*This paper is a discussion of a methodology, a distributable information system model, and an experiment used to identify potential problems for supporting such a system. The experimental model was designed and implemented in an evolutionary manner for the purpose of studying the feasibility of a system with the postulated attributes. Incentives for distribution and design of the study introduce the two main topics—the study model itself and the implementation of the study model. Results of the study provide insights into such factors in distributed information system structural design as intercomponent communication, system control, and recovery philosophy.*

# A distributed information system study

**by K. Ziegler, Jr.**

Over the past several years, we have been working very closely with computer users to determine ways of reducing the complexity associated with new business functions and ways of taking advantage of new technologies. One such technology is distributed data processing, many concepts of which have been discussed in the literature.[1-3] To test the feasibility of such a new information system design, limited experiments and walkthroughs are designed to identify problem areas before they are encountered by the commercial marketplace. Such studies involve identifying generic areas that require work, providing structure for describing environmental characteristics, and analyzing implementation alternatives. This paper discusses insights derived from such a study concerning distributed information systems.

The key incentives that motivate a business firm to assess distributed systems usually fall into one or more of the following categories: subjective, economic, technical, organizational, and a desire to reduce the operational dependence on a single resource. Subjectively, management must assess distributed systems be-

cause of industry pressure and for protection against a lost business opportunity. Economic considerations are those of potential savings in communication costs, the sharing of costly resources, a higher responsiveness to new business applications, and higher user productivity. Technical considerations stem from the opportunity to reduce the response time for specific applications, to improve availability to the end user, and to reduce the complexity associated with trade-offs required for a single node-system supporting all applications. Organizationally, distribution with communication links provides a vehicle to relate data processing to organizational structures and organizational information flow. By separating and distributing business applications, the impact to the business by a single outage can be reduced. One application area that is being considered for distribution is an information system. The overall goal of an information system is to provide controlled access and sharing of data within an organization. As dependency on these data grows, more formalized techniques to manage the access and flow are required, and the organization soon recognizes a need for coordination, security, integrity, synchronization, consistency, timeliness, and availability. The organization also recognizes their accompanying costs.

An early approach to these matters was to implement a centralized data base. This was very effective initially. However, the centralization of data processing and development personnel present some problems. This had the effect of adding greater effort on maintaining satisfactory availability, large up-front planning, management of acceptable response time, responsiveness to users' application requirements, and maintenance of privacy. It also affected the organizational philosophies of the enterprise.[4-6]

A *distributed information system* purportedly allows flexibility by reducing implementation constraints of traditional centralized information systems. The intent of such systems is to provide a logically integrated information system while providing for physical distribution of data over two or more computing facilities. Thus, an authorized user can access data from any of the participating computing facilities in the same manner as from a centralized information system. From an organizational standpoint, this capability puts data processing power where it is needed, while still allowing management control.[7] It also provides for more coordinated information flow in decentrally organized enterprises.

Distribution of information emphasizes certain technical and managerial considerations. Technical considerations include assessments of the following: a communications system to transmit data, dictionaries and directories to identify and locate the data, and data management systems to provide the required data syn-

chronization, integrity, consistency, privacy, security, and data placement. Managerial considerations involve the following: formulation and analysis of the effects of maintenance strategies, hidden costs and risks associated with design, implementation, optimization, vendor coordination, legal aspects (interstate and international), auditing, system administration, standards, and responsibility.[3,8]

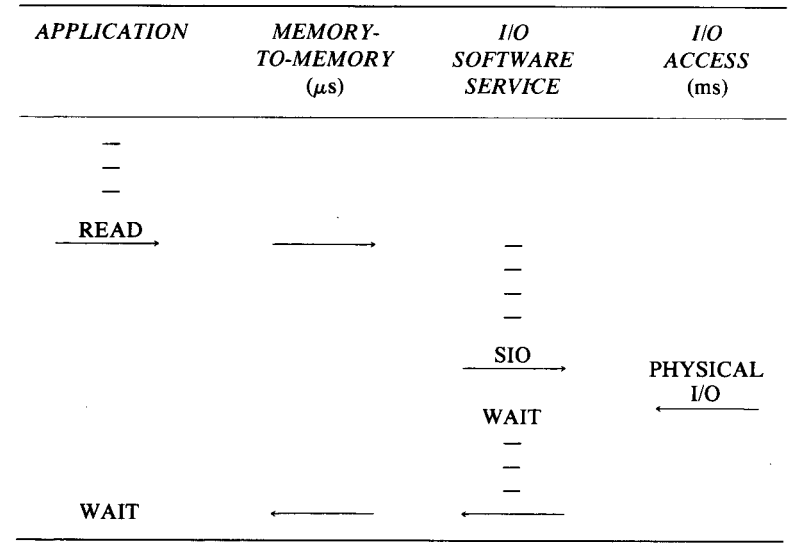
Although there are numerous papers that define "distributed," these definitions depend on the perception of each user and his specific application implementation. For this reason, the terminology used here is intended to communicate concepts only, rather than to assert a standard. In this paper, a *node* is defined as a processor or processors with a single operating system. A *distributed system* interconnects multiple homogeneous and heterogeneous nodes to exchange data, share resources, and execute multiple well-defined and interrelated software components (programs) in one or more nodes. Multiple nodes are not necessarily involved in any particular process; the system can provide an integrated appearance when multiple nodes are involved.

The primary sources of difference between distributed and nondistributed systems are in the *distribution of control* within the network and the *nonmemory connections* between software components. Distribution of control involves the allocation, finding, and accessing of network resources and the synchronization of such services as update and maintenance. Nonmemory communications via channels and lines require more sophisticated protocols to ensure data transfer. As an example, after a message is sent, some acknowledgment of successful receipt is appropriate to indicate an error if the transfer has failed. If one examines the process of communication to a subroutine at a remote location, the increase in elapsed time is apparent, as shown by comparing Figures 1A and 1B.

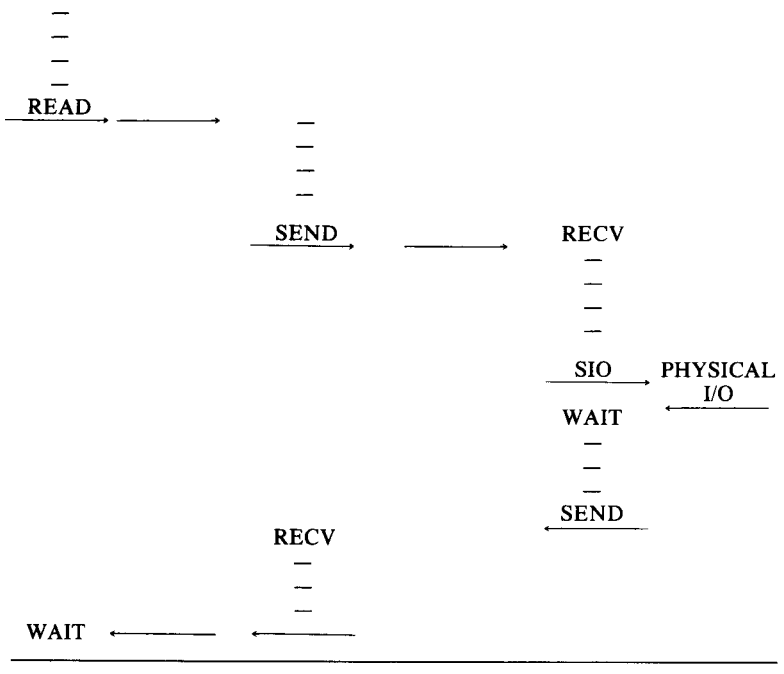
## Incentives for distribution

Such differences force new approaches to ensure adequate response time, availability, data integrity, backup, recovery, efficient use of resources, and security. These approaches may also require new procedures for processing business transactions, variations of redundant data distributed throughout the geographically distributed information system, and schemes to periodically synchronize the data.[9] This new environment involves techniques to determine the useful level of data distribution, the location of the data, internode and resource control strategies, and response-time managers, as well as monitors for evaluation and tuning.[10] Staff skills, combined with good alternatives in design, determine the degree of programmer and user transparency, economy, and manageability of the distributed system.

Figure 1 Example connection comparison: (A) Flow using memory-to-memory connection; (B) Flow using nonmemory connection




(A)




(B)

As a result of the increase of on-line workloads, users often experience degradation of system response time and availability. Users have also been required to submit to unplanned expenditures for conversion, rewrite, or "minor adjustments" to programs or the current way of doing business in order to utilize the computer resource effectively. Consequently, users are becoming more result oriented.

**isolation**

Advances in computer technology have emphasized the desirability of dedicating business applications to autonomous small computer hardware and software. This provides the individual line manager with the ability to control his own resources and service levels in a manner that transcends the methods of the currently implemented, managed, and controlled centralized complex. This also isolates the user from effects of unpredictable changes in priorities and system tuning.

**complexity**

The data processing department, in its effort to serve all users within budget and personnel constraints, must manage the sharing of resources in spite of the contradictory objectives of performance, economy, availability, productivity, and responsiveness. This effort is becoming increasingly more difficult and often exceeds the technical staff's ability. Loss of control has resulted from the increasing number of software and hardware components and applications that must coexist.[11]

Dedication of business applications to autonomous systems appears to simplify the system management and control tasks by reducing the number of component interdependencies and tradeoffs.

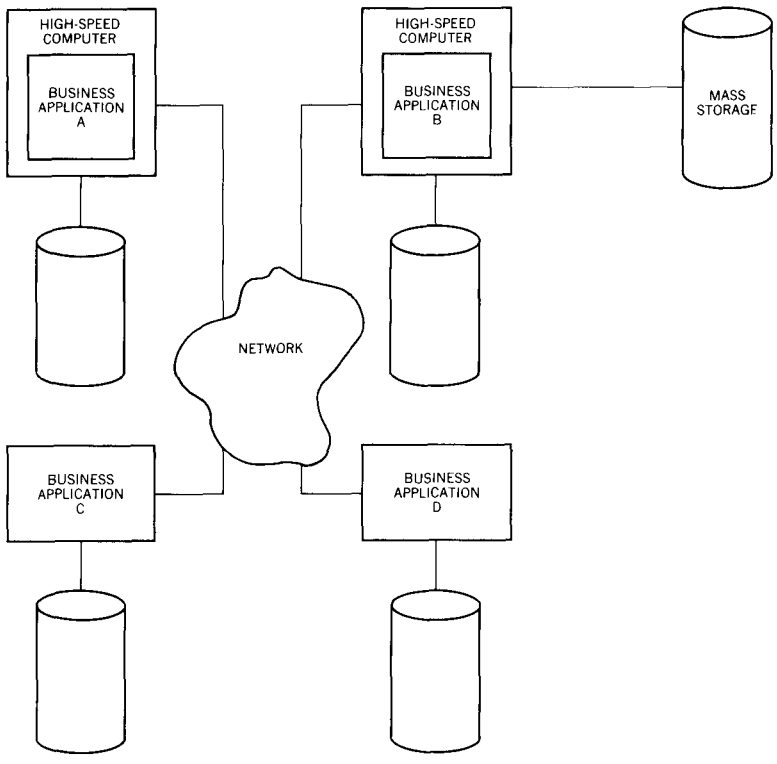
**availability and performance**

The distribution (or dedication) of hardware to business functions also makes it possible to reduce response time and communications costs and provide higher availability. These advantages can be realized if data and processing services can be placed at or near the point of work. Then the business application is less susceptible to disruptions due to line outages and uninvolved component failures.

**resource sharing**

Distribution also allows for maximizing isolation while providing a high level of sharing of costly resources, e.g., high-speed computing and mass storage, as shown in Figure 2. The system characteristics thus far discussed appear to provide a solution to some of the limitations of centralized system complexes. An assessment of these characteristics is in order because it is becoming increasingly apparent that data requirements, performance, and availability needs will continue to grow as data processing becomes cheaper and more integrated into the business organization itself.

Figure 2 Resource sharing




## Motivation for study

From a user perspective, the distribution of processing is a way of achieving simplicity, isolation from other users of the system, and better service. However, the dedication of an entire node and attached resources to service a specified set of users often has economic shortcomings, because fixed components (such as operators, peripherals, and software) often increase when the work is split.

Distribution may constrain the user to a limited set of data and services, unless changes are made to the information system implementation to allow data redundancy and new data synchronization. This may result in less than optimal resource utilization and involve a net increase in total data processing expenses. This cost is one that many businesses appear willing to pay to satisfy the increase in demand while maintaining response time and availability. Unfortunately, if left unchecked, this may result in a return to decentralized uncoordinated data processing, with all the disadvantages experienced in the pre-data base era.

Since data are corporate resources, system planners must focus on the access to data, and the ability of several data base management systems to cooperate.[12] This implies that system coordination and network-wide standards are even more important in a distributed environment.

In an effort to determine the best approach to distributing data while meeting immediate and long-range business requirements, system planners must understand the various ways of implementing and distributing systems, with their associated strengths and weaknesses. It is important for the implementation to be extendible without stretching the state of the art.[3]

Other considerations (outside the scope of this paper) include organizational issues,[4–6] managerial and economic considerations,[3] costs minimization,[13] international systems,[8] distributed data base characterization,[14] security,[15] market and people factors,[11] and the whole realm of data communications, just to highlight a few.

In current literature there are numerous articles describing new variations of distribution of processing, data, and applications, with new advantages (and some disadvantages). They involve such terms as "partitioned, replicated, discrete, horizontal, and vertical" for data base distributions, and "centralized, replicated, global, and local" for directory distributions.[14,16,17] Note that the directory is a data base of locations, names, access authority, etc. that will be partitioned and replicated for availability and performance reasons.

## Design of study

Among the questions that vex the designer of a distributed system, whose previous experience has been with centralized systems, the following have been selected as important to this study:

- Intercomponent communication;
- System control;
- Response control;
- Recovery responsibility.

More specifically, we wished to learn as much as possible regarding software interfaces and support facilities for distributed software components. An information system was used as the test business application. The study itself had the following objectives:

- Test the technical feasibility of connecting IMS DB/DC applications into a heterogeneous information system network without any application program changes:

- Determine the physical distribution of IMS-like logical functions for a single distributed information system occurrence;
- Test system support facilities necessary to support such a distributed implementation.

We decided that the study design should try to maintain the current application software investment while providing distributed services, because many current application programs depend on standard system and service interfaces, as well as certain support functions. These support functions are termed the *environment*, which is often implied as a basic element of program logic.

As an example, the automatic serialization and backout of several transactions concurrently while updating the same data segments within an IMS data base represent an *implied support function.* The IMS application programmer assumes this service and therefore provides no contingency for its absence. Thus any study design that proposes compatibility or coexistence must be able to simulate the environment, so that the application logic may remain untouched. In other words, it would not be sufficient to provide source code compatibility without the appropriate environment services.

**intercomponent communication**

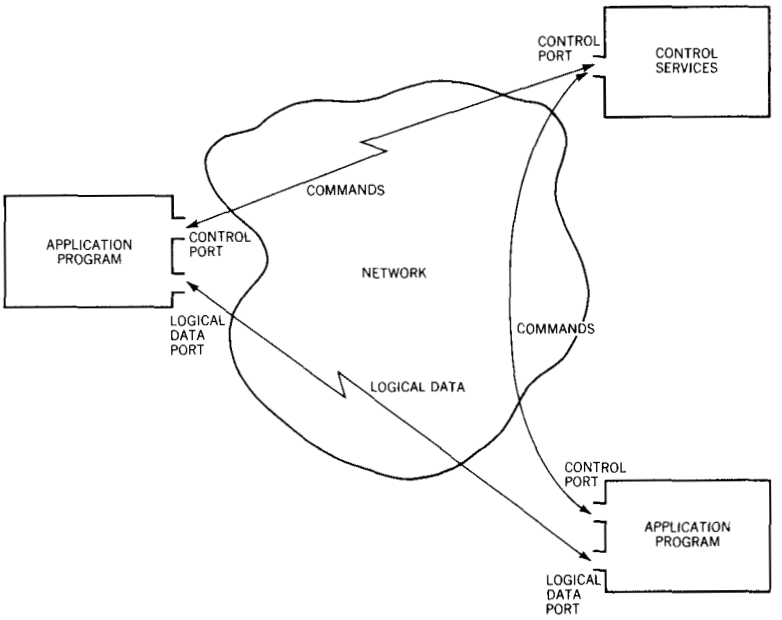
Program-to-program communications should include facilities to:

- Locate the desired component (if it already exists);
- Initiate a component (if it is not currently active);
- Terminate a component when it is no longer needed;
- Transfer control to the component;
- Communicate without passing control;
- Recover from temporary disruptions (reconnect, retransmit, etc.).

These program-to-program communications were viewed as taking one of two logical forms: control or data. Control messages (commands) are sent to a system control service that handles such requests as initiating a program, locating a component, and terminating a program. Data messages (application communications) are sent to the destination software component. Conceptually this might be characterized as two ports into a network, one for control, the other for data, as indicated in Figure 3. This allows control services to be outside the origin or destination nodes. The control port is analogous to current operating system supervisor service requests, e.g., Supervisor Call (SVC) or branch entries. Like service requests, the control port need not be in session with a specific service. The request is routed to the appropriate service only when needed.

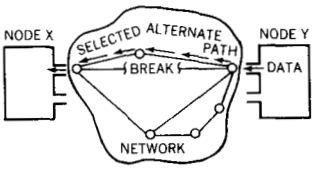
The data port is simply a way into the network used for communicating with one or more other application programs. The messages

Figure 3 Logical data ports and control ports




can then be routed by the network to the appropriate destination, based on a destination address within the message.

Figure 4 Alternate path selection

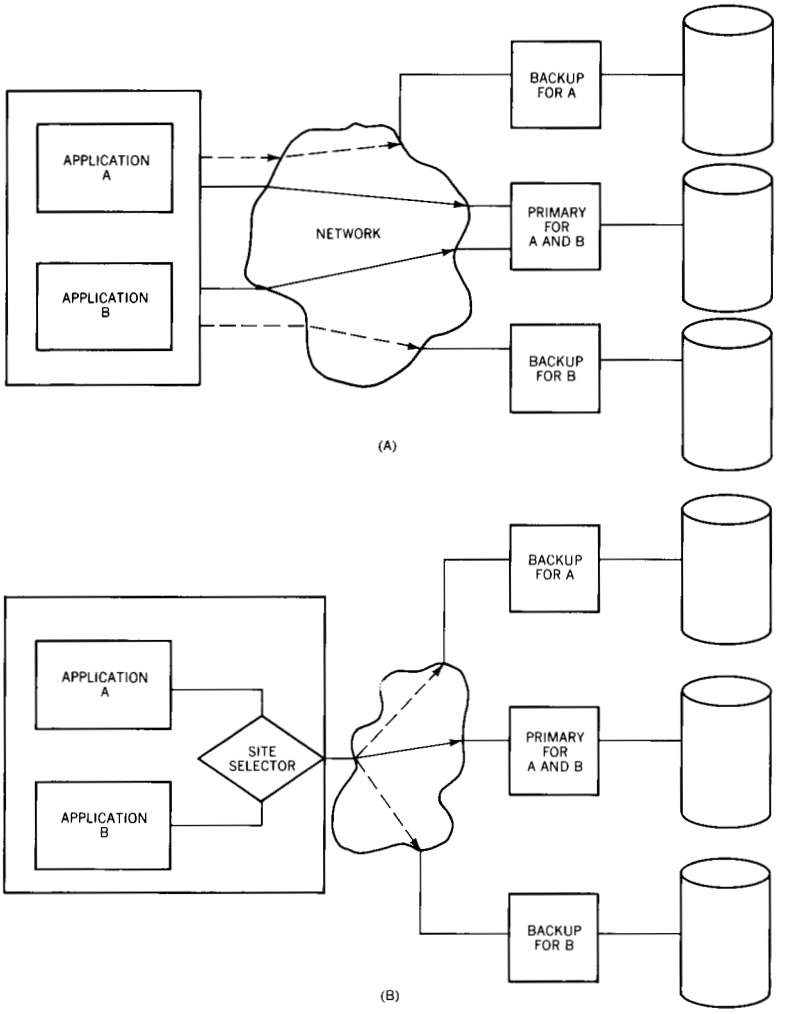



Message routing through data and control ports provides a way of minimizing the use of control blocks and also offers flexibility in dealing with instantaneous network changes, such as alternate destination, alternate servicing, or intermittent outages, as indicated in Figure 4.

Since a node could call other services, either locally or on the network, we were faced with deciding whether two application program interfaces should be provided—one for internal communications (within the same node), the other for external communications (internode). If two interfaces were available, the programmer would have to decide which interface to use, and that would make the program sensitive to its placement. An alternative would be to provide a single high-level application interface that treats all communications as internode. The issues appeared to be ones of storage integrity, security, and instruction path length, where primary differences stem from program portability (independence) and the way of handling storage.

For the purpose of this study, a single application interface was selected for both inter- and intranode communication, so as to protect the software investment and reduce storage integrity and security exposures, at the price of additional path lengths.

Figure 5 Backup site selection: (A) With peer control, the application selects the backup site; (B) With hierarchical control, a site selector selects the backup site

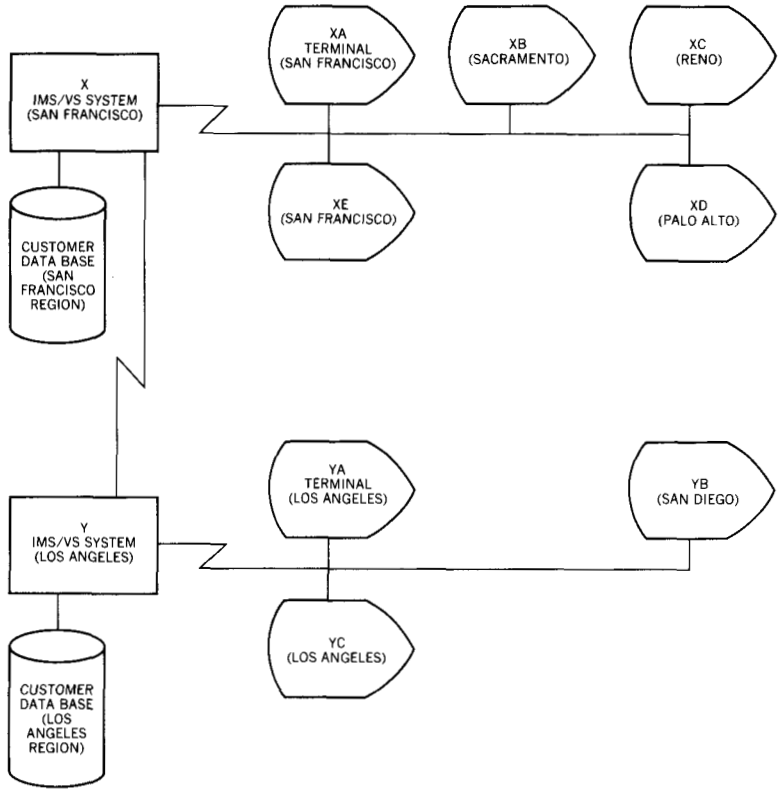



**system control**

A *hierarchical approach* to managing and controlling multiple processors that share resources employs a specific processor as the "master" and all associated processors as "slaves." The master is a critical resource that supplements many of the slave resources and services. When such dependencies are allowed, the operation of the slaves may be disrupted if the master fails or is made unavailable for maintenance. An approach to solving this problem is to provide multiple hierarchies, with the masters able to communicate with one another for access to outside data.

Another technique to reduce disruptions is to provide an alternate master when the primary is unavailable. This introduces logistical problems in data-oriented systems, where the placement of the

Figure 6 IMS multiple system coupling

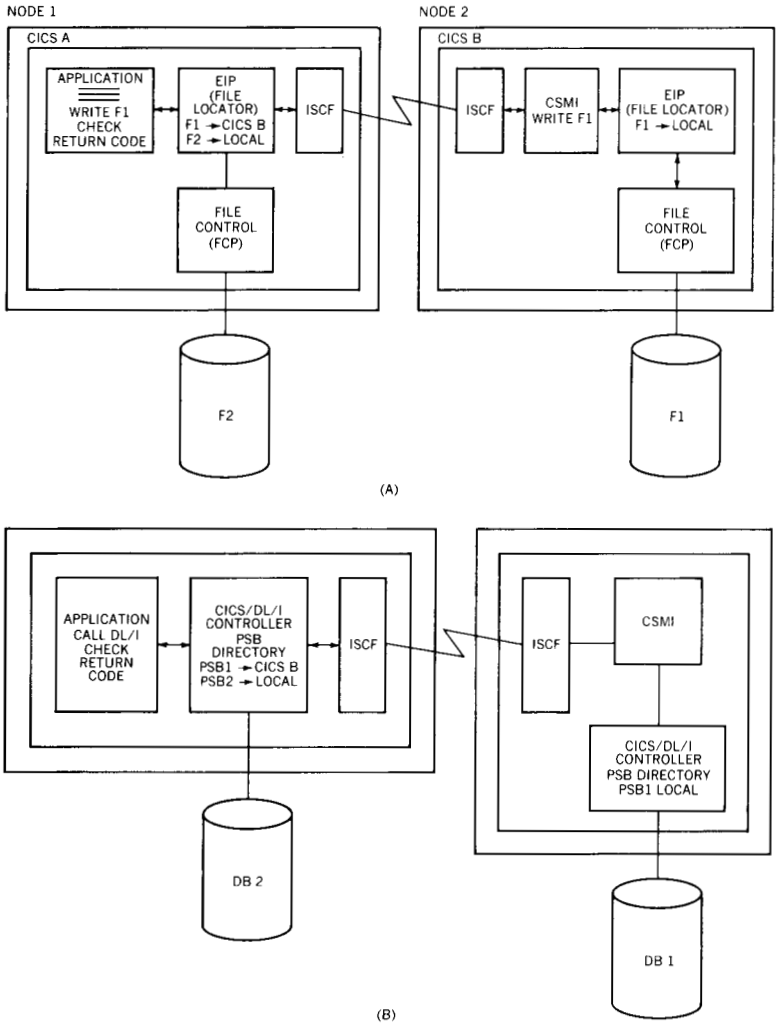



data and control is essential to continued operation. A hierarchical data-oriented system switch-over to a secondary master requires extensive application and dependent data insights. Passing control back to the original master involves extensive prior planning and support programs.

A *peer approach* emphasizes autonomy, in which all processors can be masters. Each node and its applications are responsible for their own operation. If a program in another node wishes to access some external resource (data or service) node, the servicing node manages and controls the request. In this approach, each application is responsible for its own recovery.

To compare and contrast the hierarchical and peer approaches, Figure 5 depicts two applications in a node, each with a different backup node and data base. The peer approach (Figure 5A) makes the application responsible for the selection. The hierarchical approach (Figure 5B) provides a control function for recovery site selection, on the basis of parameters and specific failure characteristics.

Figure 7 (A) Example of remote file access using CICS/VS
(B) Example of remote DL/I access using CICS/VS




IMS multiple system coupling (Figure 6) is an example of a specific peer approach implementation of a distributed information system in which the current unit of communication is at the IMS transaction level. CICS/VS represents another possible implementation of the peer approach, and its current unit of communication is at the file call or Data Language/I (DL/I) call level (Figure 7).

The peer approach was selected for this study because it appeared to be an effective strategy for managing resource sharing while protecting the primary owner of the resource. It also provided flexibility to control resources in a hierarchical fashion. A

hierarchy could be implemented when dependent software services were distributed to multiple nodes such that they were functionally managed as a hierarchy, although the nodes are peers for resource ownership and operating systems.

**directory**

The design of the directory for distributed systems should accommodate the transient nature of nodes in a network, because the nodes in a network environment may not all be operable at any one time. Therefore, the directory design should accommodate the dynamic nature of network states and avoid excessively large space consumption for nonessential entries. To address this, a structured network of directories was selected.

The structure is provided by participating components, whereby each component identifies itself to the appropriate directory or directories via basic guidelines and system defaults. Each data set, therefore, requires some scope-of-use data associated with it. For example, if a data set is intended for the exclusive use of a particular node (e.g., the one to which it is connected), a "node only" scope of use would be indicated, and only the directory for that node would record the entry. This selection could be provided via Job Control Language (JCL), the application program, or the operator. The scope might be a node, a set of nodes, or the entire network. After analysis, we decided that it is the resource's responsibility to remove itself or alter related data when a change occurs. We did this as an alternative to a system that would require a sophisticated allocation/deallocation function.

**asynchronous requests**

Time delays resulting from the transfer of requests and data through a network also required attention. Transactions that require services or data from other nodes may experience severe increases in overall response time if requests are executed serially or synchronously, i.e., issue first request and wait for response, issue next request, etc. *Asynchronous request* support allows the application to issue multiple requests before waiting for data return or receipt confirmation. This tends to reduce the transaction response time experienced for multiple geographically distributed data requests. Asynchronous request support is illustrated in Figure 8. Other ways of dealing with delays greater than that of memory-to-memory transfer include: (1) packaging requests with some decision logic; (2) using change-type requests, e.g., ADD A to field B, as used in IMS/VS fast path main storage data base updates, and (3) set-oriented operations, such as those provided by a relational data base interface.[18] These techniques attempt to reduce the number of interactions between network components for a particular transaction. A change-type request is illustrated in Figure 9B and a group-type request in Figure 9C. (Figure 9A is a traditional data access.) Since all these techniques require application program logic changes, they are candidates for use with new applications.

Figure 8 Asynchronous requests

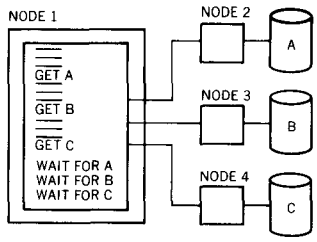
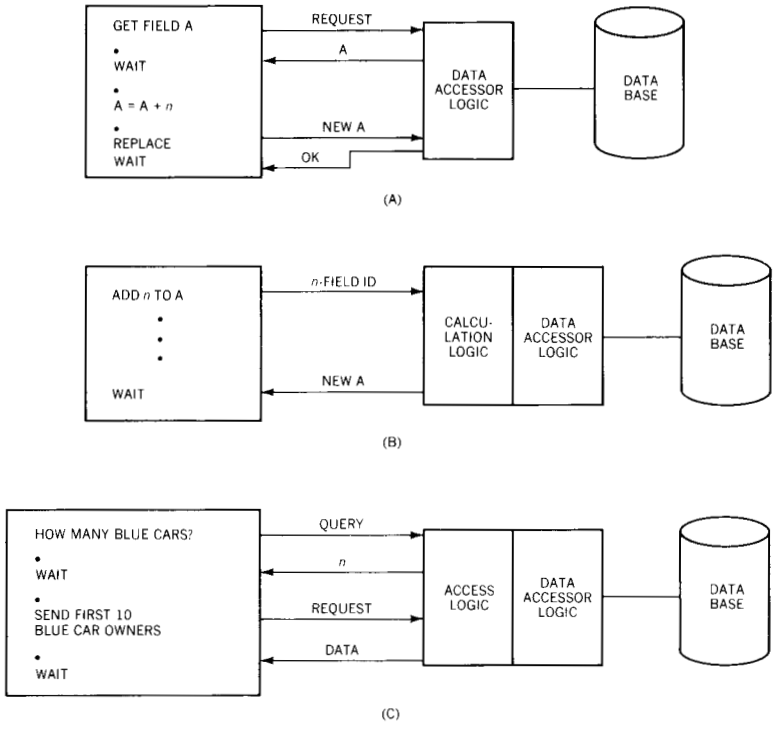


Figure 9 (A) Traditional data access; (B) Change-type request; (C) Group-type request




**point-of-origin responsibility**

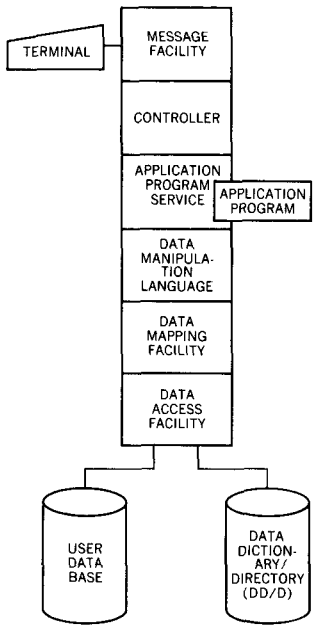
An error detection and recovery strategy also had to be established for a distributed systems environment. A controller that detects component loss and controls valid response was considered. We decided that this would require extensive insights into individual transactions and associated components. The implementation of a controller approach was believed to be very complex and potentially troublesome, because a failure could well bring down the entire system.

An alternative to such a controller is to delegate responsibility to the originator of a message. If a connection is interrupted, the originator must sense the interruption and bear responsibility for initiating corrective action, e.g., establish a new connection, seek an alternate path, wait for restoration, or ignore the loss. This is called *point-of-origin recovery*, and was the alternative selected.

## A distributed information system model

The next step was to derive a distributed information system conceptual model, shown in Figure 10. The components of the model

Figure 10 A conceptual model of a distributed information system




could reside in one or in multiple nodes and still maintain their systemic relationship, in order to present a single information system occurrence, such as is shown in Figure 11, with a centrally controlled information system and distributed data and software components. Geographic distribution of this configuration is of dubious value, whereas a local distribution has potential performance and capacity benefits.

The distributed information system model is an attempt to describe the basic distributed functions that can be mapped to any data base/data communications oriented system. The premise for this model is that multiple application implementations—not necessarily using the same interface or same facilities—must be accommodated. For our purposes, programs using and written for IMS were used for the base case application programs.

For geographically distributed nodes, *multiple* autonomous information systems interrelated and interconnected via the network are more practical. Our model accommodates a multilevel (among all peers) information system with single or multiple domain access capabilities to support the various levels of data distribution. This approach is desirable because it accommodates various degrees of autonomy while maintaining the appearance of an integrated information system. The key is a comprehensive dictionary/directory. In this configuration, each occurrence of the information system can execute alone and provide its own recovery, in order to maintain availability for its specific (primary) set of users.

The structural flexibility of the model enables the system implementor to customize specific nodes to provide an economical or expedient configuration for addressing user needs without affecting business application programs.

The model also accommodates heterogeneous hardware or software architectures if the interfaces and service functions are maintained. This concept is analogous to the relationship between such nodes as IBM 8100 Information Systems and IBM System/370, which, although they have different hardware and software architectures, communicate via a common set of rules (protocols) that were established by the Systems Network Architecture (SNA) and implemented using VTAM.

**component description**

The *message facility* in Figure 11 provides terminal interface services (e.g., message formatting, logging messages, input validation, and the routing of messages to the appropriate controller component). It provides point-of-origin service, and can be used to communicate with multiple information system controllers to provide performance benefits with alternate or duplex system selection for availability and recovery.

The information system *controller* component provides for application and service management, queuing messages, and synchronizing information system logs. Each information system domain has one controller component. The controller represents the focal point of each occurrence.

For productivity purposes, application programs are encapsulated in an environment—an *application service*—which manages the interfaces used by the application to participate in the information system software, as shown in Figure 12.

A *data manipulation language facility* provides a compiler-like service to translate an application data manipulation language into an internal data manipulation language. (This facility may be bypassed for statically bound programs that do not utilize dynamically created data manipulation commands.) The resultant internal data manipulation language statements are then passed to the appropriate data mapping facility.

The *data mapping facility* is the heart of the model; it provides security checking, logical request logging, data mapping, and data building (for logical data bases that are built dynamically for the duration of a transaction). Each one has the exclusively controlled data access facilities shown in Figure 13. The data mapping facility routes and controls data requests to other peer information systems. Communications among data mapping facilities provide for synchronizing the internetwork updates and logs while still allowing them to continue when failure occurs. The mapping facility is also responsible for the logical data bases in terms of integrity, performance, backout, and recovery for each occurrence, as well as deadlock detection and synchronization of redundant data between occurrences at specified intervals. The data mapping facility in this model detects deadlocks by using timer signals (or "pops") and status and reservation checks with the other data mapping facilities.

The data mapping facility also provides the Data Dictionary/Directory (DD/D) service, which contains information about the data and data bases and their associated locations. Although the scope of the information in such a data dictionary can encompass all system logical and physical resources (including terminals, files, reports, etc.), the focus in our study was on how to implement it to find the location and properties of a data element in a geographically distributed system. This requires the selection of the appropriate data mapping facility through globally known symbolic names. These names are associated with logical addresses, and they are resolved to a physical node by the network services. In a static environment (one that has all access paths previously identified), such elements can be placed in all directories. In our study, however, a dynamic environment was more useful.

Figure 11 Systemic relationships are maintained from node to node; (A) is the same as (B)

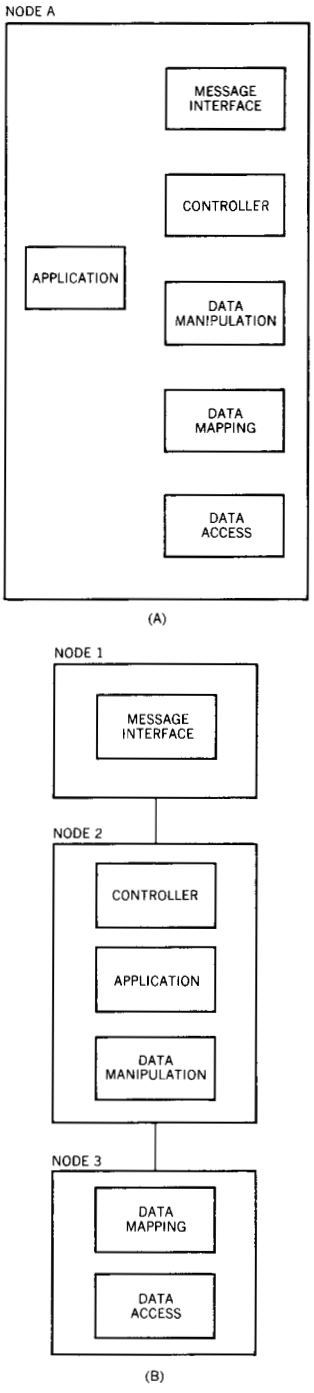
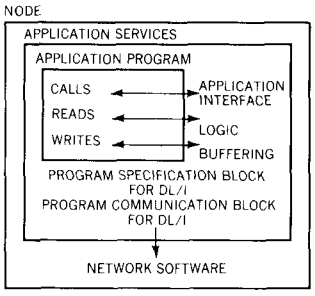


Figure 12 Application service provides the logic necessary to interface with the information system




The *data access facility* provides the logical data interface with the system access methods and storage subsystems, e.g., VSAM and DL/I data base access method and language. It provides such functions as pointer following and physical data base update logging, and is responsible for physical data base access and recovery.

## Implementing the study model

In order to constrain the scope of the study, only the minimal functions and options of a distributed system were explored. Most of the effort concentrated on the novel areas introduced by distribution, such as program-to-program communications, application interfaces, director services, and distributed control.

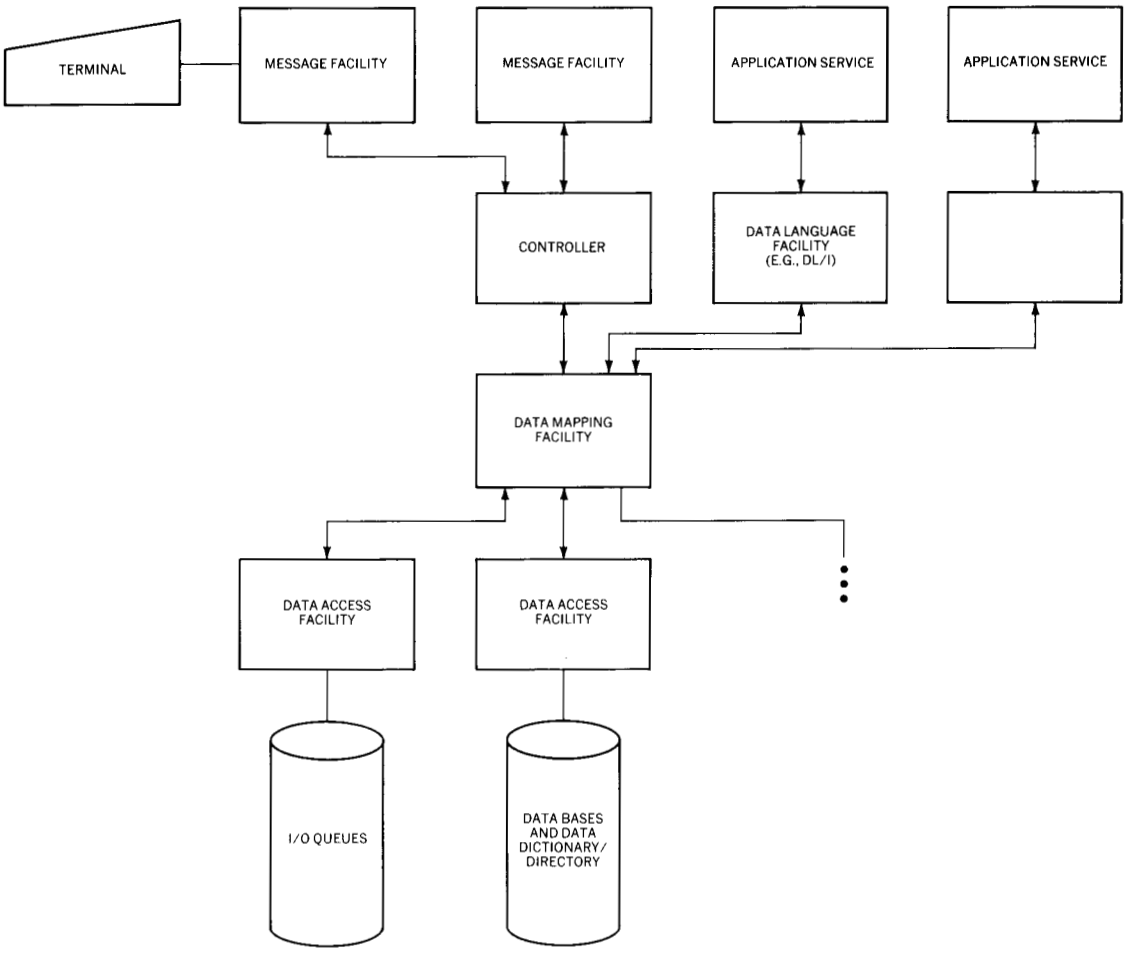
A four-peer-node configuration was selected for testing. The nodes were implemented with the use of VM/370, and the Virtual Machine Communication Facility (VMCF) was used as the vehicle by which the different virtual machines (nodes) could communicate to provide a multiple-node environment. The study model is shown in Figure 14.

With the exception of the modules that used VMCF and CPU timer facilities (which we referred to as "operating system"), PL/I was used as the implementation programming language. The programs were designed to be easy to read, debug, and rewrite. The project entailed the writing of fourteen modules, with 2000 lines of assembler language code and 3600 source lines of PL/I.

The implementation began with the selection of an IMS sample data base, and an IMS transaction was coded to access that data base. This transaction was then used to represent an application program and was mapped into the distributed information system model. The mapping process identified the interfaces and services required to proceed with each functional component of the model, e.g., the application service functions.

The next step was to identify common services in the application service component that could be provided by the support system. Once identified, this system logic was extracted and placed in the operating system services, or network transmission services. The interfaces and support components for transmitting messages in a static predefined environment were then coded and tested, using a driver program (some 300 lines of PL/I executable code) that provided an interactive interface for the application and acted as a master console. This assisted in the development of a protocol for communicating between software components and an application program interface that provided for the transmission of program-level data among multiple nodes.

Figure 13 The controller is the data control point of each information system occurrence




Subsequently, initialization and control strategies for a dynamic environment were designed, coded, and tested. This called for node and network control services with associated protocol and command language enrichments.

**five basic building blocks**

In addition to the information system components, five basic building blocks were developed for our study. They were the Node Control Interface (NCI), the System Control Interface (SCI), the Network Transmission Subsystem (NTS), the I/O Device Subsystem (IODS), and the Storage Device Subsystems (SDS, which was VSAM). These five components provided those primitive functions (such as timers and interrupt, vectoring, and storage management) that are generally available from any operating system or monitor.

Figure 14 Study model using the Virtual Machine Control Facility (VMCF)

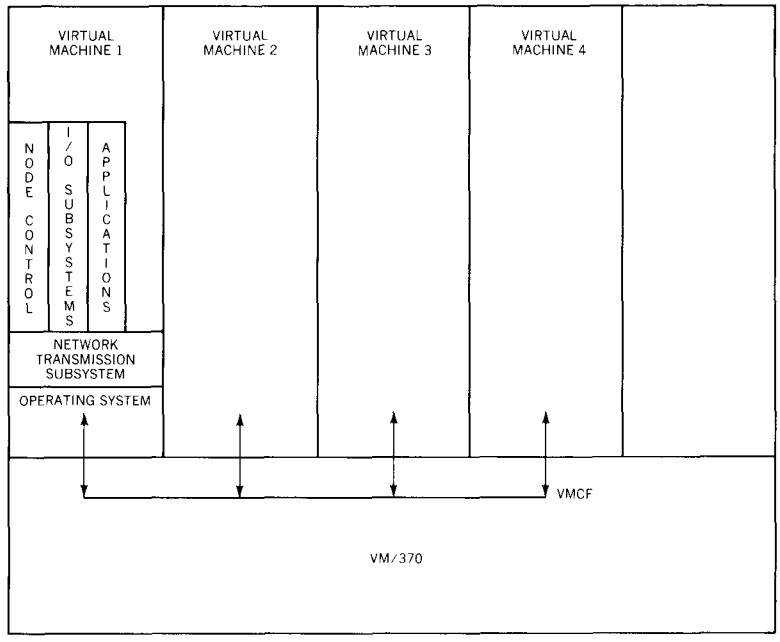



Figure 15 Node without application

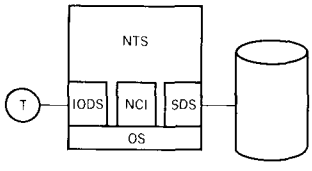


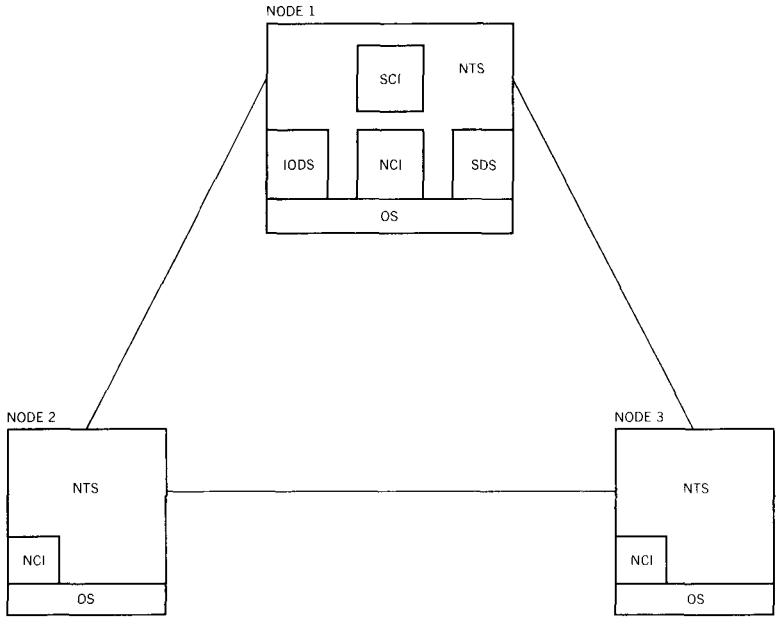

Figure 15 shows a basic stand-alone node before any application is present. The terminal is a control console used to communicate with those operating system services required to bring up any additional components, e.g., input/output device subsystem, storage device subsystem, and application within this node. In Figure 16, node 1 has an SCI, which provides facilities necessary to locate other nodes in the network (nodes 2 and 3) and has the ability to initiate requests to bring up those nodes.

The primary source of control comes from within each component in the network. Each component uses the NTS, NCI, and SCI services to provide logical resource control. These are primarily passive services; that is, they are invoked via an application program request. The NTS is invoked via an external interrupt.

The NTS provides communication among components regardless of where they reside (local or remote). NTS can be viewed as a memory-to-memory or node-to-node pipeline that is accessed via a very primitive interface and provides only point-to-point message transfer service between application programs or subsystem components. For simplicity, the NTS end-to-end protocol for transferring data consists of a destination address, an origin address, and a variable-length text, plus some security information.

The network control services such as transmission sequence checking, alternate pathing, and address resolution are provided

*Figure 16 System control interface hierarchy*




by the NTS without intervention by the application. Network data-related services such as message sequence checking, message retry, and alternate destination require significantly more heuristics. We decided that these could best be provided by the application program, application service, or application subsystem on an as-needed basis.

Each program opens a *port* to the network through which it can issue SEND/RECEIVE-type requests to any destination component address known to the NTS. This component address is established via a symbolic address that is resolved into an explicit address by the NTS. Once the explicit address has been established, the SEND/RECEIVEs are issued. The network subsystem also provides a mailbox facility to which tickler messages to the requesting component can be sent. This was used as a time-out mechanism by the application program instead of setting a timer and then handling timer interrupts. It also provided a mechanism to keep track of outstanding messages and to allow the program to regain control in order to decide whether to continue, abort, or retry another message.

The NCI provides the bring-up, bring-down, and resource allocation services that are traditionally provided by the operating systems. The NCI is logically separate from the operating system services, since it provides only service for managing resources upon request. It provides no automatic control services such as monitoring.

The NCI also provides a node directory service. The directory is the only way to find other components local to the node, e.g., users directly connected, active programs, and subsystem-named resources.

The SCI provides for locating global resources, i.e., resources outside a node (a network directory service). Thus, SCI is interrogated if a resource outside the local node is to be allocated or referenced.

The NCI and the SCI directory and directory services are functionally the same. The directories contain log-on name, internal node and element identifier, component identifier, physical and logical locations, lock/password, and authorization.

The log-on name is the symbolic name by which the system and/or node knows the execution of a component. Internal identifiers (implicit addresses) are created by the NTS at port-open time, and thus provide a way to address network components. A component identifier represents the name of a particular component. Physical and logical location entries provide symbolic location identifiers. Ordinarily these would not occur more than once in the SCI. However, multiple entries may be required to provide independence from the physical topology for testing and backup. This approach to network resource resolution supports such network management attributes as relocatable destinations, dynamic assignments, and adding new network resources in a nondisruptive manner.

The lock/password provides the directory with the locking facility against which inquiries for this component can be checked. If a request is made and the appropriate key is not supplied, the search is unsuccessful, and the attempt may be recorded as a security violation. The authorization field contains the key by which the component can be accessed. During component intercommunication, each component can check an authorization before providing access to that component's service for that message.

The NCI directory is aware only of components that have been identified to it. This reduces nonessential entry information redundancy within each directory (node or system). The SCI directory contains information on components that have identified themselves and wish to be known by other nodes. Only other SCI addresses can reoccur in each SCI directory. This results in two requests for locating outside components: one to locate the other SCI address and the other to make a request of the foreign SCI. This method of identifying and locating appears to be less difficult to maintain than multiple redundant entries in multiple system control interfaces in terms of purging unneeded entries and con-

tent synchronizing. The method also simplifies the creation tasks associated with reestablishing communications among software components.

**application interface**

The application interface is written in reentrant code. Main storage is for message queues and control blocks that are set up at application initiation time. The application service conditions the environment and acts as the primary application interface to the network and other components. Conditioning is similar to that used in the SNA bind process, when logical units are conditioned at initiation. The application interface also provides options to bypass unwanted or unsupported functions, in response to specific service requirements.

If an application wishes to send a message to another application or system resource, the application interface locates the application or resource via the appropriate directory. The location request is followed by a SEND, which causes a message to be built and encapsulated within a header that contains a destination network address, message length, optional password, and event number. When messages arrive at an application program interface, they are scanned for an event number. If that number is the awaited event number (a response to a SEND), the data are placed in the appropriate program buffer and the event number is deleted from the outstanding response queue. Should the outstanding message be timed out, the event number is deleted from the outstanding response queue and a time-out status is returned for the response. If the number is for another message, it is queued on a message stack and awaits the program's RECV request.

**summary of design decisions**

It might be useful to have a summary of our decision process and reasons for developing our study model design. In our design, we were not concerned with response implications that were topology dependent, but only with the actual communication functions.

The IMS transactions would require no recompiles, program or interface changes to conserve the program investment. The internal interface was to use a pass-the-message data interface as opposed to sharing storage in order to simplify the interface and avoid locks and complications involving data integrity and security. Two ports—one for data and one for control—were selected to accommodate the physical distribution and separation of control services and the destination application program. A single port approach would require a sophisticated intermediary to split the data and control flow if these were physically distributed.

The peer node network structure and peer software interface structure would provide a highly flexible approach to node independence and hierarchical control over interdependent sub-

systems and components. The asynchronous request interface would facilitate parallel requests by the subsystem software components to minimize serialization. The applications (IMS transactions) are neither designed for nor do they have an interface for asynchronous requests. The point-of-origin component was selected as the entity responsible for the successful completion of a transmission. This is preferable to having multiple intermediate components attempt to interact and track messages. The latter strategy would require that all the participating components have extensive knowledge about the application if a failure occurred. It was decided that it would be easier from a logic and software standpoint to let the originator determine whether to retry, route to an alternate destination, or try later.

## Concluding remarks

Our study model has allowed for the testing of basic support system component functions associated with bringing up a node, connecting it with other nodes, locating and bringing up the information system components, exploring various approaches to providing distributed system control and program-to-program communication services for a distributed information system, and the testing of a dynamic network directory (or directories) at the support systems level. The study has also provided necessary walkthroughs and paper design of information system interface and data dictionary/directory requirements. Additionally, the study has provided insights into questions associated with application programs communicating with one another, with network control services, and with node control services. Included among the questions are what to do when a failure is detected, how unsolicited messages are handled, what application program changes have to be made in order to deal with such events as response-time fluctuations, lost responses, variable length inputs, and asynchronous request coordination.

A good portion of our effort concerned the managing of responses, failure detection, maintenance, testing, debugging, and problem determination. Time-out approaches were explored for response-time monitoring, failure, and deadlock detection.

Multicomponent and multinode maintenance required a considerable effort because not all components could easily be brought to the same level of maintenance. This resulted in the requirement for a maintenance control directory and a coexistence philosophy for change, where multiple levels of maintenance are active and communicating with one another.

Testing, debugging, and problem determination required transaction-oriented traces. With the number of components and nodes

involved in our study model, and with the absence of synchronization (e.g., a dump of the application and all its related components at appropriate times), the only useful and available method was to track a transaction through the support and application system components. To do this, each component was coded to identify itself and its functional activities.

This experiment aids our understanding of the subtle technical implications of a distributed information system. Additionally it highlights some areas we must explore further. Among the problems we foresee are the developing of automation of operational aspects associated with alternate data destination, retry and data commitment, unexpected messages, errors, and component losses.

Among the conclusions we have reached as a result of the study presented in this paper are the following. An effective application interface for a distributed system must provide services not included in current CALL or READ/WRITE interfaces. Many such services can be provided by an application service function. Access to data anywhere in the network within response-time expectations requires application program interface enhancements (e.g., asynchronous requests, change commands, group calls, etc.) and logic changes. A distributed directory is a feasible approach to maintaining autonomy and providing access to network resources. Timer facilities are critical to response, failure, deadlock, and lost message detection. Greater-than-anticipated buffer and queue storage are required because of longer delays associated with intercomponent communication across multiple nodes when asynchronous or group-command approaches are implemented.

A transaction-level trace facility is necessary for debugging and problem determination. Significant redundant functions can occur in each component, which highlights the benefit of reentrant codes, modular program design, and the importance of maintenance controls. A tool to manage the maintenance levels and applications is necessary to manage the various nodes' software modules; the absence of such a tool cost many hours of debugging.

We also determined that to distribute a specific occurrence of a distributed information system without common storage requires extensive control data, redundancy, and sophisticated techniques to coordinate this control data (e.g., a directory).

## Acknowledgments

The author thanks R. Stevens, H. Titus, and W. Clement for their participation in and support of this project, with special thanks to

F. Springer and J. Weissmann for providing much of the VM/370 expertise required for testing the model.

## CITED REFERENCES AND NOTES

1. B. H. Liebowitz and J. H. Carson, *Tutorial: Distributed Processing, Compcon 77* (Fifteenth IEEE Computer Society International Conference, Washington, DC, September 6-9, 1977), IEEE, Inc., New York, NY (1977).
2. A. L. Scherr, "Distributed data processing," *IBM Systems Journal* **17,** No. 4, 324-343 (1978).
3. J. C. Emery and B. Gilchrist, "Managerial and economic issues in distributed computing," *Information Processing 77* (IFIP Conference Proceedings, Toronto, Canada, August 1977), North-Holland Publishing Company, Amsterdam, Netherlands (1977), pp. 945-955. This is a management-oriented discussion of distributed processing, function, and data in terms of centralized and decentralized systems; their benefits and hazards.
4. S. L. Mandell and F. G. Harold, "Organized Intelligence: Oasis for MIS mirage," *Data Management* **15,** No. 11, 16-22 (November 1977). Proposes as a replacement for MIS an Organization Intelligence Network (OIN). The argument is for a shift in emphasis from the term "management" that has implied concentration on the decision-making process, whereas "organizational" implies communication. An argument for distribution of processing that can still retain centralized authority is based on this shift in emphasis.
5. R. I. Tricker and B. Gilchrist, "The impact of information systems on organizational thinking," *Information Processing 77* (IFIP Conference Proceedings, Toronto, Canada, August 1977), North-Holland Publishing Company, Amsterdam, Netherlands (1977), pp. 213-221. Analyzes issues facing strategic decision-makers and discusses internal organizational problems, third party involvement in the firm, and environmental changes that are seen as constraints on information system development. Included are perspectives of power through information; the interrelatedness of information systems and organizations; the shape of information systems; coherence of organization (organizational, operational, technical), and organizational intelligence.
6. J. F. Rockart, S. Leventer, and C. V. Bullen, *Centralization vs Decentralization of Information Systems: A Preliminary Model for Decision Making*, Sloan School, Massachusetts Institute of Technology, Cambridge, Massachusetts (Draft). Provides a tabular comparison in terms of centralized systems, decentralized systems, and distributed systems for general and organizational considerations, cost factors, personnel considerations, and programming and operations technical considerations. Included is a summary of factors reported by a specific set of enterprises.
7. C. W. Spangle, "The impact of distributed systems," *Computers and People* **26,** No. 12, 7-10, 27 (December 1977). Provides a management overview of the computer industry and the changes resulting from organizational and processor changes. Also included are distributed processing and its associated impact: new possibilities, challenges, management involvement, changing DP manager role, information systems manager, as well as factors for providing success in a distributed processing system.
8. E. H. Sibley, *The Distributed Information System: Its Architecture and Management*, IFSM Technical Report No. 11, University of Maryland, Department of Information Systems Management, College Park, MD 20742 (November 1976). Discusses some of the challenges associated with designing and using distributed information systems from a managerial and technological standpoint. Included are the following: distributed information system functions, general user aspirations, distributed system user needs, international system problems, information systems architecture, centralized data architecture, decentralized data architectures, decentralized data with centralized control, decentralized control, trade-offs in architecture, design, and management control.
9. K. Ziegler, Jr., "Distributed: A New Impetus for Understanding Data,"

*JCITE Proceedings* (JCITE, Jerusalem, Israel, 1978), pp. 311–318. Discusses the growing requirement for data about data (meta data) and chronicles the trend toward eliminating data redundancy, sharing the same data, and the subsequent issues of providing data redundancy in distribution systems.

10. G. Belford, "Optimization Problems in Distributed Data Management," *Proceedings of the Third International Conference on Computer Communication* (Toronto, Canada, August 1976), pp. 297–302. Presents three areas of distributed data management (file allocation, file usage, and data organization) and the difficulties each presents in optimally organizing data. References current literature on the subject.
11. I. L. Auerbach, "The Influence of Market Factors in Future Computer Development," *Future Systems: State of the Art Report*, Infotech International, Ltd., Maidenhead, England (1978), pp. 63–77. Presents current trends in the data processing marketplace, in order to extrapolate the future. The article considers aspects of people, control, technology, protocols and system architectures.
12. M. E. Deppe and J. P. Fry, "Distributed data bases: A summary of research," *Computer Networks* **1,** No. 2, 130–138 (1976). Discusses the ability for a number of data base management systems to cooperate with one another and thereby share the data. Provides a categorization of relevant research by reviewing current approaches, program and data distribution, file distribution, deadlock, and distributed DBMS control systems. Also provides some conclusions as to the state of the art and additional areas that require research.
13. D. N. Streeter, "Centralization or dispersion of computing facilities," *IBM Systems Journal* **12,** No. 3, 283–301 (1973). Discusses cost factors involved in computing centers that tend to motivate the centralization of computing services. An evaluation and cost-minimization solution is presented. A strategy for linking large regional service centers is proposed and evaluated. The discussion includes the following: tendency toward centralization, advantages of centralization, economies of scale, duplication of data base maintenance, tendency toward decentralization (user-computer communication cost), and centralization-dispersion effects on service quality (queuing model).
14. E. Grapa, *Characterization of a Distributed Data Base System*, Thesis R-76-831, Illinois University, Urbana, IL (October 1976). Presents a multiple copy (distributed) data base system's approach to synchronizing data that includes three models (updates handled decentralized, updates handled at a single site, and a reservation approach). The major flows and extensions of these models are also presented.
15. H. B. Becker, "Securing Distributed Systems: Gaining By *Losing* Control," *Data Management* **16,** No. 3, 24–31 (March 1978). Introduces basic functions of a network and discusses the physical and logical security aspects of each, including surveillance, secure facility, shielding, password cryptography, and secure software.
16. W. W. Chiu, "Performance of file directory systems for data bases in STAR and distributed networks," *1976 National Computer Conference, AFIPS Conference Proceedings* (June 7–10, 1976) **45,** 577–587 (1976). Discusses the trade-offs associated with three classes of file directories for distributed data bases (centralized directory system, local directory systems, and distributed directory system) in cost terms for communication, storage, code translation, as well as such other factors as query rate, update rate, directory size, and directory response time.
17. G. M. Booth, "Distributed information systems," *1976 National Computer Conference, AFIPS Conference Proceedings* (June 7–10, 1976) **45,** 789–795 (1976). Introduces definitions of information network, distributed processing and distributed data base and provides examples of the varieties of data distribution. These examples include horizontally distributed processing, hierarchically distributed processing of partitioned data bases, and replicated data bases. The article also includes a discussion of dynamic load leveling and static load leveling.

18. H. S. Ames, "RDM—A Relational Database Machine," *Proceedings of the 1977 SIGMOD International Conference of Management of Data* (Toronto, Canada, August 1977). Describes an architecture based on a collection of microprocessors, each capable of independently processing a subset of the data base in parallel.

## GENERAL REFERENCES AND NOTES

H. B. Becker, "Network planning—Preparing for distributed DP," *Datacomm User*, 22–24 (January 1977). Discusses a structured and planning philosophy for distributed processing networks that includes topology, volume, processing, response, availability, security, anticipated volume flow, processing load, and response time requirements to derive an optimum and economically justifiable network configuration.

G. M. Booth, "The use of distributed data bases in information networks," *Proceedings of the First International Conference on Computer Communication* (Washington, DC, October 24–26, 1972), pp. 371–376. Introduces theories of distributed data base creation and use in a computer network. Distributed data bases include individual files, company files, grouping files, duplicating files, and splitting files. Also discussed are the association of files with jobs (user control and software) and aspects of updating data and maintaining integrity.

R. G. Casey, "Allocation of copies of a file in an information network," *Proceedings of the Spring Joint Computer Conference* (AFIPS Press, Montvale, NJ) **40,** 617–625 (1972). Discusses the impact of update traffic on the number of copies of a file in a distributed file network and presents a linear cost model to develop a least cost configuration.

A. N. Chandra, *Some Considerations in the Design of Homogeneous Distributed Data Bases*, Research Report RC4125, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 (1972). Presents design considerations for distributed data bases in the areas of compatibility with existing data base products, providing maximum flexibility to installation managers to customize their subsystems, security, and deadlock.

W. W. Chiu and G. Ohlmacher, "Avoiding deadlock in distributed data bases," *Proceedings of the ACM National Symposium* (March 1974), pp. 156–160. Discusses deadlock prevention mechanisms for distributed data base systems using process sets.

*CICS/VS Version 1, Release 4, Systems/Application Design Guide*, SC33-0068; available through the local IBM branch office.

W. T. Hardgrave, "Distributed database technology: An assessment," *Information & Management*, 157–167 (August 1978). Provides an overview of distributed data base technology from the standpoint of commercially used systems. The discussion includes the following: basic terminology, some rules of thumb, back-end and data base nodes, logical data base design (including replication and partitioning), data base machine interface language, and a cursory set of advantages and disadvantages. This article also provides twenty-six references.

*IMS/VS Version 1 System/Application Design Guide*, SH20-9025; available through the local IBM branch office.

M. S. Loomis and G. J. Popek, "A model for data base distribution," *Trends and Application 1976: Computer Networks Symposium Proceedings* (IEEE—National Bureau of Standards, Gaithersburg, MD, November 1976), IEEE, Inc., New York, NY (1976), pp. 162–169. Explores issues involved in distributed data base performance and describes a model that accommodates networks with heterogeneous processing nodes with both private and shared memories. Included are the following: a model of distributed data base, modeling parameters, operational performance evaluation, parallel processing effect of object redundancy and frequency of update, effects on processing costs, and effects on transmission cost.

R. Peebles and E. Manning, "System architecture for distributed data management," *Computer* **11,** No. 1, 40–47 (January 1978). Provides an authoritative over-

view of considerations and status of system architectures for distributed data management systems. Included are discussions of the following architectural approaches: (1) integrated or loosely coupled multiprocessors; (2) homogeneous federations; and (3) heterogeneous federations. Also presented are fundamental problems (data integration, data allocation, data location, control of concurrent access, security and integrity) and software change (CODASYL and operating system interfaces). Examples of interprocess communication are also discussed.

E. F. Severino, "Data bases and distributed processing," *Computer Decisions* **9,** No. 3, 40-43 (March 1977). Reviews the basic attributes of horizontal and hierarchical distributed processing and issues of where the data are located (distributed data base). The data distribution trade-offs are discussed in terms of centralized and distributed data bases. Included are such issues as redundancy, replication, and control.

G. R. Thomas, *Distributed Processing—A Tutorial*, Technical Report TR 01.2063, IBM System Products Division, Endicott, NY 13760. Reviews some of the incentives for distribution and discusses EFTS, industry automation systems, and corporate data communications utility examples for highlighting the benefits of different distributed configurations. Included is a discussion of the functions required for data access, resource sharing, communication, and network management.

K. M. Zemrowski, "Problems of data base use in a distributed data network," *Proceedings of the Fifteenth Annual Technical Symposium* (Washington, DC Chapter of ACM, June 1976), ACM, New York, NY (1976), pp. 51-58. Discusses such data base issues as centralized and decentralized data, users' view of data, data base administration, data base design, integrity, security, efficiency, and implications.

K. Ziegler, Jr., "Distributed data base—where are you?" *Information Processing 77* (IFIP Conference Proceedings, Toronto, Canada, August 1977), North-Holland Publishing Company, Amsterdam, Netherlands (1977), pp. 113-115. Introduces four functionally different categories of distribution (distributed function, distributed systems, distributed processing, and distributed data) and discusses their relationships to a distributed information system.

K. Ziegler, Jr., "Distributed processing (technical overview)," *Proceedings of SHARE 49*, (Washington, DC, August 22, 1977), pp. 1083-1105. Introduces a graphical distribution classification technique for comparing different software distribution implementations. It also discusses several types of distributed data structures (distributed complete DB, split key DB, split structure DB, migrating DB, architectural DB, and redundant DB) and their implications.

*The author is located at the IBM Corporation, Data Processing Division, 1133 Westchester Avenue, White Plains, NY 10604.*

Reprint Order No. G321-5101.