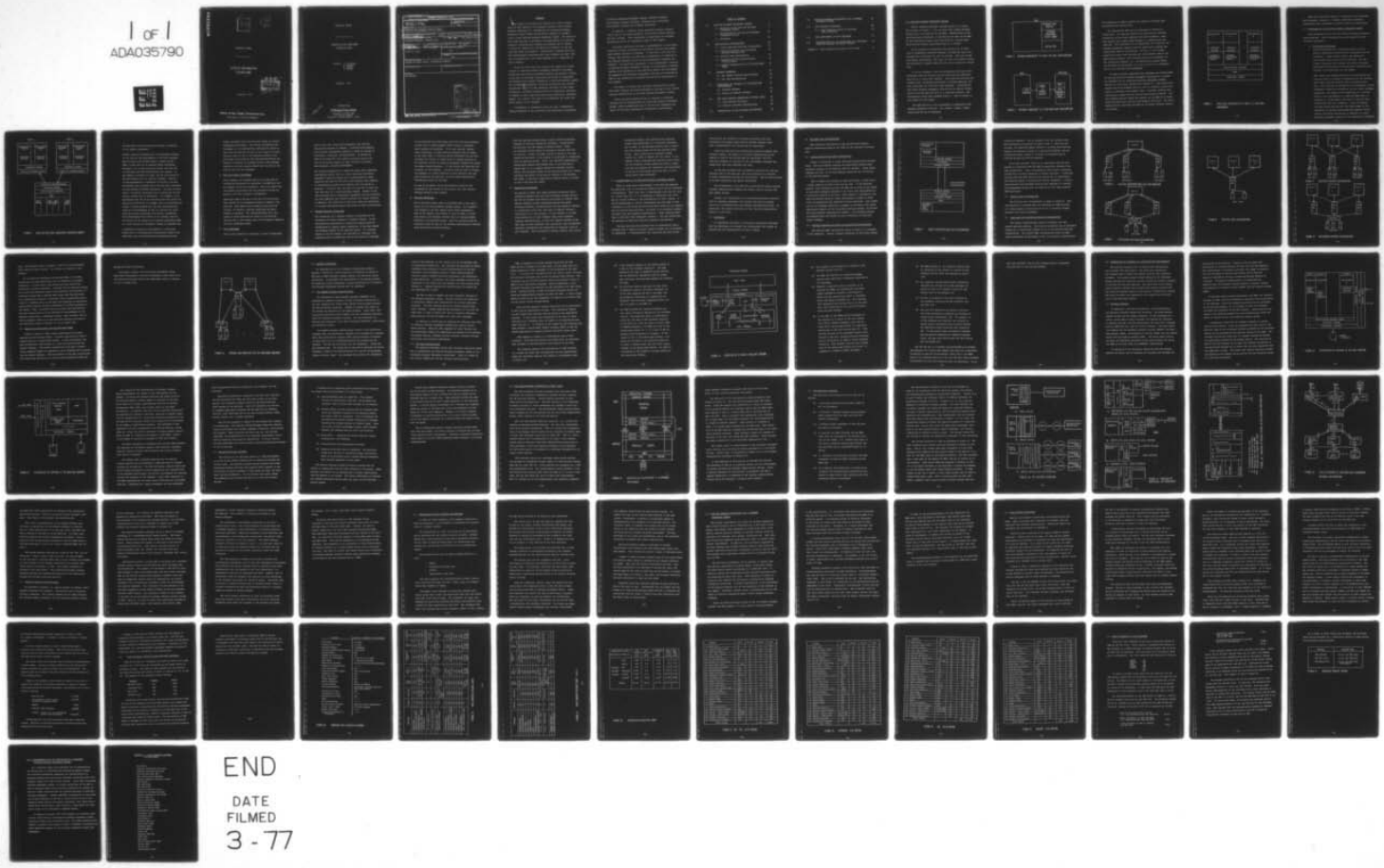
AD-A035 790 CULLINANE CORP WELLESLEY MASS F/G 9/2
COMMERCIAL DATA MANAGEMENT PROCESSOR STUDY.(U)
DEC 75 J CULLINANE, R GOLDMAN, T MEURER

UNCLASSIFIED NL

1 OF 1
ADA035790



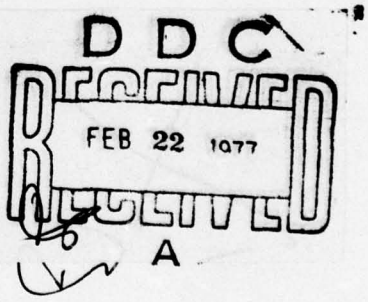
END

DATE
FILMED
3-77

(2)
NW

TECHNICAL REPORT

COMMERCIAL DATA MANAGEMENT PROCESSOR STUDY



DECEMBER 1975

Specialists in Database Software

TECHNICAL REPORT

COMMERCIAL DATA MANAGEMENT

PROCESSOR STUDY

Authors: J. Cullinane
R. Goldman
T. Meurer
R. Nawara

December 1975

Prepared by:

Cullinane Corporation

WELLESLEY OFFICE PARK
20 WILLIAM STREET
WELLESLEY, MASSACHUSETTS 02181

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Department of Defense
Washington, D. C. 20301

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
N/A

3. REPORT TITLE
(6) Commercial Data Management Processor Study.

(9) DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Technical Report.

(10) J. Cullinane, R. Goldman, T. Meurer R. Nawara

5. AUTHOR(S) *(First name, middle initial, last name)*
Cullinane Corporation - Wellesley, Mass.

6. REPORT DATE
(11) December 1975

(12) 84p.

7a. TOTAL NO. OF PAGES
77

7b. NO. OF REFS
N/A

8a. CONTRACT OR GRANT NO.

b. PROJECT NO. 31100-

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
N/A

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
N/A

10. DISTRIBUTION STATEMENT
Approved for public release. Distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT
See foreword

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

392 816

DD FORM 1 NOV 68 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

Security Classification

FOREWORD

This report will present the results of a study to determine the most promising mini-computer processor using current hardware technology for implementing the Integrated Database Management System (IDMS) and ultimately leading to a commercially viable back-end database management system. The intent of this study has been to determine the requirements of a back-end processor including both hardware and software and to analyze a variety of commercially available mini-computers to see how they meet these requirements. Certain data collected from hardware manufacturers concerning both their hardware and their software is documented in this study together with a comparison of their products.

It must be noted that this study is the result of a mutual interest in back-end database management systems by Cullinane Corporation and various government agencies and reflects a desire on the part of Cullinane Corporation and others to investigate the technical and economic feasibility of back-end database management systems. Consequently, Cullinane Corporation has invested significant time of its key management personnel in this study. In addition, Cullinane Corporation has let substantial contracts in support of this study with Kansas State University, Manhattan, Kansas. As a result, the costs in the aggregate far exceed the dollar amount of the contract.

A discussion is presented to give the user a fundamental working knowledge of the qualitative and quantitative approaches

to back-end database management systems, database software, mini-computer systems evaluation, selection and interfacing, and hardware, software and firmware integration.

In addition, a separate letter regarding Cullinane Corporation's view of the market potential of back-end database machines has been included. This information is proprietary to Cullinane Corporation and is not considered a part of this report.

Cullinane Corporation has made a recommendation in this report regarding the best candidates for successful back-end data management systems including a prototype. In arriving at any such conclusion, many factors are taken into consideration, in addition to the relative technical merits of the manufacturer's hardware and software. Of significant importance is the manufacturer's interest in back-end database management systems, his willingness to develop systems to be responsive to this potential market, his willingness to cooperate with Cullinane Corporation, the ease of conversion of IDMS to the manufacturer's hardware and the market penetration of his products.

The reader is advised that Cullinane Corporation has found both great interest and some disinterest on the part of the various computer manufacturers in back-end database management systems.

Since Cullinane Corporation has an interest in following through with the implementation of a back-end database management system, these considerations are of the greatest importance and impact on Cullinane Corporation's conclusions significantly.

TABLE OF CONTENTS

1.0 BACK-END DATABASE MANAGEMENT CONCEPT

Today's database management systems operate in a single CPU as shown in Figure 1. All run-units that request database service are in the same CPU as the DBMS. Communication across partition/region boundaries is provided to one copy of the DBMS so that concurrent access can be monitored on a task-by-task basis and main memory requirements kept to a minimum.

As an organization integrates more and more of its data into an independent corporate database, the DBMS may need to be resident for a good part (or all) of the day. Real and virtual main memory requirements, CPU time, I/O time, and channel traffic and contention to support DBMS activity can be viewed as a single cost.

It is not uncommon to see an installation with heavy teleprocessing activity off-load the T/P receiving and scheduling activity to a "front-end" computer. This front-end monitors the outside world, establishes priorities in message queueing, and only interrupts the "host CPU" when necessary. Installations with heavy database management activity may in similar fashion wish to off-load that activity to a "back-end" machine. The configuration would then resemble Figure 2. Such a concept is the subject of this report.

The back-end CPU in this arrangement is dedicated to the database management function. As a result, a small, rather specialized CPU can be employed.

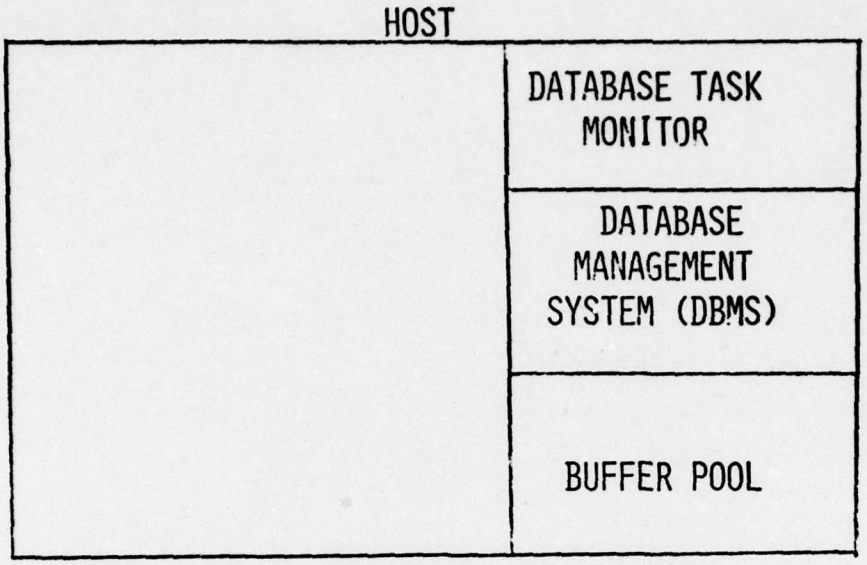
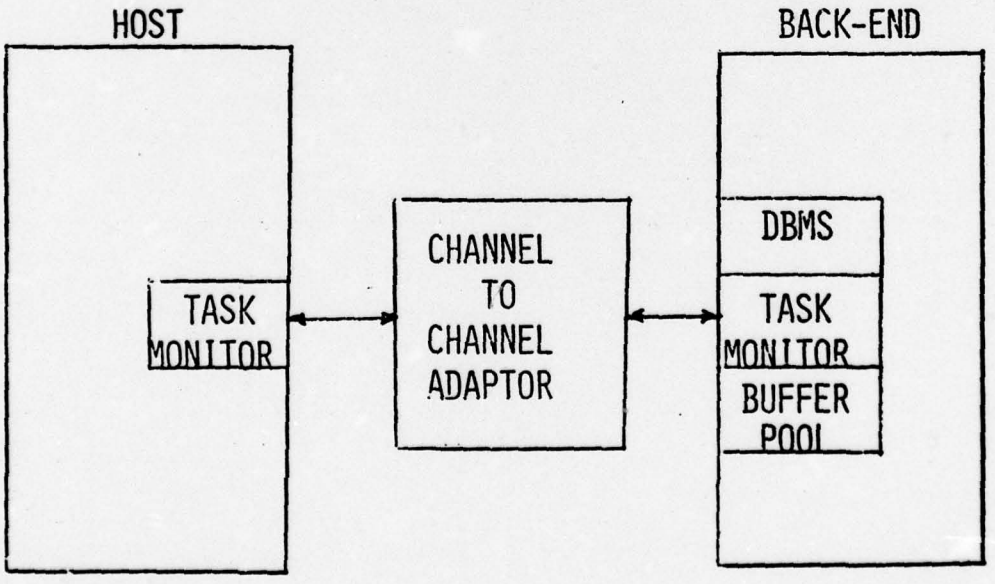


FIGURE 1 DATABASE MANAGEMENT IN A HOST CPU ONLY CONFIGURATION



FUGURE 2 DATABASE MANAGEMENT IN A HOST/BACK-END CONFIGURATION

Mini-computers in today's market are capable of handling this specialized task quite easily.

The back-end CPU need not be restricted to servicing a single host. The back-end computer can serve as the interface between one or many host computers and their databases. Each host machine will require some software to interface with the back-end. This interface will be responsible for collecting the database management requests from the application programs and transmitting them to the back-end. In turn, it will accept results and status from the back-end and distribute them to the application programs. This interface should be transparent to the application program, i.e., the application program should not be coded any differently if it is to be run on a standalone CPU or in a host/back-end arrangement.

In order to better understand the advantages and disadvantages of the host/back-end database management configuration, consider the placement of the functional database management software modules in a host-only configuration (Figure 3). Each application program requiring database service links an interface routine with itself to intercept the database calls. Interface routines then concurrently request service from the multi-tasking monitor. The multi-tasking monitor passes the requests to the DBMS after determining and resolving conflicts and deadlock situations. The multi-tasking monitor also maps the DBMS to the proper database description tables.

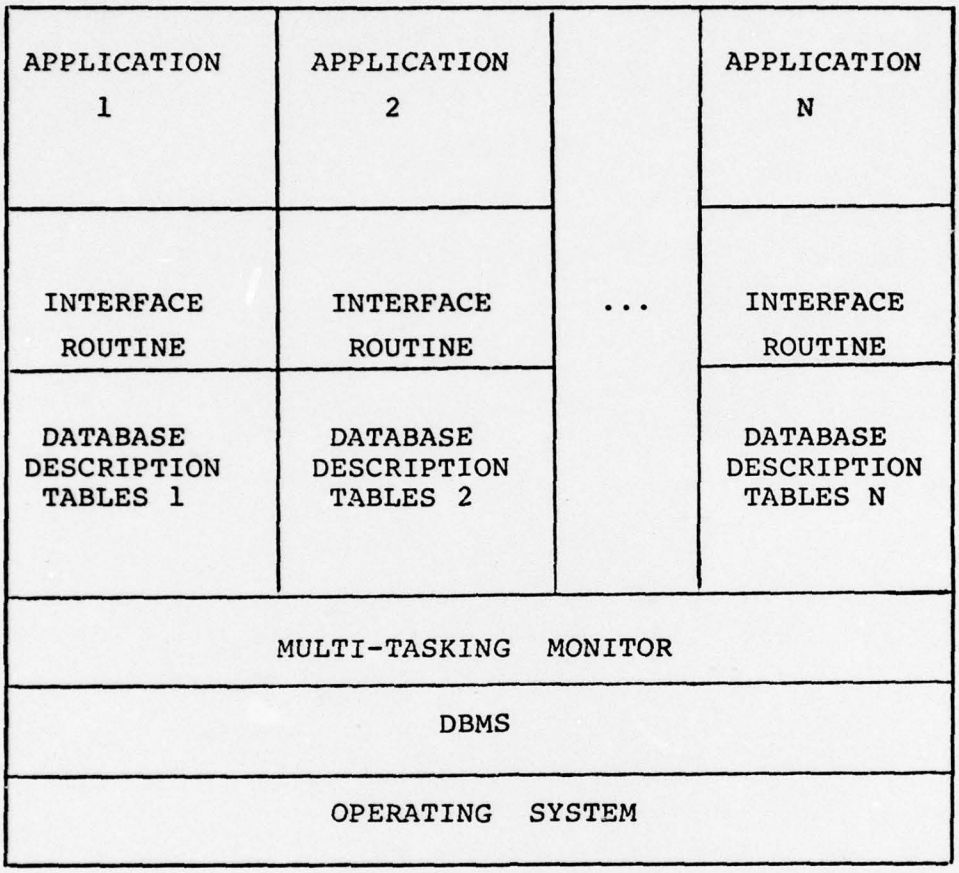


FIGURE 3 FUNCTIONAL COMPONENTS OF A DBMS IN A HOST-ONLY ARRANGEMENT

These same functional modules are required in the host/back-end arrangement (Figure 4). However, additional functional modules have been introduced to provide inter-CPU communication.

1.1 Advantages of the Back-End Database Management System

Many advantages accrue from the back-end database management system approach. These advantages have been grouped into the following categories:

A. Performance Advantages

System performance can be characterized by a set of external observable factors such as turnaround time and throughput. These factors reflect the behavior of other factors such as CPU and I/O activity. The subsequent discussion will consider the effect on system performance of both sets of factors when a back-end minicomputer is incorporated into the system.

Each request for database service in an application program may result in several I/O operations to the database. Each access requires some CPU time to operate on the data. This CPU time is consumed by file management, device management, protection procedures, and the actual database operations. The number of I/O operations is a critical factor in the performance of a back-end DBMS since it dictates to a large degree the amount of time the host program must wait for a response. Since I/O devices are much slower than the CPU, the total number of I/O operations in the back-end becomes the dominant factor. Overall I/O device utilization is improved in a host/back-end arrangement because of the separation of database

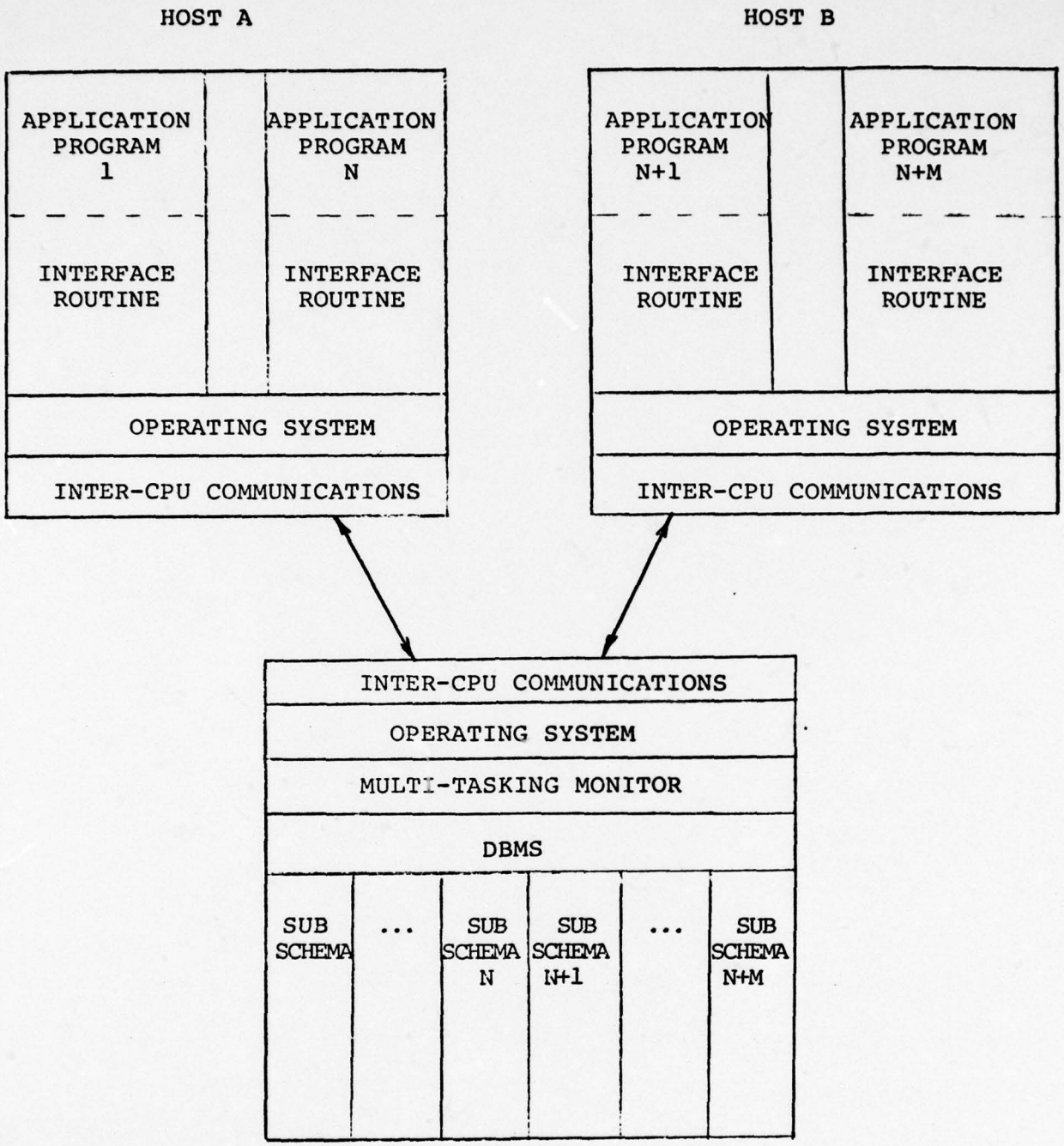


FIGURE 4 BACK-END DATA BASE MANAGEMENT PROCESSOR CONCEPT

I/O and other I/O activity which provides a reduction in I/O channel contention.

Another important consideration in performance analysis is the level of multiprogramming in the host processor. When an application program makes a request of the DBMS, it must wait for a response before proceeding. Therefore, in a single partition system, the host CPU is idle while the back-end processes the request. If the number of accesses is large, the CPU utilization of the host machine will be severely degraded. However, if multiprogramming is supported in the host, the host can process other programs while the back-end is performing the necessary database operations. The most critical factor influencing turnaround time is maximizing the CPU overlap between the two processors. For example, if an application uses 75% of the available host CPU cycles for execution of which 25% is in DBMS, then in a host/back-end system, a second program with the same characteristics could be multiprogrammed in the host against the first. Although process turnaround time would be lengthened, multi-programming would result in an increase from 50% utilization of the system to utilization approaching 100% or a near doubling of throughput (number of programs run).

A comparative analysis of performance in a host-only system and in a host/back-end configuration leads to the conclusion that the host/back-end configuration gives

higher throughput than the host-only system in a multi-programming environment. The overall performance and throughput in the host/back-end configuration would be best when the host CPU is able to give each of its processing cycles to one of the programs running in its main memory and similarly that the back-end machine can respond to requests for database service from programs running in the host quickly enough that the host is not waiting idle for the responses.

B. Host Main Memory Advantages

Since requests for database service are being sent to the back-end processor for execution, the DBMS will not be resident in the host processor. This will reduce the main memory requirements and the processing cycles required of the host processor considerably.

Additional memory savings in the host can be realized when control of all peripheral devices is moved to the back-end. Host memory which normally would hold I/O drivers, I/O buffers, spooling algorithms and spool buffers is released. The cost/performance ratio of a system in which functions are moved to the back-end improves due to the cheaper cost of mini-computer memories and their comparable speed.

C. Cost Advantages

Since a mini-computer is currently an order of magnitude

less in cost than large scale processors, the cost per instruction executed is reduced. A back-end mini-computer also provides considerable benefits in terms of economy of procurement, operation, and maintenance. By moving the DBMS to the back-end, host machine resources such as CPU cycles and primary memory are made available to process more programs, increasing throughput.

The inherent modularity of a back-end system makes upgrading considerably easier than in a large mainframe environment. Upgrading a back-end system can be accomplished by either replacing the back-end CPU with a more powerful mini-computer or incorporating another mini-computer into the system in parallel. In either case, the cost of a new CPU is again far less than that of a large mainframe upgrade. Because of the lower purchase price, maintenance agreements for mini-computers are less expensive than similar contracts for larger machines. In addition, the simplicity of the mini's make the possibility of in-house maintenance quite attractive for many installations.

D. Database Recovery Advantages

The reliability of a computer system is threatened by the occurrence of either hardware or software failure. In the host/back-end configuration, transaction logging could be accomplished by logging input transactions on the host system and database updates on the back-end system. If a failure occurs in the host system, the back-end can "rollback" any currently active transactions and log the failure of the host.

If the back-end were functioning with only one host processor, at this time it could quiesce itself forming a checkpoint where restarts are possible. If it were serving several other processors, it would continue handling them as normal. On the other hand, if the host system determines the back-end processor has failed, it could cease to request services of the back-end, notify all users of necessary shutdown and inform the master console operator. If the failure has destroyed the integrity of the database, utilities could be used to restore the database to a known condition at some previous time and, upon restarting the back-end, the host could reprocess all transactions using its log tape.

In view of the above, two or more machines would be less influenced by the failing of one system, and, upon failure, recovery should be more orderly.

E. Security Advantages

With a back-end system, data is available only to the application via the inter-computer message system. If an application program in the host system should abend requesting a dump of the system's main memory at time of abend, it could only obtain a few database records since all system buffers would be in the back-end system. Data set allocation and deallocation would be done by the database administration function using the back-end system directly.

With the back-end machine being a single function processor, breaches of security become more difficult. Authorization can be built into the system at several levels. First, all necessary authorization to process on the host CPU must be obtained. Second, the DBMS interface routine of the host system can determine if the program is authorized to communicate with the back-end system. Third, the inter-CPU communication system would examine each message sent to the back-end to ensure that it is a standard request for database service. Fourth, the subschema tables in the back-end system will control retrieval and update of any and all records in the database. And fifth, if desired, no application programs would be allowed to run in the back-end itself.

F. Shared Data Advantages

The ability to share data among different processors would eliminate redundant data and extra secondary storage devices. The CPU and elapsed time saved by not having to transfer data between computer systems--physically moving disk packs, switching secondary storage devices from one system to another, or transmitting files over some type of computer network--would result in a cost savings. If the transfer is to take place between two or more computer types using different internal representations, or even two computer systems of the same type but with different file structures, then considerable character reformatting and translation is required in one or more machines. With the back-end concept, however, data can be

transferred between very different host machines without much difficulty (if a restricted character set is used) as the back-end would be able to determine the desired format and do the required translating. This translating of character coding schemes (eg. ASCII to EBCDIC and vice versa) is considered a major advantage when multiple hosts utilizing the different coding schemes are using a single back-end and replaces the translation and duplication of entire file structures each time data is to be shared among those processors.

1.2 Disadvantages of the Back-End Database Management System

There are some major disadvantages to the back-end approach. The additional cost of the back-end machine itself can be considered a disadvantage. While this cost is considerably less than the cost of a larger host CPU, it is nonetheless an additional cost which may be a severe problem in the marketability of such a system. A second disadvantage is that an organization may have made a large investment in disk drives and disk controllers which are not compatible with the back-end processor, and there is no interface hardware available from mini-computer manufacturers. (Some interface hardware is available from independent vendors.) This may necessitate the purchase of additional back-end compatible peripherals.

The host and back-end processors may be separated by such a distance that a channel-to-channel adapter between the two machines is impossible. A teleprocessing line connection may then be sub-

stituted but the difference in channel-to-channel and teleprocessing line speeds would severely degrade response times. Also, teleprocessing line charges may be significant.

Since the host/back-end configuration requires hardware from multiple vendors, the danger of circular fingerpointing by the vendors in case of any failure must be considered. The fact that more hardware is required in this arrangement increases the probability that some component will fail.

If the host processor uses a different character set from the character set of the back-end, then data conversion is required of every transmission between the two machines. The CPU time required to do this translation may become significant.

The introduction of the inter-CPU communication system requires reliable communications hardware and offsets some of the host CPU main memory savings.

Finally, the introduction of a new hardware/software architecture for database management will require some new and special skills to be developed among the database administration staff. In particular, the control and tuning of the inter-CPU communication system is a new skill.

1.3 Conclusion

In considering the above, the Cullinane Corporation concludes that the advantages far outweigh the disadvantages and intends to proceed with the implementation of such a system.

2.0 HOST/BACK-END CONFIGURATIONS

Many different connections of host and back-end machines could be constructed based on the needs of the end-user to process data.

2.1 Single Dedicated Back-End Configuration

Figure 5 illustrates a single dedicated host/back-end configuration. The host computer executes the application program which generates requests for information from the database. These access requests are sent via the host message system and the I/O devices to the back-end system.

Once a request has reached the back-end system, it must attain a unique identity so that it will not be lost. In the back-end machine, these requests activate the DBMS and in doing so are queued as individual messages belonging to an active task corresponding to a particular program in the host. Each request is scheduled by the back-end message system to access the database a number of times until the appropriate DML operation has been performed. Upon completion of the database operation, a message is returned to the proper application program via the intervening message systems, I/O drivers and inter-computer communications channel. Such messages consist of requested data and success or error conditions on store and search requests.

2.2 Multiple Dedicated and Distributed Back-End Configuration

The back-end DBMS configuration shown in Figure 2 is composed of two computers. However, several extensions of this basic config-

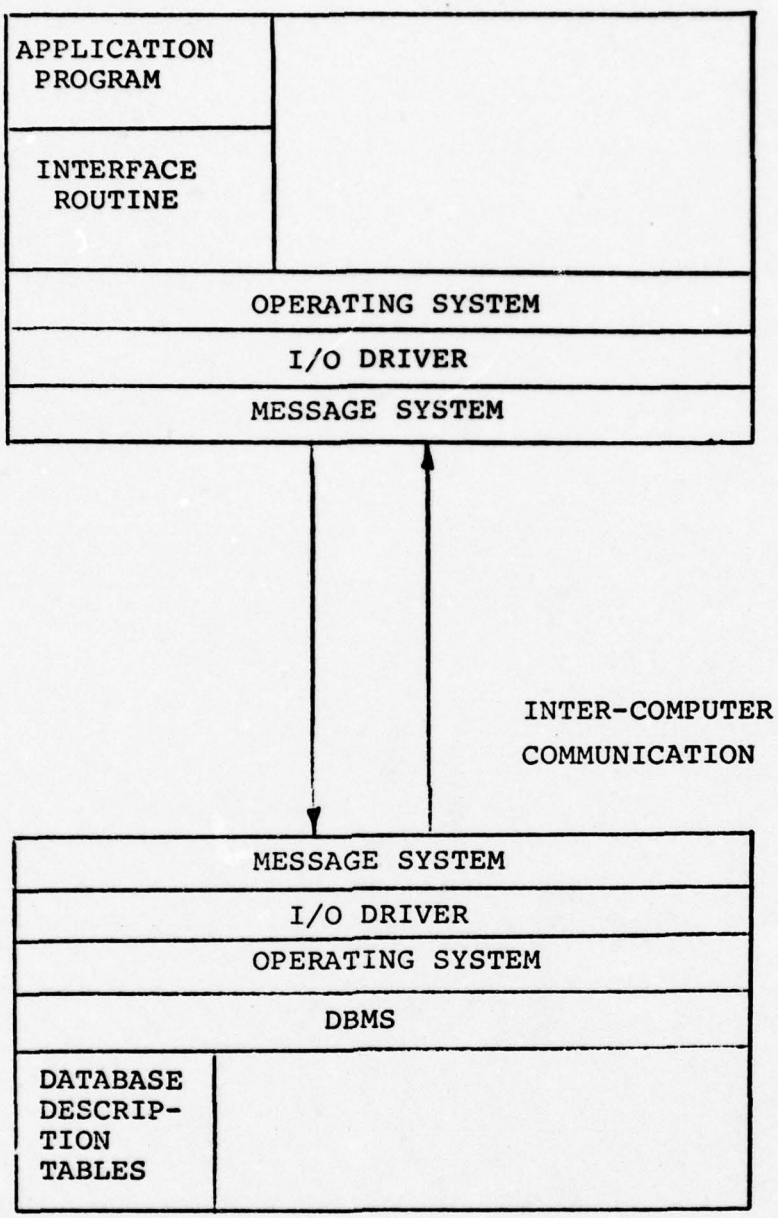


FIGURE 5 SINGLE DEDICATED BACK-END CONFIGURATION

uration are possible. Let us consider first the multiple back-end configuration portrayed in Figures 6 and 7. There are two versions, the dedicated system (Figure 6) in which each back-end computer has exclusive access to a particular database, and the distributed system (Figure 7) in which all databases may be accessed through any back-end machine.

As in most networks, there is no restriction that all components of a multiple back-end DBMS be physically located at the same installation. Such a configuration would allow a centrally located host to access databases in remote locations. A dedicated multiple back-end DBMS would be best suited for an installation with several remote or functionally distinct databases. A distributed multiple back-end DBMS would be best employed in a heavily multiprogrammed system where the majority of the tasks involved DBMS operations.

2.3 Multiple Host Configuration

The multiple host configuration is shown in Figure 8. This configuration is intended when the percentage of DBMS activity in a large system is relatively low or for a centrally located database with remote processors.

2.4 Dedicated and Distributed Network Configurations

The network configurations consist of several host and several back-end machines. The back-end processors may be dedicated (Figure 9) or distributed (Figure 10) as in the multiple back-end configuration. The network DBMS configuration allows for both remote processors and databases as well as centrally located facili-

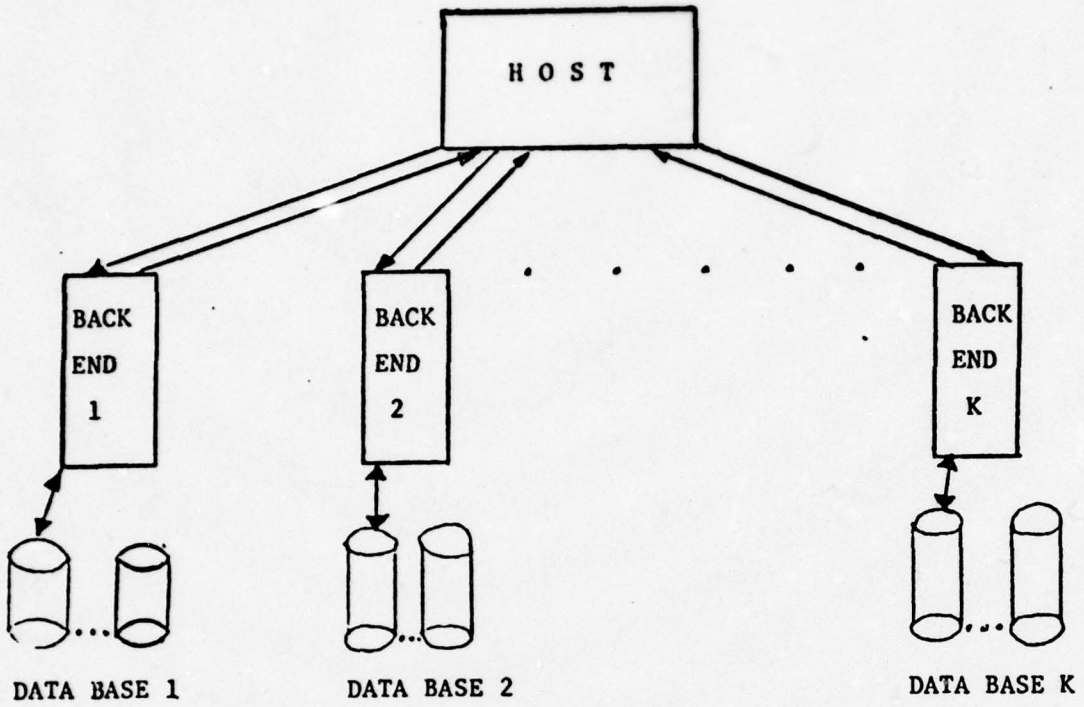


FIGURE 6 MULTIPLE DEDICATED BACK-END CONFIGURATION

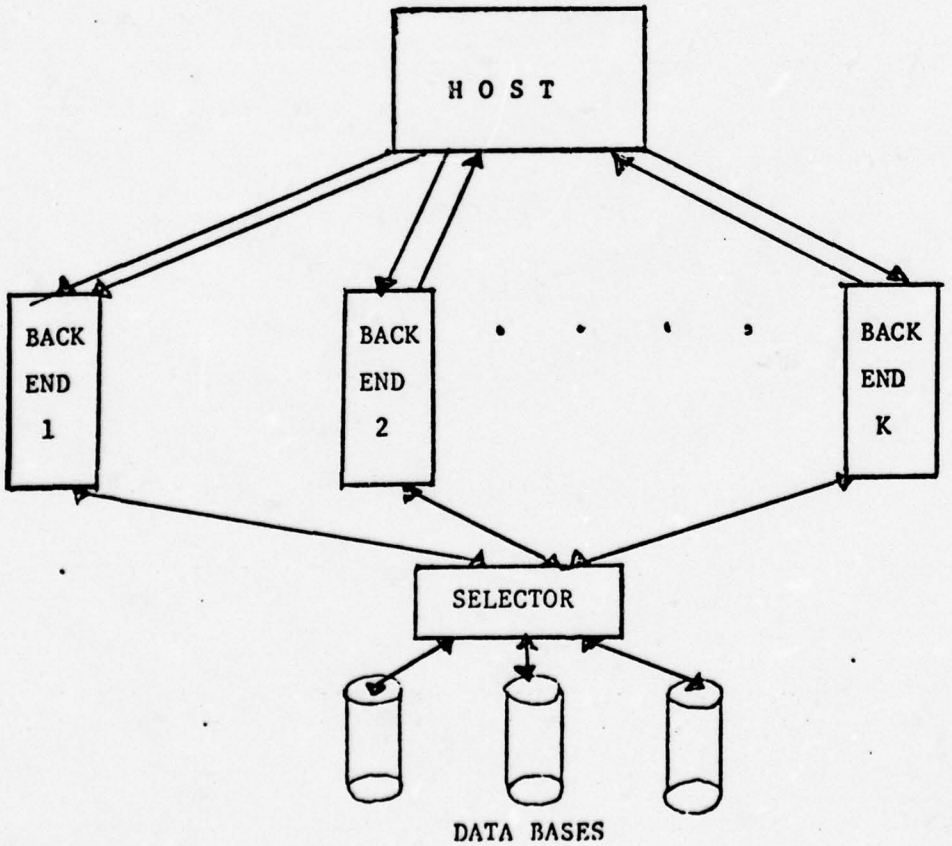


FIGURE 7 DISTRIBUTED BACK-END CONFIGURATION

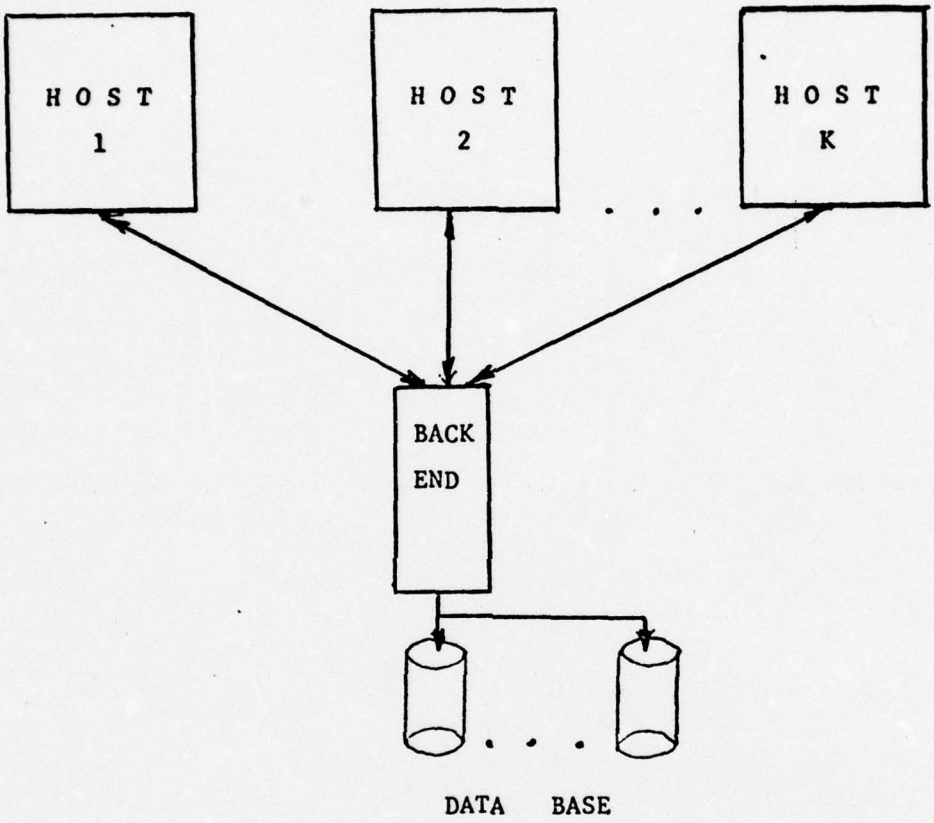


FIGURE 8 MULTIPLE HOST CONFIGURATION

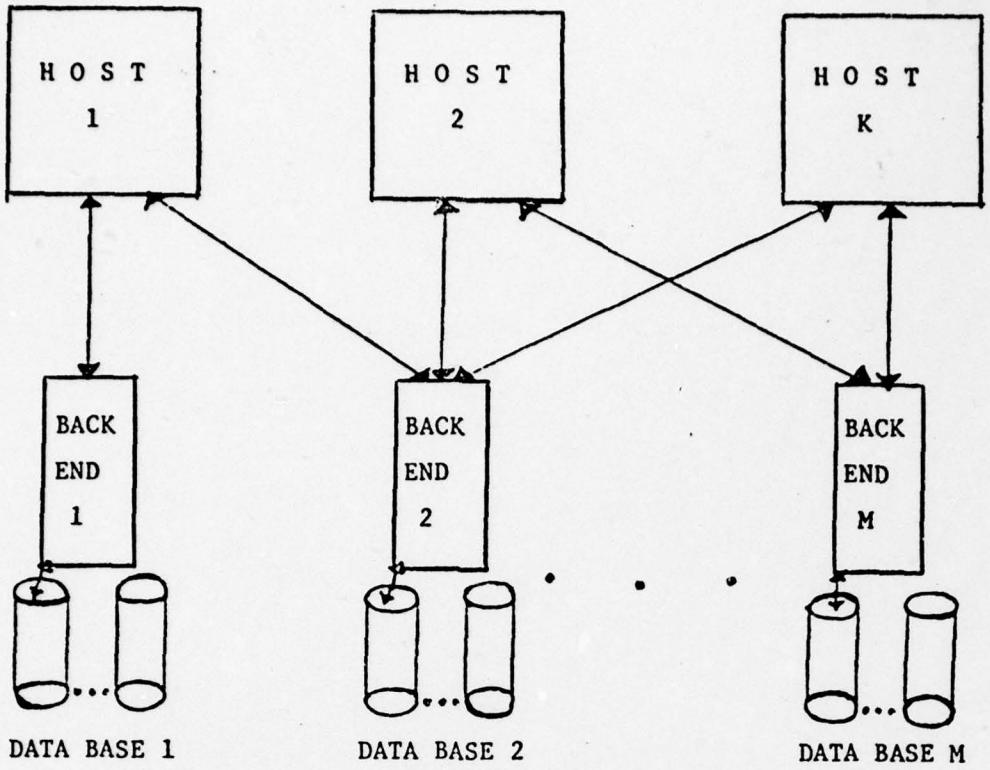


FIGURE 9 DEDICATED NETWORK CONFIGURATION

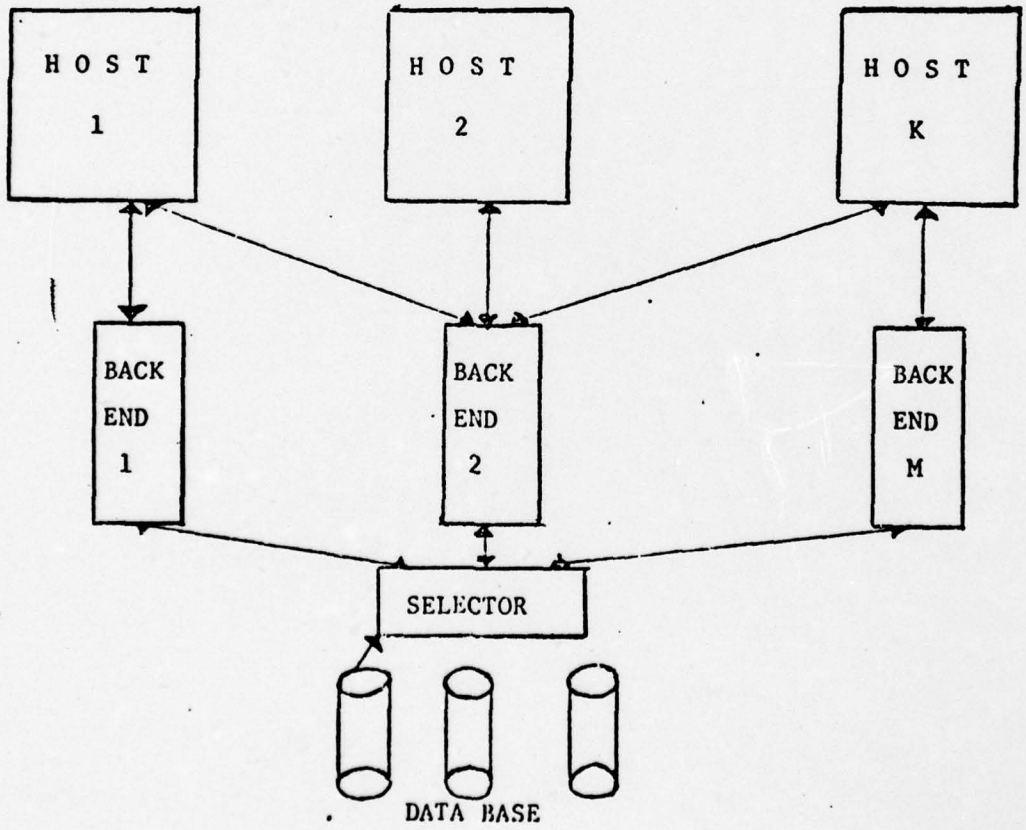


FIGURE 10 DISTRIBUTED NETWORK CONFIGURATION

ties. The selector shown in Figures 7 and 10 is a microprocessor which controls record lockout. No software is required in the selector.

In all previous discussions of back-end DBMS, it has been assumed that the back-end machine will be dedicated to database operations. In some cases, this restriction may inhibit full utilization of system resources. Provided that the back-end machine has a real time executive system, there should be no difficulty in allowing the back-end to perform tasks other than DML operations. In a generalized situation, a processor could be performing operations on a database while at the same time executing an application program which requests database information of another processor in the system. Thus, a processor may act as both host and back-end. The only restrictions as to the function of the processor is its physical connections to secondary storage. Note, however, that in this arrangement there is no true back-end processor and some of the advantages stated in Section 1.1 are no longer true.

2.5 Network Configurations with Bi-Functional Nodes

Figure 11 depicts a DBMS network configuration with host, back-end and bifunctional nodes. A typical application of such a network might be a company-wide system. In this environment, the home installation could consist of a host and back-end with the central database. Each branch installation could have a bifunctional computer capable of supporting a local database and of accessing the central database. The host machine at the home installation also would have the ability to access branch databases through the

appropriate back-end machines.

Note again, however, that while this arrangement offers some useful advantages in certain environments, since there is no true back-end processor some of the advantages stated in Section 1.1 are no longer true.

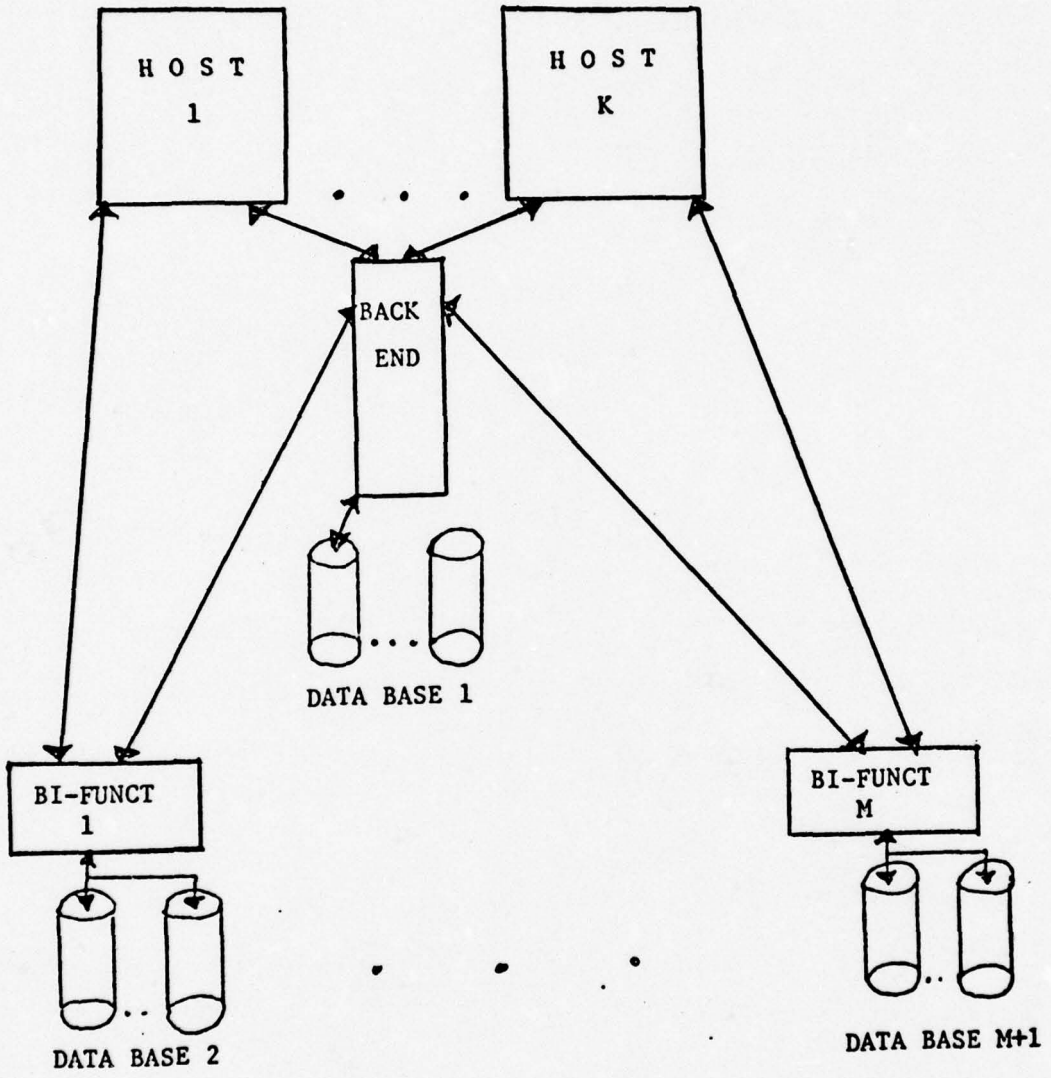


FIGURE 11 NETWORK CONFIGURATION WITH BI-FUNCTIONAL MACHINES

3.0 DATABASE TECHNOLOGY

To implement any of the hardware configurations shown in Figures 5 through 10 a large collection of software is needed in the form of DBMS programs, message systems, and operating system subroutines. To determine the requirements of the message systems and operating system subroutines, certain specifications of generalized database management systems must be presented.

3.1 The CODASYL Database Specifications

The Conference on Data Systems Languages (CODASYL) is an organization composed primarily of data processing specialists in the user community who (among other things) develop specifications for data processing languages. CODASYL is perhaps best known as the founder and developer of the COBOL language. Since 1966, task groups and committees within CODASYL have been developing specifications for database management systems. The specifications are gaining wide acceptance in the data processing community, particularly the government sector.

The CODASYL database specifications contain a data definition language (DDL) for describing a database to be accessed by a program, and a data manipulation language (DML) which prescribes the manner in which data is to be transferred between the program and the database. The DDL can be divided into two components: schema DDL and subschema DDL. The schema DDL is employed to describe the entire database in terms of the characteristics of, and the relationships between the data items. The subschema DDL provides for subdescrip-

tions of the database, or the logical view of the database that an application program has. The subschema DDL generates an object subschema which contains a logical representation of the data structure, record placement control, record characteristics, currency status, database operation statistics, and constraints on DML operations. In general, the object subschema controls the operations to and access from the database for each program which invokes it. CODASYL approved the specifications for schema DDL and subschema DDL in 1973.

The DML is used to augment the host high-level language of the database management system. The DML provides the capability of performing complex data manipulations in a single high-level statement. The host language may be any general purpose high-level language. CODASYL approved DML statements for use with the COBOL host language in March 1975, and is currently developing specifications for DML statements for other host languages.

In addition to facilities for defining and operating upon data, an effective database management package must contain certain other features. Among the most important of these features are the device-media control for specifying the mapping of the database to physical devices, database utility routines, recovery routines, and privacy and protection facilities.

3.2 The IDMS Implementation

The database management system that Cullinane Corporation would use in implementing any back-end database management system is its Integrated Database Management System(IDMS). IDMS is a subset of the current CODASYL DDL and DML language specifications.

IDMS is designed to provide database facilities for ANS COBOL and PL/I programs at the DML level, and any other host languages supporting a CALL statement or its equivalent at the CALL level. It provides DML statements which are used to store, retrieve, and otherwise manipulate data stored on random access devices. DML statements may be included anywhere within the procedure coding of COBOL or PL/I application programs. Before compilation, a preprocessor validates and converts all DML statements into CALL statements. Working storage is used to establish database record I/O areas and communications between the user and IDMS. A direct access method is used as the file management for physical I/O between IDMS and the direct access storage device.

IDMS provides separate language facilities for the description of data and the manipulation of data. This separation removes the data description function from the scope of the application programs. It also allows the integration of all data and data relations into a database which is common to all application programs that use it. In addition to the schema DDL and subschema DDL, IDMS provides a Device-Media Control Language (DMCL) to map the schema description to physical devices and define buffer pools.

IDMS supports a network or a hierarchical type of data structure. This facility permits user definitions of structures most suitable to the applications which operate on the data.

The following explains the distribution of IDMS software in a single CPU (host CPU) configuration and the communication among the individual modules (the numbers in parentheses refer to Figure 12):

(A) A DML statement appears in the object program as a CALL to the interface routine(1). The CALL identifies the type of database service desired and any additional information such as record-name, set-name, and area-name required to interpret the CALL.

(B) The interface routine analyzes the CALL using information stored in the object subschema (2). If the CALL requires the use of the DBMS, the user-supplied information is augmented by information from the object subschema before control is passed to the DBMS (3).

(C) The DBMS performs the requested database service using information supplied by the interface routine. The operation which is performed depends upon the type of DML statement executed. In the event of a CALL to locate a record (FIND or OBTAIN statement), the DBMS will look in the system page buffers (4, 5, 6) to see if the requested record occurrence is present. If the record is not in the system page buffers, a request will be made to the operating system (4) to input a database page from the direct access storage to the system page buffers. No input is initialized if the record is already present in the system page buffers.

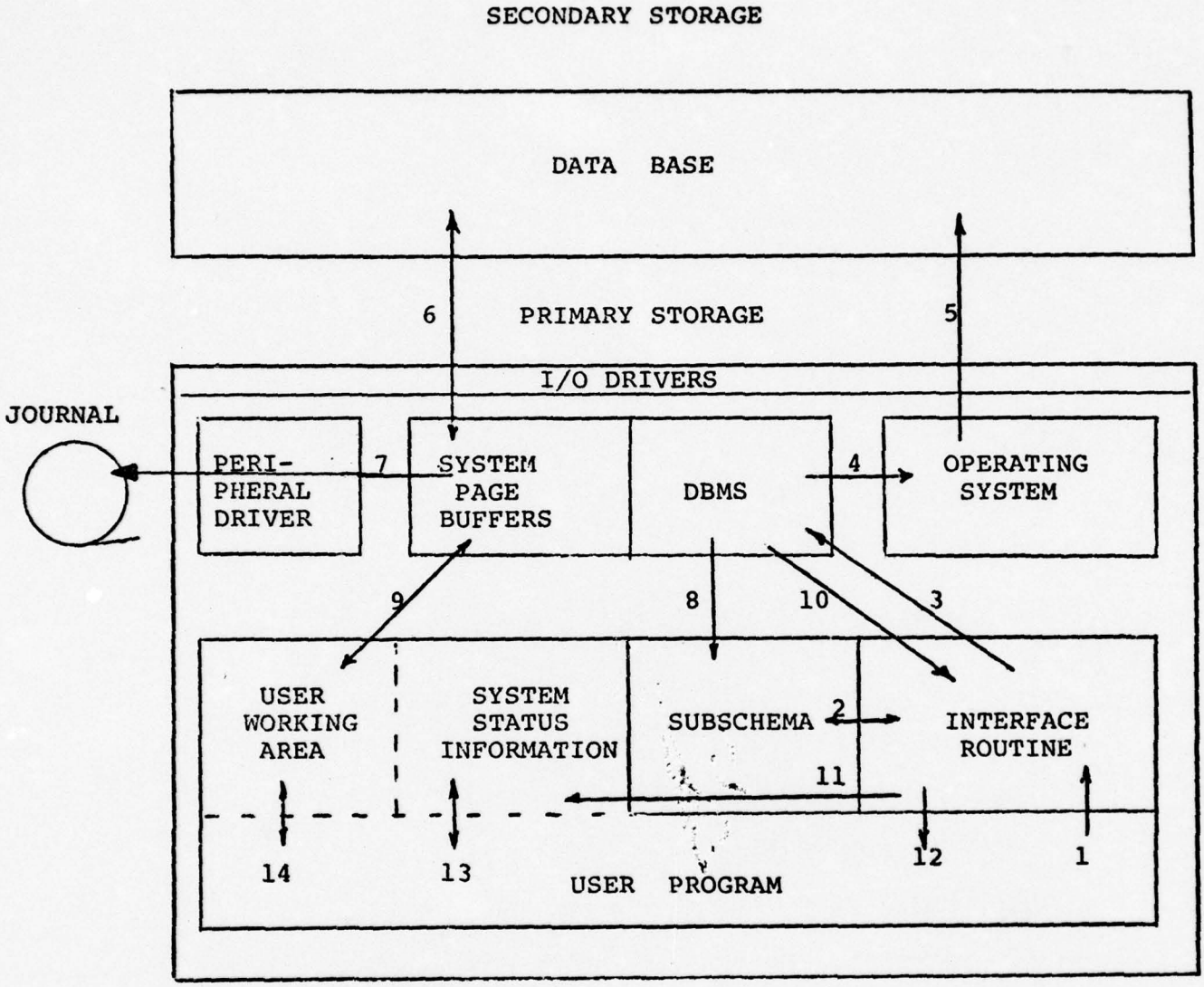


FIGURE 12 EXECUTION OF AN OBJECT COBOL/DML PROGRAM

(D) Any changes to the database are recorded on the database journal file (7).

(E) The DBMS will perform the requested database service using additional information contained in the subschema.

(F) Whenever a specified record occurrence is located by the DBMS, the database-key and other system information related to the record is moved from the system page buffer to locations within the object subschema (8). This information represents the currency status of the area, sets, and record-type of the record occurrence which has been located.

(G) If the CALL to the DBMS specified movement of the contents of a record to the user working area (GET or OBTAIN statement), data will be moved from a system page buffer to a specified record area in the user working area (9). Only the data portion of the requested record is delivered to the user; system-controlled structure data is retained by the DBMS to ensure database integrity. Data movement from the user working area to the system page buffers will occur in response to a STORE or MODIFY statement.

(H) The DBMS returns to the interface routine with an indication of the success or failure of the database service which was requested in Step C (10).

(I) The interface routine moves status information regarding the outcome of the DML statement executed in Step A to locations within the user working area of the program (11).

(J) Control is returned to the user's program at the statement following the DML statement just executed (12).

(K) The user must determine the status of the previous DML statement by examining the contents of system status information (13): For example, if a FIND statement was executed, the contents of system status information would indicate whether the specified record occurrence were located or not. If the system status information indicates the the service requested was completed successfully, the user would then access the user working area as needed (14).

One can see how in a standard multiprogramming environment the execution of a single DML command (the store of a record participating in many set relationships) could lead to the DBMS gaining and releasing control of the host CPU many times consuming considerable CPU time and requesting many I/O operations. To re-

duce this overhead, some of this software may be transferred from the host to the back-end machine.

4.0 DISTRIBUTION OF SOFTWARE IN A HOST/BACK-END CONFIGURATION

The software necessary for a host/back-end configuration can be broken into three parts. The first part consists of the software used to define and operate upon the data (database software). The second portion of the software is the operating system routines and executive routines (system and control software) in the host and back-end machines. The third part can be classified as control software and includes multi-tasking routines and the inter-CPU message systems (inter-computer communication system). The first two parts are discussed in this chapter and the third part in the following chapter.

4.1 Database Software

To implement a back-end configuration, both preprocessing and execution software modules are necessary. The preprocessing modules include the DDL schema processor, the DDL subschema processor, Device Media Control Processor for mapping the databases to physical devices, the DML preprocessors for high-level languages such as COBOL and PL/I, and all utility routines. The major execution modules are the database interface routines (DBINT), the database manager (DBMS), the database I/O routine (DBIO), and the central access and multitasking program (CAMP). The Device Media Control processor and Subschema Processors build control blocks for use by the DBMS at run-time (DMCL and SUBSCHEMA, respectively).

In keeping with the philosophy behind the back-end concept, modules and tables used in managing and accessing the database are

transferred to the back-end. Figures 13 and 14 depict the distribution of software between the host and back-end computers. This distribution is intended to minimize the number of requests the host CPU makes of the back-end machine and to keep the amount of information actually transferred between machines at a minimum. Intermachine communication is accomplished by employing the inter-computer message system to transmit information between the interface routine (DBINT) and the database manager (DBMS).

To optimize overall system performance, the DBMS, the recovery portions of CAMP, and the recordkeeping operations must be performed in the back-end. Since the database operations will be performed by the back-end machine, it is necessary to have the subschema available for validation of the DML requests. Positioning the subschema in the back-end machine will also substantially reduce the amount of traffic on the inter-computer message system.

A substantial portion of the database software is removed from the host machine. Since the subschema has been moved to the back-end, much of the validation formerly performed by the interface routine can be performed on the back-end machine by the DBMS. The interface routine now needs only to transmit information between the application program and the message system. The functions of CAMP on the host machine are to check for abnormal termination in the application program and to insure that the area associated with each DBMS task is open. The former task control functions of CAMP are handled by the message system and the back-end operating system in a back-end environment.

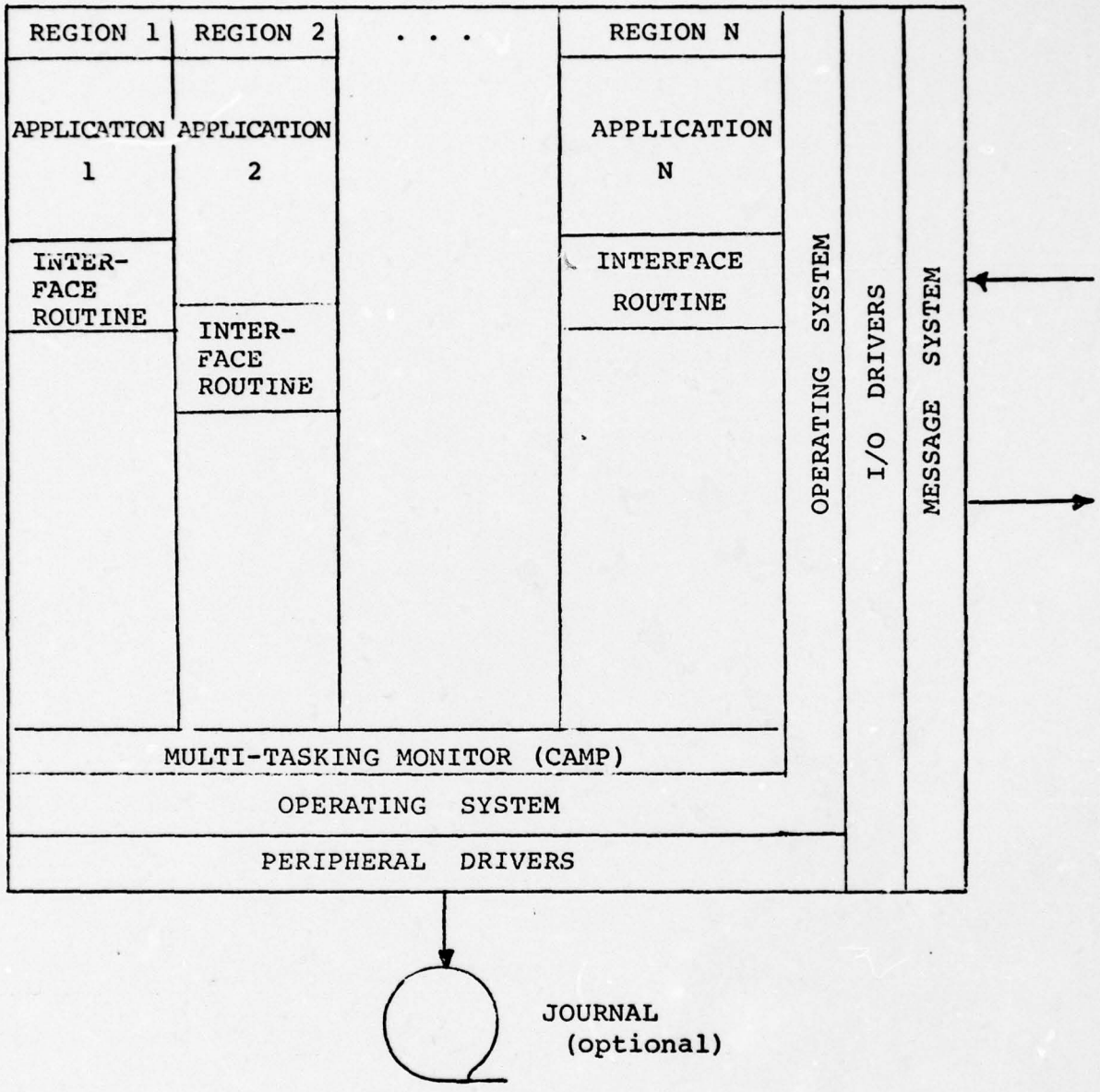


FIGURE 13 DISTRIBUTION OF SOFTWARE IN THE HOST COMPUTER

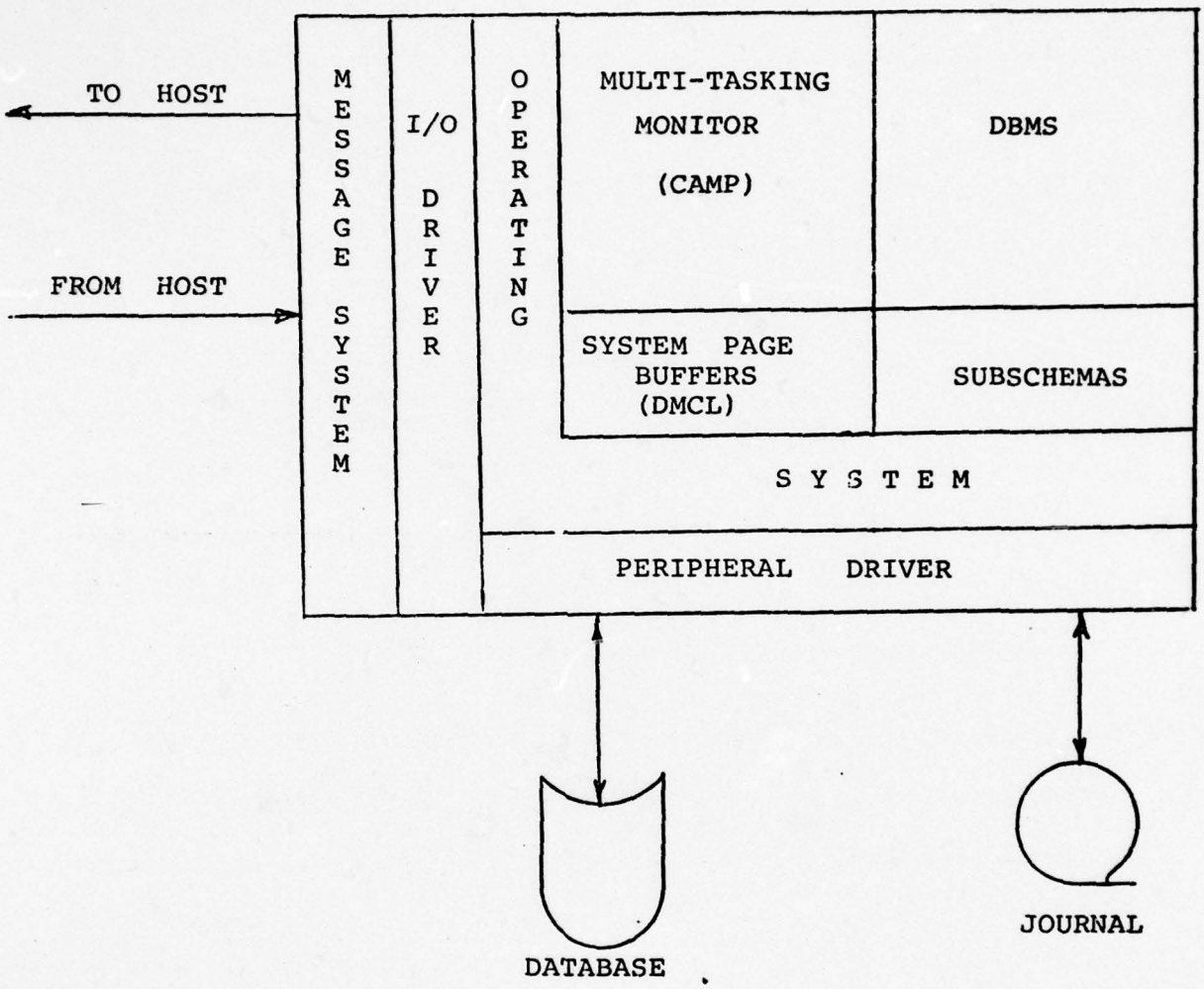


FIGURE 14 DISTRIBUTION OF SOFTWARE IN THE BACK-END COMPUTER

As a result of this distribution of software, primary memory requirements are reduced in the host machine for two reasons. By moving the database functions and buffer areas to the back-end machine, primary memory is released in the host. Using IDMS as an example, approximately 40K will be saved by transporting CAMP, DBMS, and the DMCL tables to the back-end machine. In addition, the size of the interface routine will be substantially reduced in the host, resulting in another reduction of approximately 5K. Since each application program has its own copy of the interface routine, this saving will be multiplied by the number of application programs. The subschema of each application is also transported to the back-end machine, saving about 3K. Thus, the overall reduction in primary memory requirements of IDMS in a back-end environment is 40K+(Nx8K), where N is the number of application programs in each host machine.

The above discussion is concerned with run-time DBMS software. The placement of the Schema Compiler, DMLC Compiler, Subschema Compiler, Reports Program, DML Processors and utility programs must also be considered.

The Schema, DMCL, Subschema Compilers and the Reports Program comprise the DDL Subsystem of IDMS and must all execute entirely on the back-end. The data dictionary, control blocks and status reports generated by these programs are the responsibility of the database administrator and control and document the security, privacy and integrity of the database. Since these compilers in the IDMS implementation are quite large (160K bytes for the Schema Compiler), addressability, memory management and task management

will be important factors in selecting a mini-computer for the back-end.

Application programs are compiled on the host and, therefore the DML Processors for COBOL and PL/I must execute on the host. However, the Processors require access to the data dictionary which resides on the back-end. Since the Processors are written as standard application programs and use IDMS for all database activity, they would make use of the same software modules (DBMS, message system, etc.) already described.

Most utility programs in support of the database must execute on the back-end. The programs include database dumps and restores, journal rollforward and rollback, database initialize, restructuring and displays. These utilities and their use are, like the DDL subsystem, the responsibility of the database administrator and as such are properly restricted to back-end use. A utility such as the high speed data load utility which is user-oriented would continue to run on the host.

4.2 System and Control Software

The function of the operating system in a back-end machine is to act as the common interface and to synchronize between the various tasks. The operating system accepts requests from tasks performing database functions and presents these requests to the I/O drivers to control movement of data among the host, the database, and the peripherals. The I/O drivers then merely format the information and transfer the data to or from the external devices.

A minimum list of operating system characteristics required for proper back-end performance are listed below:

(A) Multiprogramming must be supported. This permits overlap of back-end CPU, host CPU, and optimized I/O accesses, thus utilizing each resource to its maximum.

(B) Several levels of task priority must be available from which the scheduler function of the operating system can select tasks for execution. This is necessary for the back-end to provide optimal scheduling of database accessing and maximum overlap of resource usage. Since I/O is the critical performance factor, tasks issuing numerous I/O requests must be given high priority.

(C) There must be communication between tasks for status checking and error handling.

(D) Interrupts must be automatically serviced.

(E) Resource allocation must be supported. The operating system must be able to re-allocate memory and devices based on an equitable policy towards DBMS and dependent upon requests generated by such tasks.

The control software includes executive routines and I/O drivers to communicate with the other machines in the system. These routines and drivers can be standardized to handle communication with any machine in the system. The actual transmission of information between processors occurs under the aegis of the inter-CPU message system.

Several mini-computer operating systems currently provide all of the above minimum features. All operating systems are to some degree modular so that routines that are not needed can be excluded from the resident copy. Some unneeded routines may need to be present because of lack of good modularity in the operating system. A typical memory requirement for a standard mini-computer operating system is 32K bytes for 16 bit mini-computers and 60K bytes for 32 bit mini-computers. However, it is quite possible to generate an operating system for 32 bit mini's which provides all of the support necessary for a back-end DBMS and requires less than 15K bytes.

While stripped-down special purpose operating systems might save some main memory, the effort required to develop such an operating system would be considerable. Therefore, Cullinane Corporation would expect to utilize vendor-supplied system software in an actual implementation.

5.0 THE INTER-COMPUTER COMMUNICATION SYSTEM (ICCS)

The ICCS prototype has been developed apart from this study to provide the requisite communication between the host computer and the back-end computer. Certain details pertinent to this study will be discussed here. The ICCS consists of two subsystems, as shown in Figure 15, which coordinate the exchange of both control information and data. The Multicomputer Communications System (MCCS) executes on the host machine and the Inter-Task Communication System (ITCS) executes on the back-end machine.

MCCS and ITCS perform identical functions, i.e., control the exchange of messages between machines. But they were constructed under quite diverse constraints. MCCS is a single task implementation of the message control system and was targeted to execute under a single task operating system such as IBM's CMS. ITCS is a multi-tasking version and is constructed to run under a multi-tasking system with an efficient inter-task communications system such as those provided in real-time operating systems on mini-computers. The MCCS would need to be upgraded in a prototype implementation to support multi-tasking.

ICCS currently exists as a prototype system which controls the exchange of information between tasks on a back-end Data General NOVA and on a host IBM 370. These machines are connected via a 1200 baud asynchronous line. The teleprocessing control software in the CMS machine which executes MCCS views the NOVA as if it were a teletype; and the ITCS views the 370 as a paper tape reader/punch. That is, neither end of the communication line requires a sophisti-

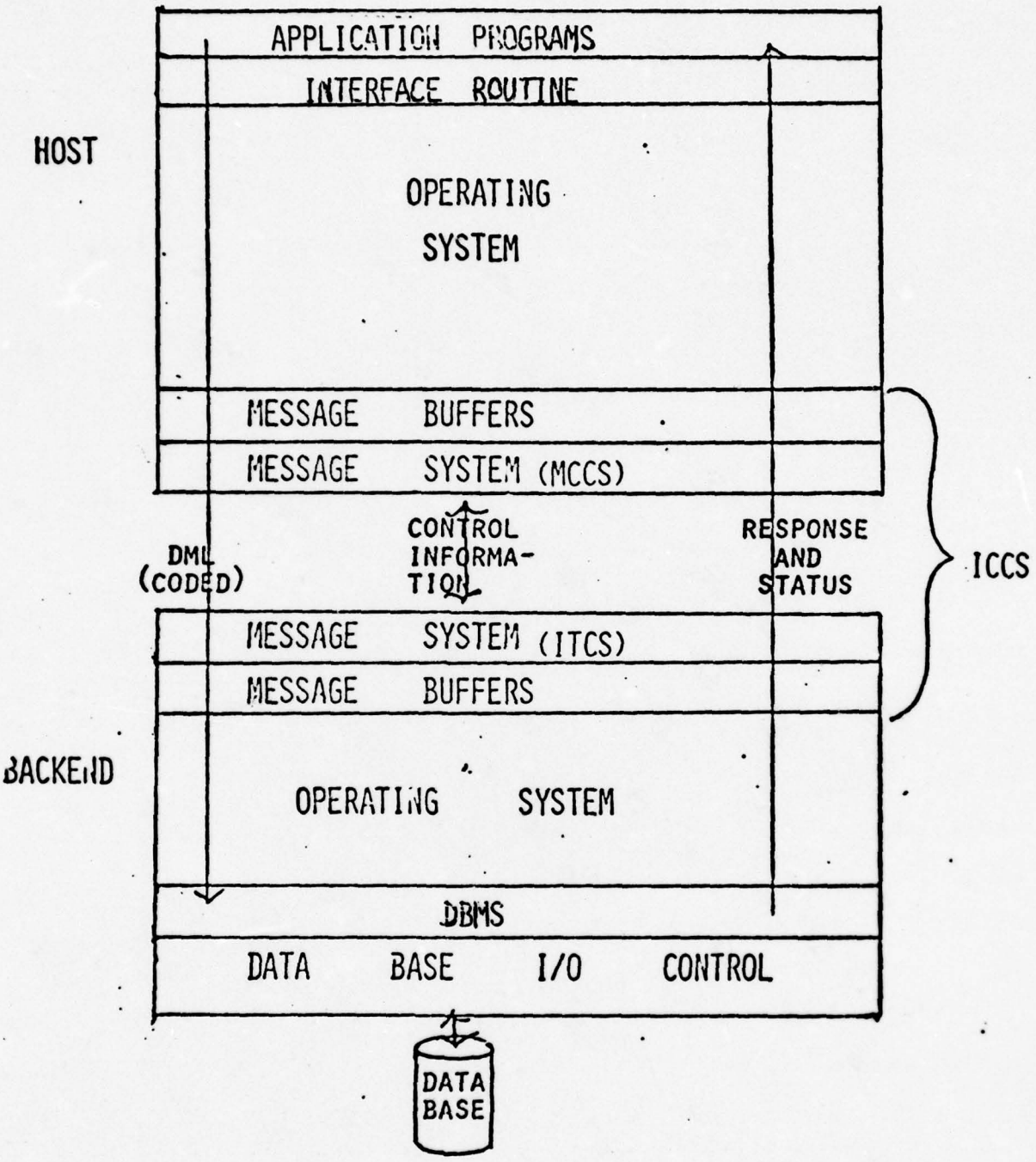


FIGURE 15 HOST/BACK-END RELATIONSHIP IN A DATABASE ENVIRONMENT

cated hardware interface to support ICCS (but as we will see later, it will provide additional efficiency).

The operation of the host/back-end system proceeds as displayed in Figure 15. The application program issues DML statements. These DML requests are converted to a series of messages to be exchanged between the interface routine and the DBMS tasks. Each message is sent to a unique DBMS task in the back-end via MCCS and ITCS. Each is routed to the appropriate task (indicated by a unique name) via the chosen hardware link (telecommunication or channel-to-channel adapter). Upon receipt of a message at ITCS, it is queued until requested by the DBMS task. Upon completion of the DML function, messages are transmitted back to the application program interface. These messages compose the response to the task in the host that issued the DML statement. These messages are again transmitted via the MCCS/ITCS communication link.

The current level of sophistication of the prototype implementation reflects only the level of effort achievable in a few months of time. Further work is proceeding to support all of the hardware configurations discussed in Chapter 2.0.

In the remainder of this section we describe the function and structure of ICCS as it currently exists, and its relationship to current software and hardware communications devices. First, a general discussion of the software is presented; and then a general description is presented of the hardware communications devices which are necessary to support such software.

5.1 ICCS Software Structure

The functional characteristics of the ICCS are as follows:

(A) It provides synchronization between tasks as well as processors.

(B) It provides a message exchange system between tasks through which can pass both data and control information.

(C) It handles buffer management in both the host and back-end processor.

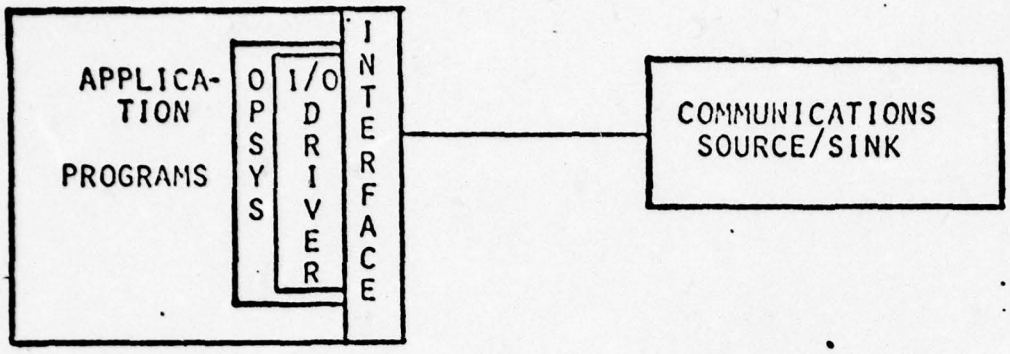
(D) It isolates the IDMS interface and the DBMS tasks from any knowledge of the physical location of the others, i.e., whether they exist in the same or different machines and whether the host and back-end are connected locally or remotely.

(E) It provides a well-defined interface (the CALL statement) to both the IDMS interface and the DBMS task.

(F) It is modular and hierarchical in nature so as to permit the straightforward modification necessary to adapt ICCS to most operating and teleprocessing system environments.

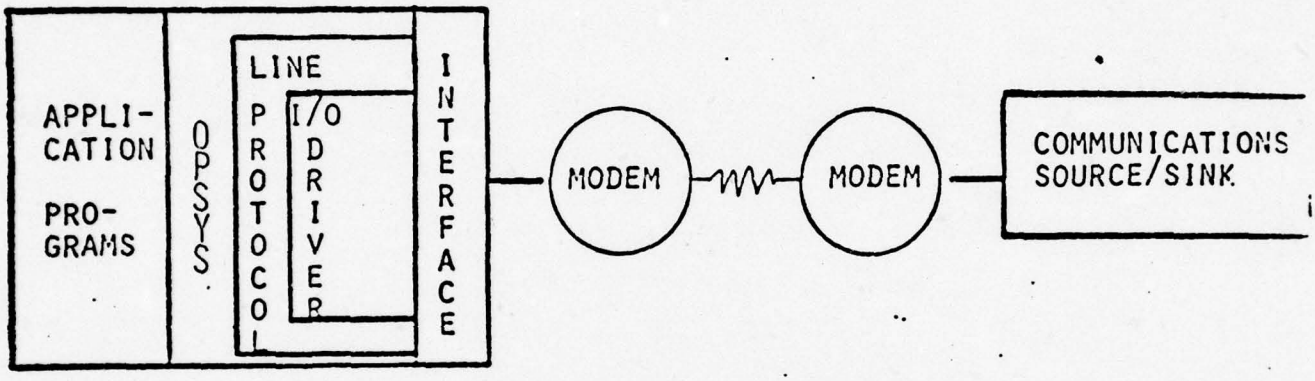
The hierarchical structure of ICCS will be discussed in terms of its integration into the operating system, I/O control, and teleprocessing systems on conventional computers. Figures 16 A, B, and C display the different levels of software necessary for application programs to do I/O. Figures 17 A, B, and C display examples of currently available systems. For local devices, the I/O driver isolates the operating system for interrupt handling. In order to handle remote devices, the line protocol handles error detection and retransmission, code transparency, and line control. (See next section for clarification of terms.) In all cases, the application program activates the operating system by a supervisor service call. But at this level the application programmer still must do his own synchronization, buffer management, and routing (addressing to machine as well as task). The message system ICCS was created to relieve the application programmer of such complexity.

The current structure of ICCS is presented in Figure 18. The IDMS interface gains control from the application program via the CALL. IDMS in turn invokes ICCS with a sequence of fixed block messages which compose the data and/or request to be made of (i.e., sent to) the DBMS tasks in the back-end machine. The CALL specifies only a task name as the address of DBMS, and has no concern as to its location. MCCS either sends the sequence of buffers to ITCS (if it has space available in the back-end) or queues the messages in its own buffer space until space becomes available. If IDMS wishes to wait for a result, it will issue another CALL to MCCS to receive data from a task. This is noted by MCCS and when ITCS sends a response (also using the CALL interface between ITCS and

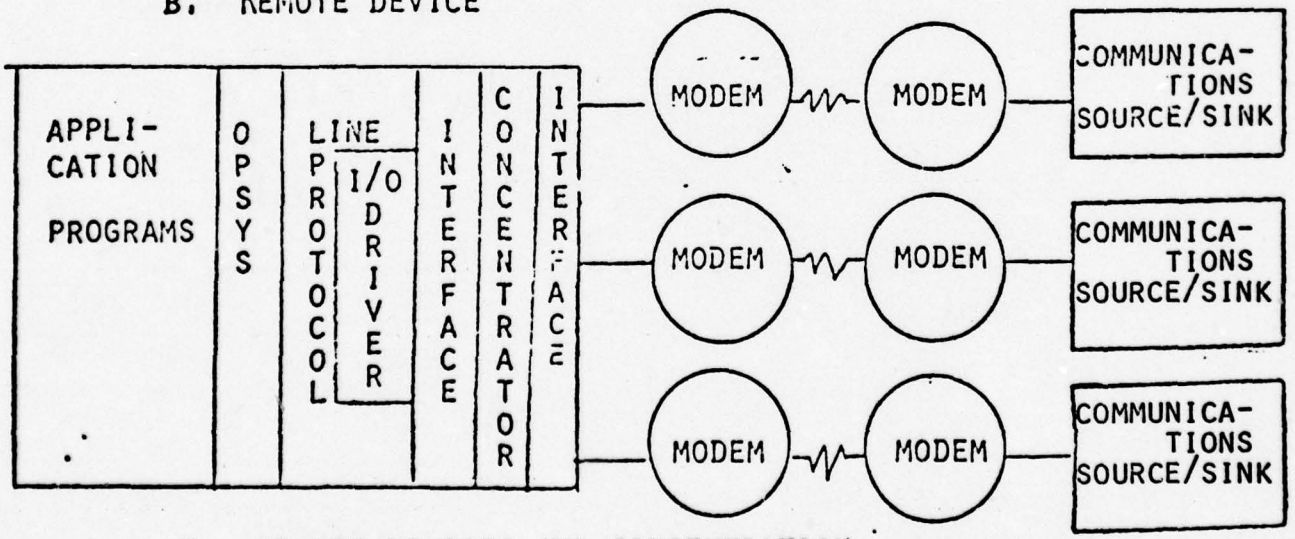


COMPUTER

A. LOCAL DEVICE

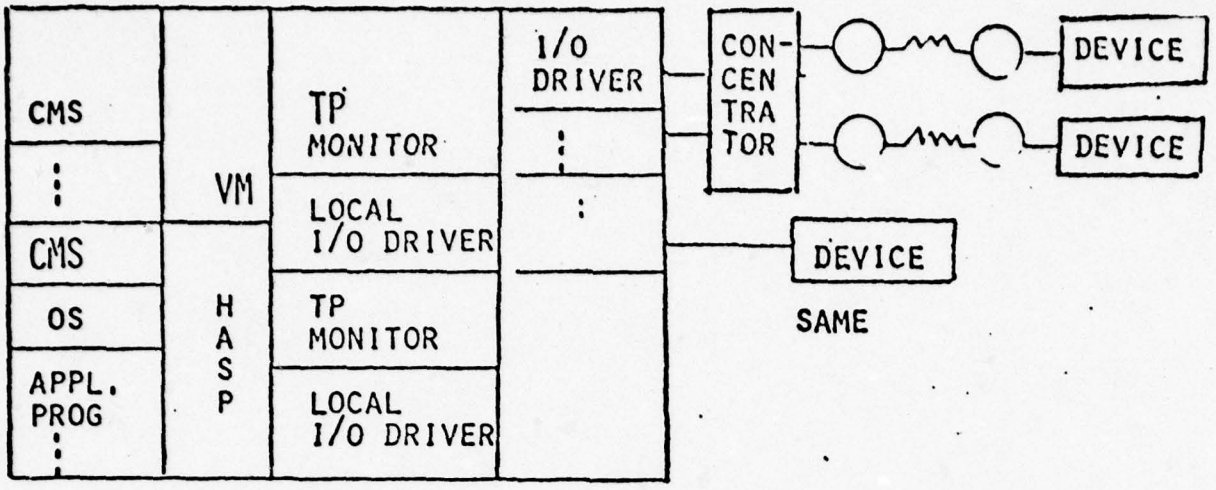


B. REMOTE DEVICE

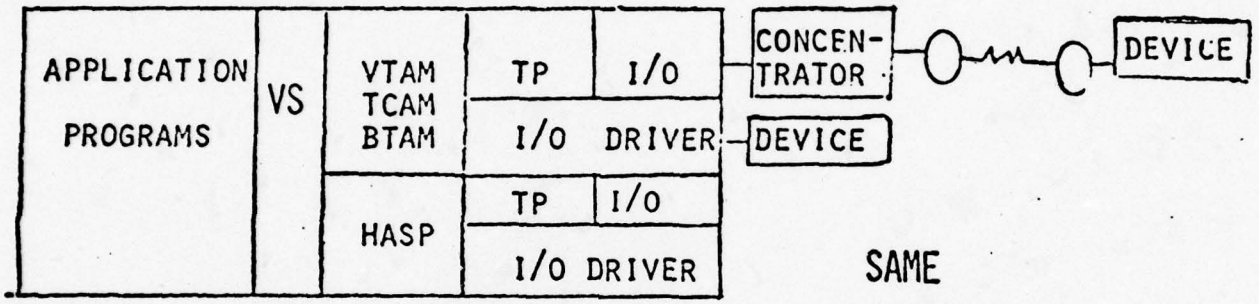


C. REMOTE DEVICES AND CONCENTRATION

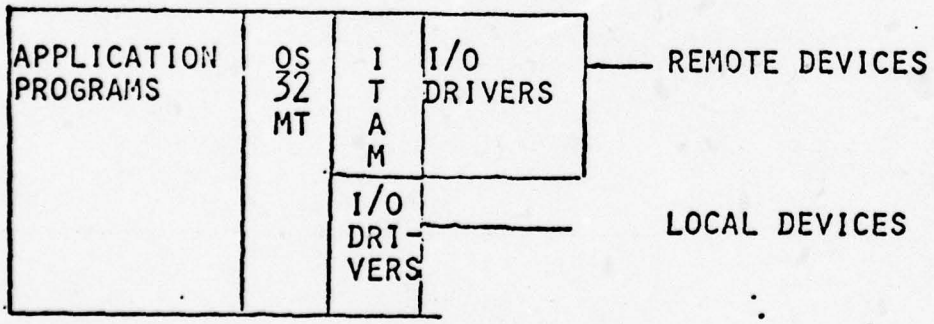
FIGURE 16. I/O SOFTWARE STRUCTURE



A. IBM VM/370 WITH CMS MACHINES AND OS ACCESSING BOTH REMOTE AND LOCAL DEVICES.



B. IBM VS WITH BOTH REMOTE AND LOCAL DEVICES



C. INTERDATA (7/32 - 8/32) OS/32-MT

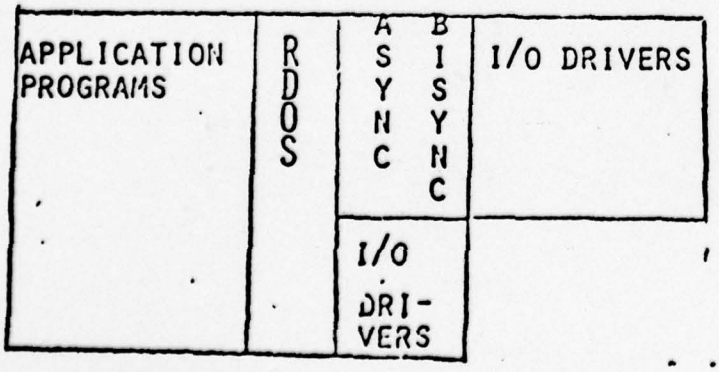
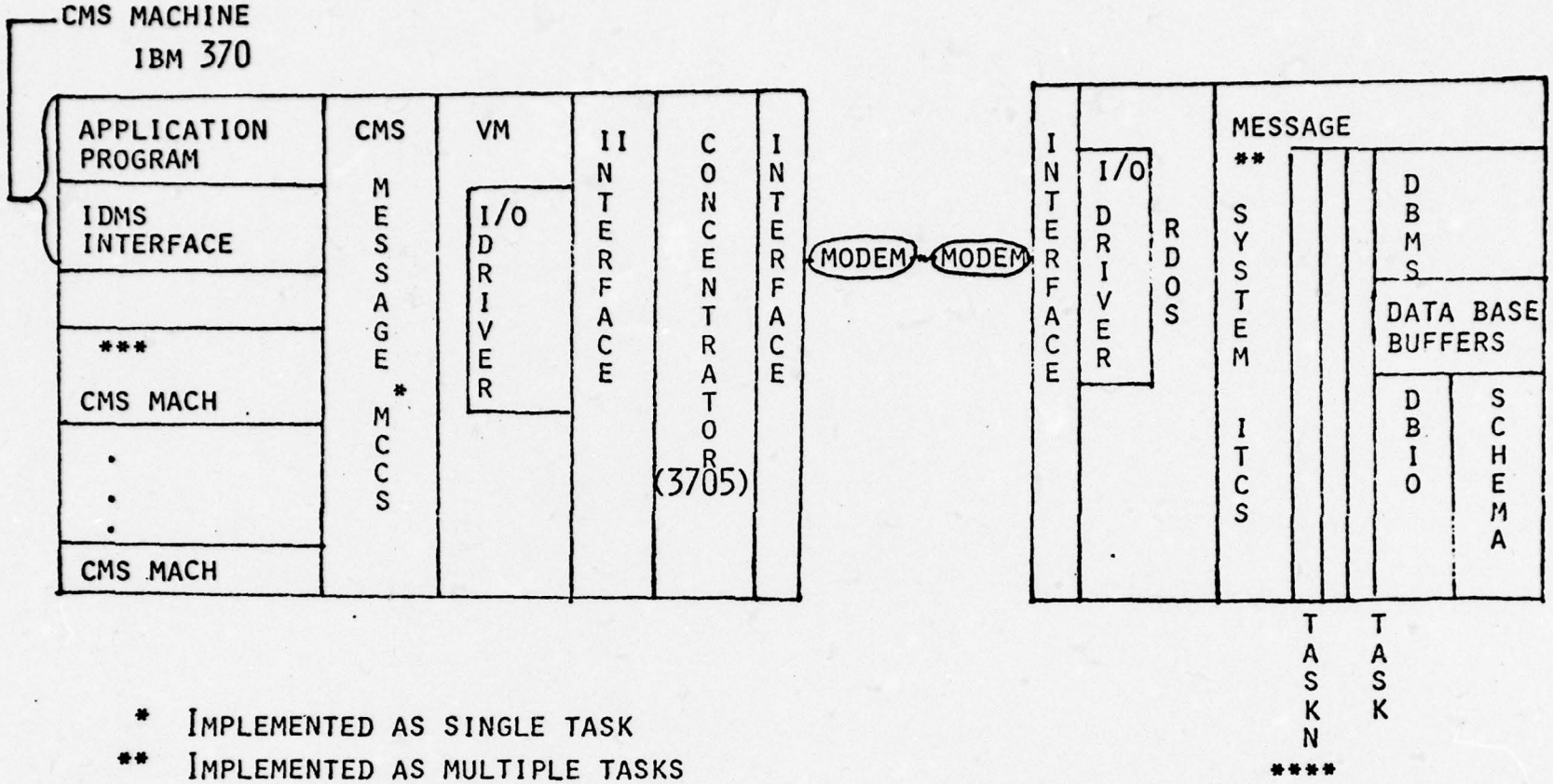


FIGURE 17 EXAMPLES OF COMMERCIAL I/O STRUCTURES



* IMPLEMENTED AS SINGLE TASK

** IMPLEMENTED AS MULTIPLE TASKS

*** POSSIBLE HOST VERSION OF DBMS.

**** POSSIBLE MULTI-THREAD DBMS TASKS

FIGURE 18. STRUCTURE OF MESSAGE SYSTEM IN HOST/BACKEND CONFIGURATION

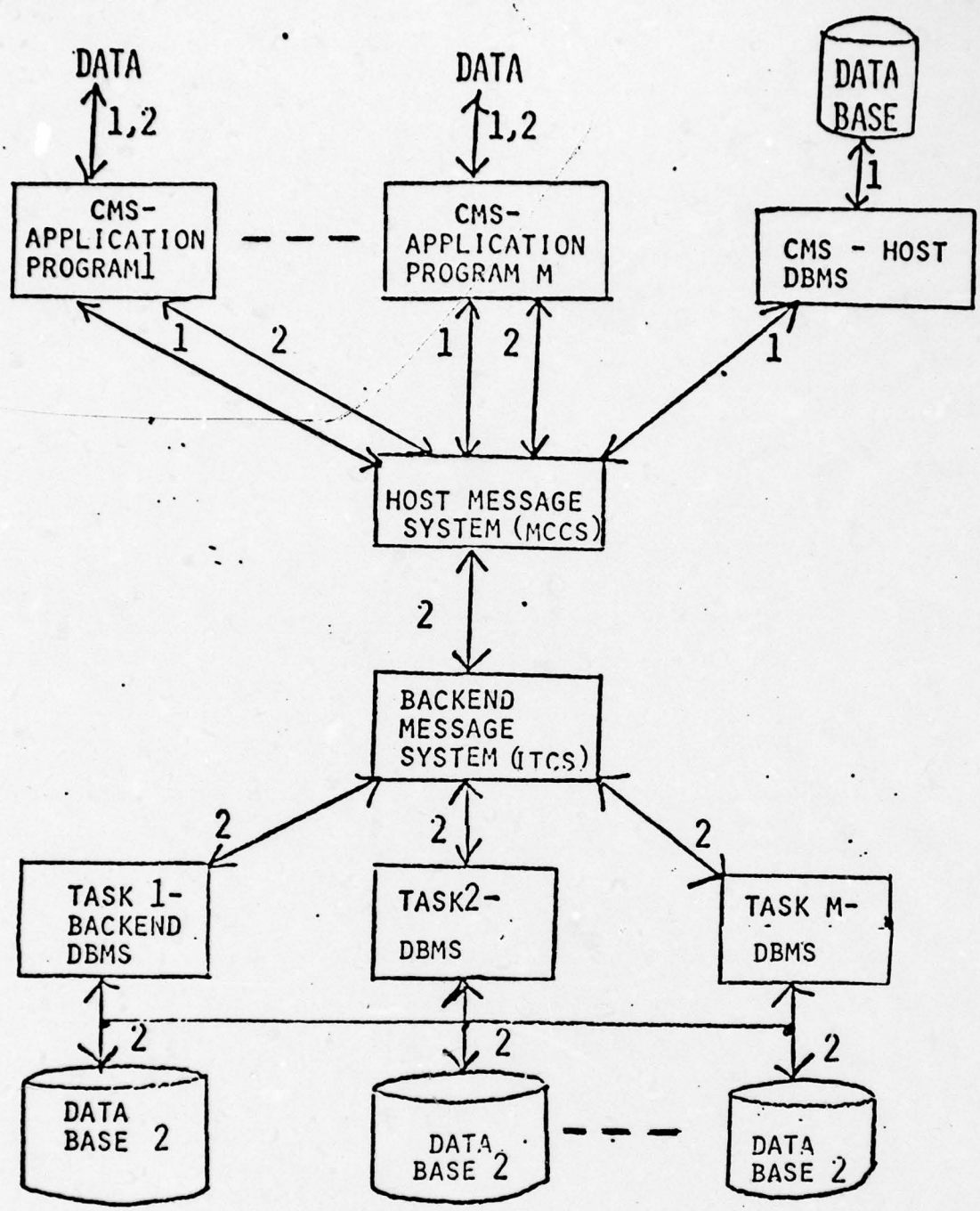


FIGURE 19 FLOW OF MESSAGES IN HOST/BACK-END DISTRIBUTED DATABASE CONFIGURATION

the DBMS task) MCCS communicates the message to the appropriate application program - again by its unique process (program, task) name. (See Figure 19 for message flow possibilities).

Since ICCS is comprehensive in its message exchange capabilities, an application to distributed databases is possible. That is, since communication is at the task level, the DBMS task may be running on the host or on the back-end. In either case, ICCS will perform the correct transmission of buffers. Note in Figure 19 that the application program can activate DBMS tasks in the host using path 1; and it can invoke a back-end DBMS task via path 2.

ICCS queues messages destined for a host in the "out" list of MCCS first. Then it merely links it to the "in" list of MCCS. In the case where a back-end DBMS task is to be invoked, the messages are sent by MCCS via the hardware connection to be queued, when buffer space is available, in ITCS. The reverse operation is achieved in the same manner. This generalized message exchange mechanism achieves the symchronized operation of the application program and the DBMS tasks that serve it.

5.2 Hardware Interface Considerations

As described in Chapter 2.0, several methods of hardware intercomputer connection are possible. The machines can be connected locally or remotely. The distance between the two nodes dictates the software support necessary, and the requisite hardware devices

of the connection. All software and hardware upon which ICCS depends are commercially available. Note that efficiency is best achieved if the machines are attached locally. The prototype implementation of ICCS can be upgraded to support any of the software and hardware structures shown in Chapter 2.0.

The requisite hardware interface can be a channel-to-channel attachment or a telecommunications support device. The direct channel devices run at speeds which range from 300KB to 4 MB per second. These compare with speeds of 1200 baud to 50K baud for telecommunications lines. Some special interface devices to large mainframes (IBM, CDC, UNIVAC) are available from mini-computer manufacturers such as Data General, Interdata, DEC, Varian, and others.

Each device utilizes a unique mode of operation and, therefore, requires unique software drivers which are again available from the manufacturer. For example, the Interdata to IBM 360/370 interface emulates a paper tape reader punch; and the current prototype NOVA to IBM 360/370 configuration emulates a magnetic tape unit. Both are sequential devices which are supported by I/O drivers. Therefore, the I/O operations available to ICCS are well-defined and directly adaptable. The utilization of such software is via the basic sequential access method available on the mini-computer back-end (ITCS) system. The connection of MCCS to such software in the host is via the channel-to-channel software drivers already in MCCS. That is, in order to permit host application programs to access host IBM DBMS tasks in CMS machines under VM/370, MCCS

implements a virtual channel-to-channel connection between CMS machines. This software is directly utilizable on real channel adapters.

The alternative inter-computer connection is via telecommunications lines. Here the efficiency of the host/back-end is degraded somewhat by the differential in interface speeds as previously specified. Additional overhead in error detection and retransmission, code transparency conversions, and control information is inherent in the line protocol. This added software overhead is obvious in Figures 16 and 17. In each case, there is an additional level of software between the I/O device and the application program for the remote connection versus the local connection.

The ICCS prototype includes its own error detection and retransmission procedures, and its own code transparency conversions. This direction was chosen to permit ICCS to establish inter-task communication over the simplest of all line connections, i.e., asynchronous lines with no associated protocol. However, this is inefficient since the software must perform all these operations at the software level above the operating system. This means that all access to external devices must go through several levels of software invocation and that such procedures execute at software speeds as opposed to hardware speeds.

The use of binary synchronous or SDLC line protocol would allow the removal of the code transparency and error detection procedures since those are included in the standard line proto-

col drivers. As a result, efficiency would increase significantly.

As can be seen from Figure 19, DML requests can be initiated in the host and receive responses from either the back-end DBMS tasks or the host DBMS tasks. However, the ICCS is general enough with its inter-task message exchange philosophy that additional application programs could be running in the back-end. It could then issue DML requests to host DBMS tasks or back-end DBMS tasks. This permits databases to be distributed across machines. This mechanism is not sufficient to allow shared access by multiple machines to a single random access device containing the data. The logic to provide this shared access and to determine and resolve database deadlock problems among application programs is provided by the multi-tasking monitor (CAMP).

6.0 INTRODUCTION TO MINI-COMPUTER ARCHITECTURE

In order to select properly a mini-computer system for a back-end processor, it is necessary to understand the architecture of a computer system.

The architecture of a mini is the key to its performance and an indication of its relative utility and power. Machine design has a direct bearing on the mini's operating characteristics, such as speed and efficiency, how instructions are carried out, and how easy or difficult it is to program the system.

The following units are the basic building blocks in a mini:

- Memory
- Arithmetic and logic unit
- Control
- Input/Output (I/O) unit

The above subunits are interconnected by several lines or wires which provide paths for data. These lines are commonly called data highways or buses.

The memory, also referred to as the main storage unit, stores actual data as well as instructions that tell the control unit what to do with the data. The arithmetic and logic unit temporarily stores data received from memory and performs calculations and logic operations on this data. The arithmetic and logic unit contains one or more registers, which, in turn, contain

the data being operated on as defined in the instruction.

The control unit, as the name implies, controls the flow of data in the system, fetches instructions from memory, and decodes the instructions in one or more instruction registers. The control unit executes instructions by enabling the proper sequence of operations performed by the arithmetic and logic unit and the input/output unit. Finally, it changes the state of the computer to that required by the next operation.

The input/output unit provides the interface and, in some systems, buffering to peripherals connected to the computer, transferring data to and receiving data from the outside world.

The control unit and the arithmetic and logic units with buses and registers are referred to as the central processing unit (CPU). The earliest, and still the most widely used, architecture is based on the CPU reading from and writing to memory over a high-speed memory bus and to all external devices over an I/O bus.

Some mini-computers, however, treat the memory as an external device and communicate with it over the same I/O bus, just as they would with an external peripheral device. Interface registers act exactly the same as main memory registers. There is, therefore, no need for special I/O instructions. The full power of the entire instruction set can thus be used in manipulating the interface registers. The single bus organization offers several advantages over the more conventional

mini-computer architecture for more general systems. It permits the user to mix various speed memories in the same system. Furthermore, it permits a considerable amount of standardization to be applied to the interface design. The interface itself is somewhat more complex due to the universality of the single bus and the volume of traffic through the single bus may degrade seriously performance. The main advantage of the single bus architecture lies in the simplified functional programming for control applications.

Most mini-computers employ a technique of staging instructions to be executed into some higher speed memory than main memory. This staging area can be a cache or lookahead stack.

A cache is usually several hundred words of high speed memory into which a page (series of consecutive main memory addresses) is loaded. The cache can contain instructions and data. When the next instruction to be executed is outside the cache, the current page of the cache is rewritten to main memory if changes have been made to it while in the cache, and the page containing the next instruction is read into the cache.

Lookahead stacks are typically employed in pairs and are a word or two each in size. Instructions only are loaded into the stacks in an interleaved fashion--while the CPU is executing the instruction from one stack, a fetch of the next instruction into the other stack is taking place.

7.0 BACK-END HARDWARE REQUIREMENTS FOR A DATABASE MANAGEMENT SYSTEM

The minimal requirements for a back-end computer supporting IDMS include features such as addressability of 64K bytes of main memory, main memory speed comparable to large-scale hosts, 12-16 general purpose registers internal to the CPU, several levels of automatic I/O interrupt and a general instruction set. Most standard 16-bit mini-computers have sufficient power to satisfy these minimal requirements. In addition, the peripherals supported by the mini-computer must include standard magnetic tape drives and movable head disc drives which accommodate auxiliary long and short term storage and a high-speed swapping device for program rolling and roll-out.

The back-end's performance can be enhanced in several ways. Such obvious improvements as faster memory, interleaved memory, and cache memory improve performance at an additonal cost. In addition, certain mini-computers provide tailored instruction sets for manipulating chains and lists. Each occurrence of these instruction sets improves the efficiency of dealing with database information structures. A primary consideration in the performance of the back-end computer will be the management of the database in main memory. Therefore, another factor in performance will be the extent of directly addressable memory without memory management overhead.

Since the back-end machine serves as the I/O handler, message switcher and DML executor, it is the central interrupt handler

in the configuration. It, therefore, must service the interrupts from the host and all the I/O devices accessed by the DBMS in an efficient manner. A performance factor in handling interrupts is the amount of system state that needs to be saved and later recovered on each event. Therefore, it is highly desirable that the back-end possess several sets of internal registers so that these registers need not be saved on every interrupt.

A related performance issue involves the I/O data rates needed between the host and the back-end and the I/O devices and the back-end. The minimum acceptable rate is considered to be 5 megabytes per second. Lower rates immediately introduce response times slower than a host only configuration. In addition to this I/O rate, the data bus organization is important. For example, if all I/O is over a single, asynchronous bus, serious throughput problems occur with a heavy I/O load.

Database management systems in the future will take advantage of microprogramming techniques to speed processing. Microprogramming can take the form of a Read Only Memory (ROM) or a Writeable Control Store (WCS). ROM is only accessable by the CPU. The instructions sequenced in this memory are "hardwired" by the manufacturers and not changeable in the field. Information (instructions) in this form can only be constructed by the manufacturer. WCS is another level of very fast access memory and the user under program control can input and output information from this level of memory, and execute instructions in it.

In order to use microprogramming, one must modularize the DBMS into a tree structure of hierarchy, then define those most heavily used modules that are the lowest part of the hierarchy. By placing these modules in the control store they can be executed faster than in main memory. The routines are executed or invoked when controlled by a user level instruction in main memory. In terms of performance, if modules are written to reside in the writeable control store, they would typically run 2 to 3 times faster than if they were in main memory. If these modules are known prior to machine manufacture and can be put in the ROM, then they will typically execute 4 to 5 times faster than if they were in main memory.

The microprogramming feature is anticipated to be of significant value in reducing the instruction access times of a DBMS and a query facility in the mini-computer.

8.0 MINI-COMPUTER EVALUATION

Several mini-computer vendors were interviewed during this study. Where interviews were impossible or refused, data was collected from available documentation. Substantial detail was considered for each machine in Appendix A.

Statistics were collected on the usage of IDMS at several locations. Factors such as CPU time, I/O time and main memory requirements were adjusted to represent the load and requirements on a back-end machine. As a result of this exercise, the table of important mini-computer features to support database management functions (Figure 20) was constructed. By comparing the data of each mini-computer considered to the requirements in Figure 20, four machines were selected for final consideration: DEC PDP 11/70, Interdata 8/32, MODCOMP IV/25, and SEL 32/55.

Figure 21 shows a comparative analysis of the important mini-computer features for these four machines and also statistics for an IBM 370/158 to provide a base of reference. The following discussion addresses each of those features in sequence.

The PDP 11 and the MODCOMP IV have 16 bit word sizes, the Interdata 8/32 and the SEL 32/55 have 32 bit word sizes. With the exception of the PDP 11/70, all of the finalists have a 32 bit internal data path. The Interdata and SEL, therefore, are the only full 32 bit machines.

While the maximum memory of each machine is large enough to hold DBMS routines, the direct addressability limit of 64K for

the PDP 11 and MODCOMP IV requires considerable software overhead of task switching in the implementation because some IDMS programs are larger than 64K. Also a query facility operating in the back-end is expected to occupy more than 100K bytes, requiring even more overhead in these two machines.

A cache memory improves throughput by prefetching the next or a series of instructions into a staging area which can be accessed faster than main memory. The one word lookahead technique of the Interdata, MODCOMP, and SEL are considered equal to the cache page technique of the PDP 11 for the IDMS implementation.

The number of registers and levels of interrupts are very important in terms of efficiency of task switching and ordering priorities. While one set of 16 registers is satisfactory, a second set of 16 is valuable. The first set of 16 could be used for the message system, the second could be dedicated to the DBMS module, thereby saving and reloading one set. Several interrupt levels are required to establish levels of priorities - for example, the message system might need a higher priority than disk I/O which in turn might need a higher priority than the journal file or console communications.

All systems provide the desirable data rates and addressing schemes. SEL has a clear advantage in the I/O data rate category with an astounding 26.7 megabyte per second rate, even though no I/O device can operate at that speed. All four systems utilize a DMA technique to attain these I/O speeds.

While the number of instructions available is not important per se, the addressing attribute of the instruction is. Interdata provides a long instruction format (6 bytes) which provides a growth potential of 16 megabytes of direct addressing. The Interdata instruction set is also very similar to the IBM instruction set and cross-assembly of many IDMS programs is possible.

The effective memory speed of each machine is in the 300 to 600 nanosecond range. Activity in most database structures is I/O bound but data structures can be designed which result in a heavy CPU load. It is important to note that these mini-computer memory speeds are competitive with large mainframes. Effective memory speed is somewhat of an estimated speed and as such is open to comment and criticism. In order to better rate this feature, the execution times of certain instructions which are frequently used in a database management system were charted in Figure 22. The total time for these instructions was considered in addition to the effective memory speed value to determine scores. It is interesting that Interdata and SEL can process 32 bits as fast as the PDP 11 can process 16 bits.

Only Interdata and SEL offer hardware I/O. Hardware I/O simply means that no software program needs to be executed to accomplish the I/O; the logic for the transfer is in the form of a microprocessor. All machines provide a real-time clock.

While all the machines have relatively powerful macro assemblers, only DEC has a COBOL release in the field. Interdata has an apparently quite good ANS COBOL compiler in beta test and is due to release it in December, 1975. A COBOL compiler is valuable

as several IDMS off-line processors are written in COBOL. Without COBOL these programs would require rewrite in the macro language. The COBOL compiler may also be an important item if any user problem programs are to be written for the back-end.

Interdata offers the best in power and flexibility in the microprogramming area with the availability of 2000 bytes of writeable control store.

The finalists all support synchronous communication, binary synchronous communication, and all state that they will eventually support SDLC. SEL and MODCOMP appear to have the best flexibility in telecommunications support because their control units are microprocessors and can be programmed to support any function.

One way of achieving interprocessor communication is through common memory. Multiple processors can be attached to the same memory bank and can communicate by placing data in memory. The SEL 32/55 can attach up to 16 processors on a common memory and the Interdata 8/32 has a prototype 14 processor system on common memory now. This is important for a future expansion of the back-end computer system. A less tightly constricted arrangement of multiprocessors is through a network arrangement in which each computer is tied to another computer by a telecommunication link, either synchronous or asynchronous. At this point in time, MODCOMP with its network operating system (MAXNAT) and DEC with DECNET are the only systems that support the distribution of tasks through the system in a network. These vendors support a network control language that allows the movement of tasks and data throughout the network

and allows communication between computers at either a local or a global arrangement. Interdata is soon to announce a similar system.

All four systems support a direct access method with a relative block addressing scheme. Each has file and block size restrictions but these restrictions can be masked from the user by the IDMS device media control language.

The prices shown were obtained from a marketing representative of each vendor. Pricing is highly competitive with various discounts available for quantity orders and to OEM suppliers. The amounts shown are averages including discounts for the purchase of five configurations.

Based on the hardware prices shown in Figure 21, the cost to acquire the hardware and software necessary to operate a single host/single back-end database management configuration can be estimated as follows:

Back-end CPU	$ 75,000
500 megabytes disk storage (based on database size)	125,000
Modems	2,000
Complete IDMS Software	60,000
TOTAL: (except for teleprocessing lines and host ports)	$262,000

Teleprocessing lines and host ports would most likely be rented. Hardware and software maintenance contracts would cost approximately $20,000 per year.

Finally, in the area of vendor interest and willingness to cooperate and participate, two vendors stand out. Both DEC and Interdata have been accomodating throughout the study and aggressive in their efforts to demonstrate their machines. Assistance in the development of a back-end database management machine by either of these two vendors is considered a real possibility.

8.1 Final Technical Rating of Back-End DBMS Candidates

Each of the four mini-computers was rated on each of the items in Figure 21. The rating was multiplied by the weight factor to determine a score. The score in each category was then totaled. The detailed ratings and scores are shown in Figures 23, 24, 25 and 26. The summary of the technical ratings follows:

Machine	Rating	Score
DEC PDP 11/70	210	2000
Interdata 8/32	258	2470
SEL 32/55	228	2130
MODCOMP IV/25	209	1920

Consistent with these scores, the Cullinane Corporation finds the use of the Interdata 8/32 with 256K bytes of main memory has superior technical characteristics for back-end database management. Performance of the system is expected to be excellent due to the large direct addressability, number of optional registers, levels of interrupts and effective memory speed. The availability of ANS COBOL in December of 1975 will allow the machine to be used for business data processing as well as serving the back-end function.

Based on our experience in converting IDMS to various computer mainframes, a prototype system with an IBM 370 host and an Interdata 8/32 back-end would require five man-years of effort spread over two calendar years. The cost of such an effort is estimated at $300,000, consisting of $200,000 salary and overhead expense, and $100,000 machine and material expense.

FEATURE	DESIRED QUANTITY OR ATTRIBUTE
Word Size	32 bits
Internal Data Path	32 bits
Maximum Memory	1 Megabyte
Directly Addressable Memory	256K bytes
Cache Memory	Yes
Number of Registers	16
Levels of Interrupts	4 levels of 16 each
Data Rate	5 Megabytes/second
Addressing Schemes	Direct/Indirect/Indexed/Stack
Number of Instructions	----
Task Switching	Yes
Effective Memory Speed	500 nanoseconds
Hardware I/O	Yes
Real Time Clock	Yes
COBOL Compiler	Yes
Macro Assembler	Yes
Coding Scheme	ASCII or EBCDIC
Microprogramming	Writeable Control Store or Read Only Memory
Synchronous Comm	Yes
Asynchronous Comm	-----
Bisynchronous Comm	Yes
SDLC	-----
Multiprocessing	-----
Direct Access Method	Relative Block Addressing
Price: CPU + 256K	$75,000 maximum
Vendor Interest	-----
Vendor Willingness to Cooperate	-----

FIGURE 20 IMPORTANT MINI-COMPUTER FEATURES

FEATURE	WEIGHT	DESIRED	IBM 370/ 158	DEC PDP 11/ 70	INTERDATA 8/32	SEL 32/ 55	MODCOMP IV 25
Word Size	5	32	32	16	32	32	16
Internal Data Path	5	32	32	16	32	32	32
Maximum Memory	10	16 meg	16 meg	2 meg	1 meg	1 meg	512K
Directly Acc. Memory	10	16 meg	16 meg	64K	1 meg	512K	64K
Cache Memory	10	NO	NO	2K	4 bytes lookahead	4 bytes lookahead	6 bytes lookahead
No. of Registers	10	16	16	16	8 sets of 16	8	16 sets of 15
Levels of Interrupts	10	4	5	4 levels of 128	4 levels of 256	128 levels	16
Data Rate (I/O)	10	5 meg	6.7 meg	5 meg	6 meg	26 meg	2 meg
Addressing Schemes	10	Dir/Ind/ Index/ Stack	Dir/Ind/ Index	All	All	All	All
Instructions	10	- -	167	400+	227	152	255
Task Switching	10	YES	YES	YES	YES	YES	YES
Effective Memory Speed	20	500	250	300 ns.	300 ns.	600 ns.	500 ns.
Hardware I/O	10	YES	YES	NO	YES	YES	YES
Real Time Clock	10	YES	YES	YES	YES	YES	YES
COBOL	10	YES	YES	YES	YES	NO	NO
Macro Assembler	10	YES	YES	YES	YES	YES	YES
Avg. Disk Access Time	20	- -	30 ms. 100 meg (3330)	36 ms. 88 meg (RP04)	32 ms. 67 meg (M16)	30 ms. 80 meg (9321)	35 ms. 84 meg (4138)

FIGURE 21 MINI-COMPUTER FEATURE CHART

FEATURE	WEIGHT	DESIRED	IBM 370/ 158	DEC PDP 11/ 70	INTERDATA 8/32	SEL 32/ 55	MODCOMP IV 25
Coding Scheme	NR	- -	EBCDIC	ASCII	ASCII	ASCII	ASCII
Microprogramming	10	YES	NO	NO	2K WCS+ROM	ROM	ROM
Sync Comm.	5	YES	YES	YES	YES	YES	YES
Async Comm.	5	YES	YES	YES	YES	YES	YES
Bisync Comm.	5	YES	YES	YES	YES	YES	YES
SDLC	5	YES	YES	NO	NO	NO	NO
Multiprocessing	5	YES	YES	YES	YES	YES	YES
Direct Access Method	10	YES	YES	YES	YES	YES	YES
Price: CPU + 256K	10	$75,000	$1,800,000 (512K)	$80,000	$70,000	$60,000	$55,000
Price: 500M bytes	10	$100,000	$148,000 (3330 MOD 11)	$150,000 (6 RP04)	$110,000	$135,000	$150,000
Vendor Interest	10	YES	- -	YES	YES	- -	- -
Vendor Willingness	10	YES	- -	YES	YES	- -	- -

FIGURE 21 MINI-COMPUTER FEATURE CHART (CONT.)

INSTRUCTION TIMES (Register to Memory)		IBM 370/158 (32 bits)	DEC PDP 11/70 (16 bits)	INTERDATA 8/32 (32 bits)	SEL 32/55 (32 bits)	MOD COMP (32 bits)
	Load	.588	1.30	1.25	.75	1.60
	Store	.645	1.8	2.00	.15	2.16
Integer	Add	.900	1.7	1.25	1.20	2.08
Integer	Multiply	2.000	3.9	3.54	6.00	5.32
Integer	Divide	9.900	8.3	5.80	10.80	11.80
	Compare	.703	1.35	1.78	1.20	2.08
TOTAL:		14.736	18.35	15.62	20.10	25.04

FIGURE 22 INSTRUCTION EXECUTION TIMES

	FEATURE	VALUE	WEIGHT	RATING	SCORE
5	Word Size	16	5	6	30
5	Internal Data Path	16	5	6	30
10	Maximum Memory	2 meg	10	10	100
10	Dir. Adr. Memory	64K	10	4	40
10	Cache Memory	2K	10	10	100
10	No. of Registers	16	10	7	70
10	Levels of Interrupts	4	10	7	70
10	Data Rate (I/O)	5 meg	10	7	70
10	Addressing Schemes	All	10	10	100
10	Instructions	400+	10	8	80
10	Task Switching	Yes	10	8	80
20	Effective Memory Speed	300ns	20	8	160
10	Hardware I/O	No	10	7	70
10	Real Time Clock	Yes	10	10	100
10	COBOL Compiler	Yes	10	7	70
10	Macro Assembler	Yes	10	10	100
20	Avg. Disk Access Time	36ms	20	7	140
NR	Coding Scheme	ASCII	NR	-	-
10	Microprogramming	No	10	0	0
5	Sync Comm	Yes	5	10	50
5	Async Comm	Yes	5	10	50
5	Bisync Comm	Yes	5	10	50
5	SDLC	No	5	0	0
5	Multiprocessing	No	5	8	40
10	Direct Access Method	Yes	10	10	100
10	Price: CPU & 256K	80,000	10	5	50
10	Price: 500M bytes	150,000	10	5	50
10	Vendor Interest	Yes	10	10	100
10	Vendor Willingness	Yes	10	10	100
	TOTAL SCORE:			210	2000

FIGURE 23 DEC PDP 11/70 RATING

	FEATURE	VALUE	WEIGHT	RATING	SCORE
5	Word Size	32	5	10	50
5	Internal Data Path	32	5	10	50
10	Maximum Memory	1 meg	10	10	100
10	Dir. Adr. Memory	1 meg	10	10	100
10	Cache Memory	4bytes	10	10	100
10	No. of Registers	8sts.of16	10	10	100
10	Levels of Interrupts	4 lvls.	10	7	70
10	Data Rate (I/O)	6 meg	10	7	70
10	Addressing Schemes	All	10	10	100
10	Instructions	227	10	10	100
10	Task Switching	Yes	10	10	100
20	Effective Memory Speed	300ns	20	10	200
10	Hardware I/O	Yes	10	10	100
10	Real Time Clock	Yes	10	10	100
10	COBOL Compiler	Yes	10	8	80
10	Macro Assembler	Yes	10	10	100
20	Avg. Disk Access Time	32ms	20	9	180
NR	Coding Scheme	ASCII	NR	-	-
10	Microprogramming	2K WCS	10	10	100
5	Sync Comm	Yes	5	10	50
5	Async Comm	Yes	5	10	50
5	Bisync Comm	Yes	5	10	50
5	SDLC	No	5	0	0
5	Multiprocessing	Yes	5	10	50
10	Direct Access Method	Yes	10	10	100
10	Price: CPU & 256K	$ 70,000	10	7	70
10	Price: 500M bytes	$110,000	10	10	100
10	Vendor Interest	Yes	10	10	100
10	Vendor Willingness	Yes	10	10	100
	TOTAL SCORE:			256	2,470

FIGURE 24 INTERDATA 8/32 RATING

	FEATURE	VALUE	WEIGHT	RATING	SCORE
5	Word Size	32	5	10	50
5	Internal Data Path	32	5	10	50
10	Maximum Memory	1 meg	10	10	100
10	Dir. Adr. Memory	512 K	10	10	100
10	Cache Memory	4 bytes	10	10	100
10	No. of Registers	8	10	3	30
10	Levels of Interrupts	128	10	10	100
10	Data Rate (I/O)	26 meg	10	10	100
10	Addressing Schemes	All	10	10	100
10	Instructions	152	10	8	80
10	Task Switching	Yes	10	10	100
20	Effective Memory Speed	600 ns	20	5	100
10	Hardware I/O	Yes	10	10	100
10	Real Time Clock	Yes	10	10	100
10	COBOL Compiler	No	10	0	0
10	Macro Assembler	Yes	10	10	100
20	Avg. Disk Access Time	30 ms	20	10	200
NR	Coding Scheme	ASCII	NR	-	-
10	Microprogramming	ROM	10	6	60
5	Sync Comm	Yes	5	10	50
5	Async Comm	Yes	5	10	50
5	Bisync Comm	Yes	5	10	50
5	SDLC	No	5	0	0
5	Multiprocessing	Yes	5	10	50
10	Direct Access Method	Yes	10	10	100
10	Price: CPU & 256K	$ 60,000	10	9	90
10	Price: 500M bytes	$135,000	10	7	70
10	Vendor Interest	-	10	5	50
10	Vendor Willingness	-	10	5	50
	TOTAL SCORE:			228	2,130

FIGURE 25 SEL 32/55 RATING

	FEATURE	VALUE	WEIGHT	RATING	SCORE
5	Word Size	16	5	6	30
5	Internal Data Path	32	5	10	50
10	Maximum Memory	512K	10	5	50
10	Dir. Adr. Memory	128K	10	6	60
10	Cache Memory	60 bytes	10	10	100
10	No. of Registers	16sts.of15	10	10	100
10	Levels of Interrupts	16 lvls.	10	10	100
10	Data Rate (I/O)	2 meg	10	2	20
10	Addressing Schemes	All	10	10	100
10	Instructions	255	10	8	80
10	Task Switching	Yes	10	10	100
20	Effective Memory Speed	500ns	20	4	80
10	Hardware I/O	Yes	10	10	100
10	Real Time Clock	Yes	10	10	100
10	COBOL Compiler	No	10	0	0
10	Macro Assembler	Yes	10	10	100
20	Avg. Disk Access Time	35ms	20	7	140
NR	Coding Scheme	ASCII	NR	-	-
10	Microprogramming	ROM	10	6	60
5	Sync Comm	Yes	5	10	50
5	Async Comm	Yes	5	10	50
5	Bisync Comm	Yes	5	10	50
5	SDLC	No	5	0	0
5	Multiprocessing	Yes	5	10	50
10	Direct Access Method	Yes	10	10	100
10	Price: CPU & 256K	$ 55,000	10	10	100
10	Price: 500M bytes	$150,000	10	5	50
10	Vendor Interest	-	10	5	50
10	Vendor Willingness	-	10	5	50
	TOTAL SCORE:			209	1920

FIGURE 26 MODCOMP IV/25 RATING

9.0 DBMS PERFORMANCE IN MINI-COMPUTERS

Tests have been completed on an early stand-alone version of IDMS on the PDP 11/70. After issuing a representative mixture of DML commands in a sample program, an average elapsed time of 80 ms. per DML call was measured. This test made use of no special hardware or peripherals. The test issued 620 DML functions as follows:

STORE	50
OBTAIN	360
FIND	15
MODIFY	180
DELETE	15
	620

In order to perform these particular DML functions for the DBMS module issued 1698 I/O operations split into 924 reads and 774 writes. The reason for the large number of I/O operations is that the PDP 11 direct access methods requires blocks to be formatted to a size of 512 characters. In this test the IDMS page was 1024 characters in size requiring 2 I/O's for each page read or write.

The same measurements on the IBM version of IDMS on a 370/158 gave an average time of 40 ms. per DML call. The different between the 40 ms. average time for this problem on the IBM 370/158 and the 80 ms. average on the PDP 11/70 can be explained as follows:

Dual I/O for each read or write (2.7 I/O operations per DML function)	20 ms.
Speed difference in 3330 and RP04 (1.3 operations per DML function)	8 ms.
2 task switches in PDP 11 version per DML call	6 ms.

Instruction speed difference (see Figure 22) 30% of CPU time	3 ms.
Implementation differences required by 16 bit architecture	3 ms.
	40 ms.

A new software feature will alter the PDP 11/70 times: multi-sector I/O or multiple chained reads for one start I/O command. This feature is being implemented by DEC at Cullinane's request and will reduce the average time from 80 ms. by the 20 ms. stated above to an expected 60 ms. per DML call. Optimization of the PDP 11 version of IDMS is expected to eliminate the task switching resulting in an expected timing for this particular problem of 54 ms. per DML call. This number is used in Figure 27.

The average instruction time of the Interdata 8/32 is 20% faster than the DEC PDP 11/70. In addition, the Interdata 8/32 processes 32 bits at a time like the 370/158. With the large direct addressability of the Interdata 8/32 no task switching is required to process DML functions. The average access time for RP04 disk on the PDP 11 is 36 ms. and for the M46 disk on the 8/32 it is 32 ms. In view of the above, we estimate the execution time of this same sample problem at 45 ms. per DML call on the Interdata 8/32. This assumes that any implementation problems or software deficiency such as those experienced on the PDP 11 would be corrected by Interdata as they were by DEC.

As a result of these timings and estimates, the following table can be presented for a stand-alone version of IDMS running this particular sample problem:

MACHINE	ELAPSED TIME
IBM 370/158	40 ms. per DML call
DEC PDP 11/70	54 ms. per DML call
Interdata 8/32	45 ms. per DML call (estimated)

FIGURE 27 BENCHMARK PROBLEM TIMINGS

10.0 RECOMMENDATION FOR THE DEVELOPMENT OF A PROTOTYPE BACK-END DATABASE MANAGEMENT MACHINE

As a relatively quick and economical way of demonstrating the feasibility of a host/back-end database management system, the Cullinane Corporation recommends the implementation of a prototype system using the Digital Equipment Corporation PDP 11/70 computer linked to an IBM 370 host machine. Since IDMS (Integrated Database Management System) is already operational on the PDP 11, such an approach would allow Cullinane Corporation to concern itself with other considerations and problems pertinent to back-end database management. Another important consideration is that there are so many computers in the PDP 11 family either in use or contemplated being used by the Federal Government, that should such a system prove satisfactory, there would be a ready market for additional users of the prototype or enhanced system.

In summary, by using a PDP 11/70 computer the Cullinane Corporation could develop a host/back-end database management system prototype within a year of project start. Any other approach would require a minimum of two years or longer to develop a prototype with great additional expense to the Cullinane Corporation and/or the government.

APPENDIX A - MINI-COMPUTERS REVIEWED IN THIS STUDY

CAL DATA-1
Computer Automation LSI 2/60
Computer Hardware CH 12130
Data General Nova 840
Data General Nova Eclipse
Digital Computer Controls D-116H
DEC PDP/15
DEC PDP/11/45
DEC PDP/11/70
Digital Scientific Meta 4
Electronic Associates Pacer
General Automation SPC-16/85
GRI-99 Model 50
Harris System 240
Hewlett-Packard 2108A
Hewlett-Packard 3000CX
Honeywell System 7000
Information Comp. System ALP3
Interdata 7/32
Interdata 8/32
IBM System 7
Lockheed MAC 16
Micro Data 3200
Modcomp IV/25
Nanodata QM-1A
Prime 300
Raytheon RDS-500
ROLM 1602
SEL 32/55
Texas Instruments 980B
Univac 1816
Varian V75
Westinghouse 2500