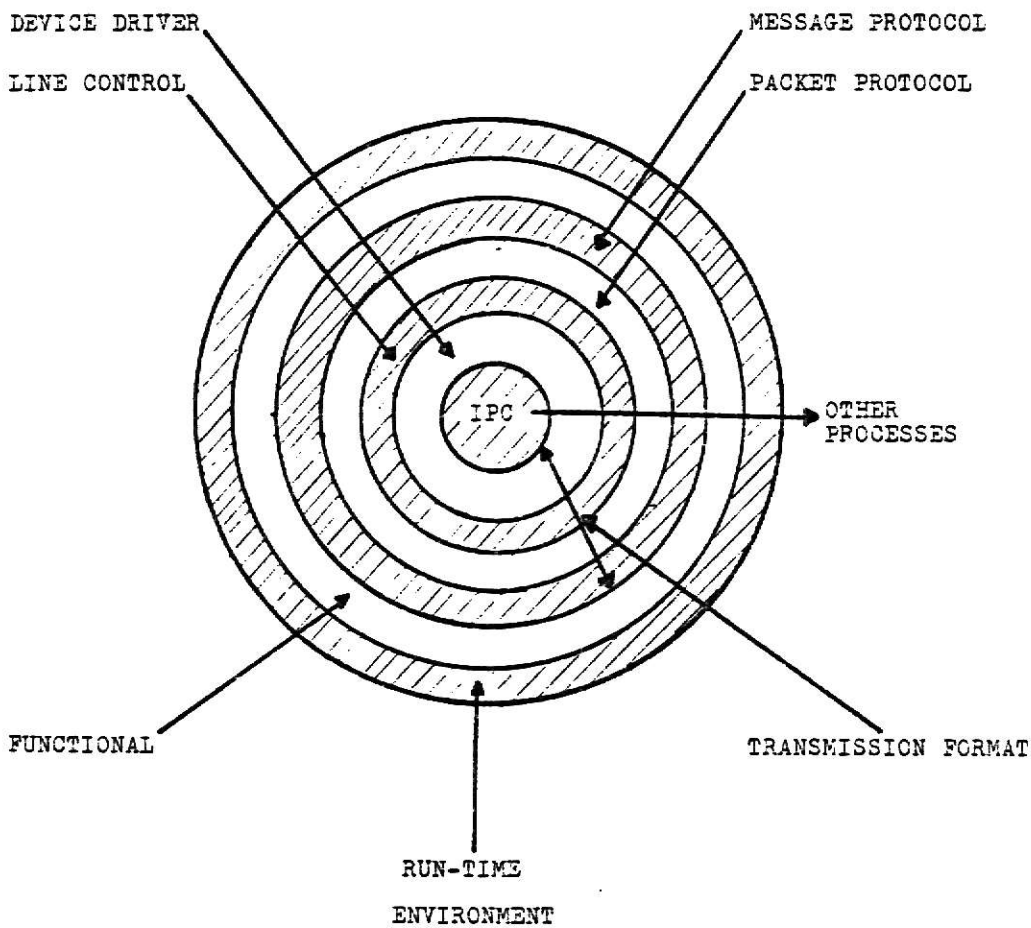


Figure 6-2: Interprocess Communications

References

1. ACTS (ASSIST Conus Testbed System) Functional Description, Jointly prepared by Office of the Assistant Chief of Staff, Department of the Army and Forces Command Intelligence Center, April 1973.

2. Adiba, M., et. al., "Issues in Distributed Data Base Management Systems: A Technical Overview," Proc. Fourth International Conference on Very Large Data Bases, October 1978, pp. 89-110.

3. Anders, F., et. al., "Intelligence Security Subsystem--Final Technical Report," Harris Corporation, Electronic System Division, March 1978.

4. Anderson, D.R., "Data Base Processor Technology," Proc. AFIPS NCC 1975 Vol. 44, pp. 389-395.

5. Anderson, George A. and Jensen, E. Douglas, "Computer Interconnection Structures: Taxonomy, Characteristics, and Examples," Computing Surveys, Vol. 7, No. 4, December 1975, pp. 197-213.

6. Aschim, F.,"Data Base Networks--An Overview," Management Informatics, Vol. 3, No. 1, February 1974, pp. 12-28.

7. Bakkom, D.E. and Behymer J.A., "Implementation of a Prototype Generalized File Translator," ACM-SIGMOD International Conference on Management of Data, May 1975, pp. 99-110.

8. Baum, R.I. and Hsiao, D.K., "Data Base Computers--A Step towards Data Base Utilities," IEEE Transactions on Computers, Vol. C-25, No. 12, December 1976, pp. 1254-1259.

9. Banerjee, Jayanta and Hsiao, David K., "Concepts and Capabilities of a Database Computer," ACM Transactions on Database Systems, Vol. 3, No. 4, December 1978, pp. 347-384.

10. Banerjee, Jayanta and Hsiao, David K., "DBC--A Database Computer for Very Large Databases," IEEE Transactions on Computers, Vol. C-28, No. 3, March 1979.

11. Becker, Hal B., "Let's Put Information Networks Into Perspective," DATAMATION, Vol. 24, No. 3, March 1978, pp. 81-86.

12. Belford, Geneva G., et.al., "Initial Mathematical Model Report-- Research in Network Data Management and Resource Sharing," Center for Advanced Computation (CAC) Document No. 169, University of Illinois at Urbana-Champaign, Urbana, Illinois.

13. Belford, Geneva G., "Network File Allocation--Research in Network Data Management and Resource Sharing," Center for Advanced Computation (CAC) Document No. 203, University of Illinois at Urbana-Champaign, Urbana, Illinois.

14. Belford, Geneva G., "Technology Summary: Research in Network Data Management and Resource Sharing," Center for Advanced Computation, University of Illinois at Urbana-Champaign, Urbana, Illinois, May 1975.

15. Berkowitz, B., "A Distributed Database Management System for Command and Control Applications--Semi-annual Technical Report III," Computer Corporation of America, July 30, 1978.

16. Bernstein, Philip A., et. al., "A Distributed Database Management System for Command and Control Applications--Semi-annual Technical Report I," Computer Corporation of America, July 30, 1977.

17. Bernstein, Philip A., et. al., "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," IEEE Transactions on Software Engineering, Vol. SE-4, No. 3, May 1978, pp. 154-168.

18. Bernstein, Philip A. and Shipman, David W., "A Formal Model of a Concurrency Control Mechanism for Data base Systems," Proc. Berkley Workshop on Distributed Data Bases and Computer Networks, September 1978.

19. Berra, P. Bruce and Oliver, E., "Associative Array Processor Utilization in Data Base Management," Computer, March 1979, pp. 53-61.

20. Berra, P. Bruce, "Data Base Machines," SIGIR Newsletter, Winter 1977, pp. 4-23.

21. Birss, E. W. and Fry, J. P., "Generalized Software for Translating Data," Proc. AFIPS NCC 1976, Vol. 45, pp. 889-897.

22. Booth, G. M., "Distributed Information Systems," Proc. AFIPS NCC 1976, Vol. 45, pp.789-794.

23. Booth, G. M., "The Use of Distributed Data Bases in Information Networks," Proc. First International Conference on Computer Communication: Impacts and Implications, October 1972, pp. 371-376.

24. Bray, O. and Thurber, K. J., "What's Happening with Data Base Processors?", DATAMATION, Vol. 25, No. 1, June 1979, pp. 146-156.

25. Burr, William E. and Gordon, Robert, "Selecting a Military Computer Architecture," Computer, Vol. 10, No. 10, October 1977, pp. 16-23.

26. Canaday, R. E., et. al., "A Backend Computer for Data Base Management," Communications of the ACM, Vol. 17, No. 10, October 1974, pp. 575-582.

27. Casey, R. G., "Allocation of Copies of a File in an Information Network," Proc. AFIPS STCC, Vol. 40, 1972.

28. Cate, Paul E. (Major), "Large-Unit Operational Doctrine," Military Review, Vol. LVIII, No. 12, December 1978, pp. 40-47.

29. Champine, George A., "Four Approaches to a Data Base Computer," DATAMATION, Vol. 24, No. 13, December 1978, pp. 101-106.

30. Champine, George A., "Six Approaches to Distributed Data Bases," DATAMATION, Vol. 23, No. 5, May 1977, pp. 45-48.

31. Chu, Wesley W., "Optimal File Allocation in a Multiple Computer System," IEEE Transactions on Computers, October 1969, pp. 885-889.

32. Chu, Wesley W., "Performance of File Directory Systems for Data Bases in Star and Distributed Networks," Proc. AFIPS NCC 1976, Vol. 45, pp. 577-587.

33. Chu, Wesley W. and Ohlmacher, G., "Avoiding Deadlocks in Distributed Data Bases," Proc. ACM Annual Conference, November 1974, pp. 156-160.

34. Codespoti, D. J. and Maryanski, F. J., "A Microprocessor Distributed Operating System for a Minicomputer," Computer Science Department, Kansas State University, TR CS 78-09, March 1978.

35. Coleman, Aaron H. and Smith, William R., "The Military Computer Family: A New, Joint-Service Approach to Military Computer Acquisition," Computer, Vol. 10, No. 10, October 1977, pp. 12-15.

36. Comba, P. G., "Needed: Distributed Control," Proc. International Conference on Very Large Data Bases, Vol. 1, No. 1, September 1975, pp. 364-375.

37. Computer Corporation of America, "Datacomputer Project Semi-Annual Technical Report (January 1, 1975 to June 30, 1975)."

38. Cotton, Ira W., "Computer Network Interconnection: Problems and Prospects," NBS Special Publication 500-6, April 1977.

39. Curtice, Robert M., "The Outlook for Data Base Management," DATAMATION, Vol. 22, No. 4, April 1976, pp. 46-50.

40. Davenport, R. A., "Distributed or Centralised Data Base," The Computer Journal, Vol. 21, No. 1, February 1978, pp. 7-13.

41. Department of Defense Annual Report Fiscal Year 1979.

42. Deppe, M. E. and Fry, J. P., "Distributed Data Bases: A Summary of Research," Computer Networks, Vol. 1, No. 2, February 1976.

43. Eckhouse, Richard H.,Jr., "Issues in Distributed Processing--An Overview of Two Workshops," Computer, Vol. 11, No. 1, January 1978, pp. 22-26.

44. Ellis, C. A., "A Robust Algorithm for Updating Duplicate Data Bases," Proc. 2nd Berkley Working on Distributed Data Management and Computer Networks, May 1977, pp. 146-158.

45. Elovitz, Honey S. and Heitmeyer, Constance L., "What Is a Computer Network," IEEE 1974 NTC Record, pp. 149-156.

46. Enslow, Philip H., Jr., "What is a 'Distributed' Data Processing System?", Computer, Vol. 11, No. 1, January 1978, pp. 13-21.

47. Falk, Gilbert and McQuillan, John M., "Alternatives for Data Network Architecture," Computer, Vol. 10, No. 11, November 1977, pp. 22-29.

48. Falor, Ken, "'Distributed' Doesn't Mean 'Random'," DATAMATION, Vol. 24, No. 9, September 1978, pp. 226-229.

49. Fisher, Paul S. and Maryanski, Fred J., "Design Considerations in Distributed Data Base Management Systems," Computer Science Department, Kansas State University, TR CS 77-08, April 1977.

50. Fisher, Paul S., Hankley, William J., and Maryanski, Fred J., "porting Software to Multiple Minis: A DBMS Case Study," Computer Science Department, Kansas State University, TR CS 77-12, December 1976.

51. Fisher, Paul S. and Maryanski, Fred J., "Concepts and Problems in Distributed Data Base Management Systems," Computer Science Department, Kansas State University, TR CS 78-17, February 1978.

52. Foster, J. D., "The Development of a Concept for Distributed Processing," Proc. Twelfth IEEE Computer Society International Conference, 1976.

53. Foster, J. D., "Distributed Processing for Banking," DATAMATION, Vol. 22, No. 7, July 1976, pp. 89-90.

54. Fosdick, Harry C., Schantz, Richard E., and Thomas, Robert H., "Operating Systems for Computer Networks," Computer, Vol. 11, No. 1, January 1978, pp. 48-57.

55. Frank, Randall L. and Yamaguchi, Koichi, "A Model for a Generalized Data Access Method," Proc. AFIPS NCC 1974, Vol. 43, pp. 45-52.

56. Fry, J. P., et. al., "Stored Data Description and Data Translation: A Model and Language," Information Systems, Vol. 2, No. 3, March 1977, pp. 95-148.

57. Fry, J. P., Smith, D. P., and Taylor, R. W., "An Approach to Stored Data Definition and Translation," Proc. ACM SIGFIDET Workshop on Data Description and Access, November 1972, pp. 13-55.

58. Fry, J. P. and Sibley, E. H., "Evolution of Data Base Management Systems," ACM Computing Surveys, Vol. 8, No. 1, March 1976, pp. 7-42.

59. Fuller, Samuel H. and Burr, William E., "Measurement and Evaluation of Alternative Computer Architectres," Computer, Vol. 10, No. 10, October 1977, pp. 24-35.

60. Ghosh, S. P., "Distributing a Data Base with Logical Associations on a Computer Network for Parallel Searching," IEEE Transactions on Software Engineering, Vol. SE-2, No. 2, June 1976, pp. 106-113.

61. Goodell, Eugene K., "Control Computer Local Driver Routines in a Functionally Distributed Data Base Management System," Computer Science Department, Kansas State University, TR CS 77-27, December 1977.

62. Greene, William (Major) and Pooch, Udo W., "A Review of Classification Schemes for Computer Communications Networks," Computer, Vol. 10, No. 11, November 1977, pp. 12-21.

63. Hardgrave, W. T., "Distributed Database Technology: An Assessment," Information & Management 1978, pp. 157-167.

64. Heacox, H. C., Cosloy, E. S., and Cohen, J. B., "An Experiment in Dedicated Data Management," Proc. International Conference On Very Large Data Bases, Vol. 1, No. 1, September 1975, pp. 11-13.

65. Healey, David C., "Experimental System Progress Report," Research in Network Data Management and Resource Sharing, Center for Advanced Computation (CAC) Document 209, University of Illinois at Urbana-Champaign, Urbana, Illinois.

66. Helms, Robert F. II, "The Indirect Approach," Military Review, Vol. LVIII, No. 9, September 1978, pp. 2-9.

67. Henson, Carle C., et. al., "Preliminary Design Study for VTS Processing Display Subsystem (Final Report)," International Computing Company, June 1978.

68. Hilsman, William J. (Major General), "C3I Communications Vital in Integration of the Force-Multipliers," ARMY, Vol. 29., No. 3, March 1979, pp. 31-33.

69. Hollaar, Lee A., "Text Retrieval Computers," Computer, Vol. 12, No. 3, March 1979, pp. 40-50.

70. Housel, B. C., Lum, V. Y., and Shu, N. C., "Architecture to an Interactive Migration System (AIMS)," ACM-SIGMOD Workshop on Data Description, Access, and Control, May 1974, pp. 157-170.

71. Hsiao, David K., "Database Machines Are Coming, Database Machines Are Coming," Computer, Vol. 12, No. 3, March 1979, pp. 7-9.

72. Hsiao, D. K., Kanan, K., and Kerr, D. S., "Structure Memory Design for a DataBase Computer," Proc. ACM 77, December 1977, pp. 343-350.

73. Hsiao, David K., Kerr, Douglas S., and Madnich, Stuart E., "Privacy and Security of Data Communications and Data Bases," Proc. Fourth International Conference on Very Large Data Bases, October 1978, pp. 55-67.

74. Jensen, E. Douglas, "The Honeywell Experimental Distributed Processor--An Overview," Computer, Vol. 11, No. 1, January 1978, pp. 28-38.

75. Katzan, Harry Jr., An Introduction to Distributed Data Processing, New York: Petrocelli Books, Inc., 1978.

76. Kent, Stephen T., "Network Security: A Top-Down View Shows Problem," DataCommunications, Vol. 7, No. 6, June 1978, pp. 57-75.

77. Kerr, Douglas S., "DataBase Machines with Large Content-Addressable Blocks and Structured Information Processors," Computer, Vol. 12, No. 3, March 1979, pp. 64-79.

78. Kimbleton, Stephen R. and Mandell, Richard L., " A Perspective on Network Operating Systems," Proc. AFIPS NCC 1976, Vol. 45, pp. 551-559.

79. Kirkley, J. L., "Happiness Is Distributed Processing," DATAMATION, Vol. 24, No. 3, March 1978, p. 79.

80. Kleinrock, Leonard, "On Communications and Networks," IDEEE Transactions on Computers, Vol. C-25, No. 12, December 1976, pp. 1326-1335.

81. Kunii, T. L. and Kunii, H. S., "Design Criteria for Distributed Database Systems," Proc. Third International Conference on Very Large Data Bases, October 1977, pp. 93-104.

82. Liebowitz, B. H. and Carson, J. H., Distributed Processing Tutorial, New York: IEEE, 1977.

83. Lin, C. S., Smith, Diane C. P., and Smith, John M., "Design of a Rotating Associative Memory for Relational Data Base Applications, ACM Transactions on Database Systems, Vol. 1, No. 1, January 1976, pp. 53-65.

84. Liu, S. and Heller, J., "A Record Oriented, Grammar Driven Data Translation Model," ACM-SIGMOD Workshop on Data Description, Access and Control, May 1974, pp. 171-190.

85. Lowental, E. I., "The Backend Computer," MRI System Corp., April 1976.

86. Lowental, E. I., "A Survey--The Application of Data Base Management Computers in Distributed Systems," Proc. Third International Conference on Very Large Data Bases, 1977, pp. 85-92.

87. Mahaffey, Fred K. (Major General), "C3I for Automated Control of Tommorrow's Battlefield," ARMY, Vol. 29, No. 3, March 1979, pp. 26-30.

88. Marill, T. and Stern, D., "The Datacomputer--A Network Data Utility," Proc. AFIPS NCC 1975, Vol. 44, pp. 389-395.

89. Maryanski, Fred J., Fisher, Paul S., and Wallentine, Virgil E., "Evaluation of Conversion to a Back-end Data Base Management System," Computer Science Department, Kansas State University, TR CS 76-08, March 1976.

90. Maryanski, Fred J., "The Management of Redundant Data in a Distributed Data Base," Computer Science Department, Kansas State University, TR CS 78-21, September 1978.

91. Maryanski, Fred J. and Nikravon, Nasrin, "Simulation of a Functionally Distributed Computing Facility: Central System Model," Computer Science Department, Kansas State University, TR CS 78-29, July 1978.

92. Maryanski, Fred J., "Distributed Data Base Management Systems," Computer Science Department, Kansas State University, TR CS 78-19, July 1978.

93. Maryanski, Fred J., Fisher, Paul S., and Wallentine, Virgil E., "Data Access in Distributed Data base Management Systems," Computer Science Department, Kansas State University, TR CS 78-14, December 1978.

94. Maryanski, Fred J., et. al., "A Prototype Distributed DBMS," Computer Science Department, Kansas State University, TR CS 78-08, January 1979.

95. Maryanski, Fred J. and Kreimer, Daniel E., "Effects of Distributed Processing in a Data Processing Environment," Computer Science Department, Kansas State University, TR CS 77-23, November 1977.

96. Maryanski, Fred J., et. al., "Distributed Data Base Management Using Minicomputers," Computer Science Department, Kansas State University, TR CS 77-22, October 1977.

97. Maryanski, Fred J., "Performance of Multi-Processor Back-end Data Base Systems," Computer Science Department, Kansas State University, TR CS 77-07, April 1977.

98. Maryanski, Fred J. and Fisher, Paul S., "Rollback and Recovery in Distributed Data Base Management Systems," Computer Science Department, Kansas State University, TR CS 77-05, February 1977.

99. Maryanski, Fred J., "A Deadlock Prevention Algorithm for Distributed Data Base Management Systems," Computer Science Department, Kansas State University, TR CS 77-02, February 1977.

100. Maryanski, Fred J., "A Survey of Developments In Distributed Data Base Management Systems," Computer Science Department, Kansas State University, TR CS 77-01, January 1977. (Another version of this TR can be found in Computer, Vol. 11, No. 2, pp. 28-38.)

101. Maryanski, Fred J., "Memory Management in Distributed Data Base Systems," Computer Science Department, Kansas State University, TR CS 76-14, October 1976.

102. Maryanski, Fred J., "Language Specification for a Distributed Data Base Management System," Computer Science Department, Kansas State University, TR CS 76-13, May 1976.

103. Maryanski, Fred J. et. al., "Distributed Data Base Management Using Minicomputers," INFOTECH State of the Art Report Minis Verses Mainframes, 1978, pp. 141-157.

104. Maryanski, Fred J., et. al., "A Minicomputer Based Distributed Data Base Management System," Proc. IEEE NBS Trends and Applications Symposium: Micro and Mini Systems, May 1976, pp. 113-117.

105. Maryanski, Fred J., Norsworthy, Kevin E., and Norsworthy, Kirk S., " A System Architecture for Distributed Data Base Management," Computer Science Department, Kansas State University, TR CS 78-15, April 1978

106. Maryanski, Fred J. and Wallentine, Virgil E., "A Simulation Model of a Back-end Data Base Management System, Proc. Pittsburgh Conference on Modeling and Simulation, April 1976, pp. 243-248.

107. McClellan, Stephen T., "Distributed and Small Business Computing--A Fast Track," DATAMATION, Vol. 25, No. 6, May 25, 1979, pp. 124-127.

108. Menasce, Daniel A., Popek, Gerald J., and Muntz, Richard R., "Centralized and Hierarchical Locking in Distributed Databases," Proc. 2d Berkley Workshop on Distributed Data Management and Computer Networks, May 1977, pp. 178-195.

109. Meredith, Dennis L., "Lasercom Speeds Unjammable Messages Through Space on Light Beams," Popular Science, Vol. 215, No. 5, November 1979, pp. 86-88.

110. Merten, A. G. and Fry, J. P., "A Data Description Language Approach to File Translation," ACM-SIGMOD Workshop on Data Description, Access and Control, May 1974, pp. 191-205.

111. Miller, Myron, "A Survey of Distributed Data Base Management," Information & Management 1978, pp. 243-264.

112. Morgan, D. E., Taylor, D. J., and Custeau, G., "A Survey of Methods for Imporoving Computer Network Reliability and Availability," Computer, Vol. 10, No. 11, November 1977, pp. 42-50.

113. Morgan, Howard L. and Levin, K. Dan, "Optimal Program and Data Locations in Computer Networks," Communications of the ACM, Vol. 20, No. 5, May 1977, pp. 315-322.

114. Morris, P. and Sagalowicz, D., "Managing Network Access to a Distributed Data Base," Proc. 2d Berkley Workshop on Distributed Data Management and Computer Networks, May 1977, pp. 58-67.

115. Navathe, S. B. and Merten, A. G., "Investigation into the Application of the Relational Model to Data Translation," ACM-SIGMOD International Conference on Management of Data, May 1975, pp. 123-138.

116. Needham, Roger M. and Schroeder, Michael D., "Using Encryption for Authentication in Large Networks of Computers," Communications of the ACM, Vol. 21, No. 12, December 1978, pp. 993-999.

117. Ozkarahon, E. A., Schuster, S. A., and Smith, K. C., "RAP - An Associative Processor for Data Base Management," Proc. AFIPS NCC 1975, Vol. 44, pp 379-388.

118. Peebles, R. and Manning, E., "A Computer Architecture for Large (Distributed) Data Bases," Proc. International Conference on Very Large Data Bases, Vol. 1, No. 1, 1975, pp. 405-427.

119. Peebles, R. and Manning, E., "System Architecture for Distributed Data Management," IEEE Transactions on Computers, Vol. 11, No. 1, January 1978, pp. 40-47.

120. Person, Ron, "How TI Distributes Its Processing," DATAMATION, Vol. 25, No. 4, April 1979, pp. 98-103.

121. Pliner, M., McGowan, L., and Spalding, K., "A Distributed Data Management System for Real-Time Application," Proc. 2d Berkley Workshop on Distribued Data Management and Computer Networks, May 1977, pp. 68-86.

122. Porrera, David P., "New Tactics and Beyond," Military Review, Vol. LIX, No. 5, May 1979, pp. 21-29.

123. Ramamoorthy, G. S., Ho, T. K., and Wah, B. W., "Architectural Issues in Distributed Data Base Systems," Proc. Third International Conference on Very Large Data Bases, October 1977, pp. 121-126.

124. Ramirez, J. A., Rin, N. A., and Prywes, N. S., "Automatic Generation of Data Conversion Programs using a Data Description Language," ACM-SIGMOD Workshop on Data Description, Access and Control, May 1974, pp. 207-226.

125. Rehme, Erwin and Wallentine, Virgil E., "MIMICS (Asynchronous) Control Line Protocol," Computer Science Department, Kansas State University, TR CS 77-15, January 1978.

126. Rosenberg, Alan P., "Building Block Approach Shows Structure of Network Programs," DataCommunications, Vol. 7, No. 9, September 1978, pp. 79-90.

127. Rosenkrantz, D.J., Stearns, R.E., and Lewis, P.M., "System Level Concurrency Control for Distributed Database Systems," ACM Transactions on Database Systems, Vol. 3, No. 2, June 1978, pp. 178-198.

128. Rothnie, James B. and Goodman, Nathan, "A Survey of Research and Development in Distributed Database Management," Proc. Third International Conference on Very Large Data Bases, October 1977, pp. 48-61.

129. Rothnie, James B. and Goodman, Nathan, "An Overview of the Preliminary Design of SDD-1: A System for Distributed DataBases," Proc. 2d Berkley Workshop on Distributed Data Management and Computer Networks, May 1977, pp. 38-57.

130. Rothnie, James B., Goodman, Nathan, and Bernstein, Philip A., "The Redundant Update Algorithm of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," First International Conference on Computer Software and Applications, November 1977.

131. Sanders, Ray W., "Comparing Networking Technologies," DATAMATION, Vol. 24, No. 7, July 1978, pp. 88-94.

132. Schneider, G. M. and Desautels, E. J., "Design of a File Translation Language for Networks," Information Systems, Vol. 1, No. 1, January 1975, pp. 23-31.

132. Schneider, G.M., "DSCL--A Data Specification and Conversion Language for Networks," ACM-SIGMOD International Conference on Management of Data, May 1975, pp. 139-148.

134. Schultz, Brad, editor, "Special Report--The Move to Distributed Processing," COMPUTERWORLD, July 1979.

135. Shepherd, Alan J., "A British Example of Distributed Computing," DATAMATION, Vol. 24, No. 3, March 1978, pp. 87-91.

136. Shepherd, Mark Jr., "Distributed Computing Power: A Key to Productivity," Computer, Vol. 10, No. 11, November 1977, pp. 66-74.

137. Shoshani, A., "A Logical Level Approach to Data Base Conversion," ACM-SIGMOD International Conference on Management of Data, May 1975, pp. 112-122.

138. Shoshani, A. and Brandon, K., "On the Implementation of a Logical Data Base Converter," Proc. International Conference on Very Large Data Bases, Vol. 1, No. 1, September 1975, pp. 529-531.

139. Shu, Nan C., et. al., "CONVERT: A High Level Translation Definition Language for Data Coversion," Communications of the ACM, Vol. 18, No. 12, December 1975, pp. 557-567.

140. Sibley, E. H. and Taylor, R. W., "A Data Definition and Mapping Language," Communications of the ACM, Vol. 16, No. 12, December 1973, pp. 750-759.

141. Slonim, Jacob, Farrell, Michael W., and Maryanski, Fred J., "Highlights of a Survey of Data Base Management Systems," Computer Science Department, Kansas State University, TR CS 77-24, September 1977.

142. Slonim, Jacob, Schmidt, David, and Fisher, Paul, "Considerations for Determining the Degree of Centralization or Decentralization in the Computing Environment," Computer Science Department, Kansas State University, May 1978.

143. Slonim, Jacob, Unger, Elizabeth A., and Fisher, Paul S., "Data Base Management System Environments Present and Future," Computer Science Department, Kansas State University, May 1977.

144. Smith, Diane C.P. and Smith, John M., "Relational Data Base Machines," Computer, Vol. 12, No. 3, March 1979, pp. 28-38.

145. Starry, Donn A. (General), "A Tactical Evolution--FM 100-5," Military Review, Vol. LVIII, No. 8, pp. 2-11.

146. Stonebraker, Michael, "Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES," Electronic Research Laboratory Memorandum No. UCB/ERL M78/24, May 24, 1978, pp. 1-24.

147. Straut, Robert P., "Fighting Outnumbered and Winning," Military Review, Vol. LIX, No. 5, May 1979.

148. Stubblebine, Albert N.,III (Brigidier General), "C3I for Automated Focus on Intelligence Picture," ARMY Vol. 29, No. 3, pp. 34-36.

149. Su, Stanley Y.W., "Cellular-Logic Devices: Concepts and Applications," Computer, Vol. 12, No. 3, March 1979, pp. 11-25.

150. Su, Stanley Y. W. and Lipovski, G. Jack, "CASSM: A Cellular System for Very Large Data Bases," Proc. International Conference on Very Large Data Bases, Vol. 1, No. 1, September 1975, pp. 456-472.

151. Su, Stanley Y.W. and Lam, H., "A Semi-Automatic Data Translation Scheme for Achieving Data Sharing in a Network Environment," ACM-SIGMOD Workshop on Data Description, Access and Control, May 1974, pp. 227-247.

152. Sykes, D.J., "Positive Personal Identification," DATAMATION, Vol. 24, No. 11, November 1, 1978, pp. 177-186.

153. Tajibnapis, William D., "A Correctness Proof of a Topology Information Maintenance Protocol for a Distributed Computer Network," Communications of the ACM, Vol. 20, No. 7, July 1977, pp. 477-485.

154. The Department of the Army Manual, January 1977.

155. Thomas, Robert H., "A Solution to the Concurrency Control Problem for Multiple Copy Data Bases," COMPCON 1978 Digest of Papers, pp. 88-94.

156. Unger, Elizabeth A., et. al., "Integration of a DBMS into a Network Environment," Computer Science Department, Kansas State University, April 1979.

157. US Army Command and General Staff College, "Unified Direction of the Armed Forces," PT 101-2, March 1978.

158. Van Rensselaer, Cort, "Centralize? Decentralize? Distribute?", DATAMATION, Vol. 25, No. 4, April 1979, pp. 88-97.

159. Wald, Bruce and Salisbury, Alan, "The Computer Family Architecture Project: Service Perspectives and Overview," Computer, Vol. 10, No. 10, October 1977, pp. 8-11.

160. Wallentine, Virgil E., "Project Report for Functionally Distributed Computer Systems Development: Software and Systems Structure, Part I," Kansas State University, October 1977.

161. Wallentine, Virgil E., et. al., "MIMICS Design Overview Functionally Distributed Computer Systems Development: Software and System Structure," Computer Science Department, Kansas State University, TR CS 77-4, December 1976.

162. Wallentine, Virgil E. and Maryanski, Fred J., "Implementation of a Distributed Data Base System," Computer Science Department, Kansas State University, TR CS 75-11, November 1975.

163. Wendt, Robert L. (Lieutenant Colonel), "Army Training Development--A Quiet Revolution," Military Review, Vol. LVIII, No. 8, August 1978, pp. 74-83.

164. White, James E., "A High-Level Framework for Network-Based Resource Sharing," Proc. AFIPS NCC 1976, Vol. 45, pp. 561-570.

165. Whitney, V.K.M., "Fourth Generation Data Management Systems," Proc. AFIPS NCC 1973, Vol. 42, pp. 239-244.

166. Wong, Eugene, "Retrieving Dispersed Data from SDD-1: A System for Distributed Databases," Proc. 2d Berkley Workshop on Distributed Data Management and Computer Networks, May 1977, pp. 50-68.

167. Wu, Frederick H., "Distributed DP Provides for Data Needs," COMPUTERWORLD, July 30, 1978, p. SR/3.

168. Yamaguchi, K., and Merten, A.G., "Methodology for Transfering Programs and Data," ACM-SIGMOD Workshop on Data Description, Access and Control, May 1974, pp. 141-155.

DATA EXCHANGE BETWEEN FULLY DISTRIBUTED HETEROGENEOUS MILITARY SUPER NETWORKS

by

THOMAS RUSSELL KELLY

A.B., Wofford College, 1964

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas
1980

ABSTRACT

This Master's Report concentrates on four methods for exchanging data (data conversion/translation) between two distinct (independently developed) fully distributed heterogeneous networks of networks or super networks. While this report focuses on the problem of data exchange between super networks in a military environment, the discussion is equally applicable to systems designers and users who must interface any independently developed fully distributed heterogeneous networks or super networks.

The larger problem of strategic/tactical intelligence interface provides the overall context for the discussion in this report. Although the origins of this strategic/tactical interface problem can be traced to the pre-Vietnam era, it has been a key topic of concern/discussion within the Army and DOD for only the last seven or eight years. Unfortunately, many aspects of the problem have received little more than cursory attention. Nowhere is this more evident than in computer systems development. The strategic intelligence producers have long maintained their intelligence holdings in automated files/data bases. Many of their computer systems have already been linked together into loosely federated networks. Provided the Congressional mandate to delegate strategic intelligence production responsibilities continues and distributed DBMS's are perfected, a fully distributed

heterogeneous super network composed of all strategic intelligence producers could be a reality as early as the mid-1980's. Unlike their strategic counterparts, the tactical intelligence producers currently have no automated support. This situation should change drastically by the mid-to-late 1980's, when systems currently in the study or early developmental phases are scheduled to be fielded. These systems will eventually also evolve into super networks. It will be at this juncture that the problem of interfacing the super networks must be solved. To date, little, if anything has been done to insure that these two evolving super networks can interface. The notional super networks that are described in this report will simulate both the projected strategic and the projected tactical super networks that are likely to be operational in the mid-to-late 1980's.

The following methodologies with their attendant advantages and disadvantages are discussed in this report: (1) the "brute force" approach, i.e., a unique translator for each pair of data base systems that will interface across the two super network boundaries; (2) the University of Michigan's Data Translation Project approach, i.e., the Stored Data Definition Language (SDDL); (3) the Data Specification and Conversion Language (DSCL) method proposed by G. M. Schneider; and (4) the ARPA network Datacomputer Datalanguage approach.