27
3-11-77
25 - NTIS

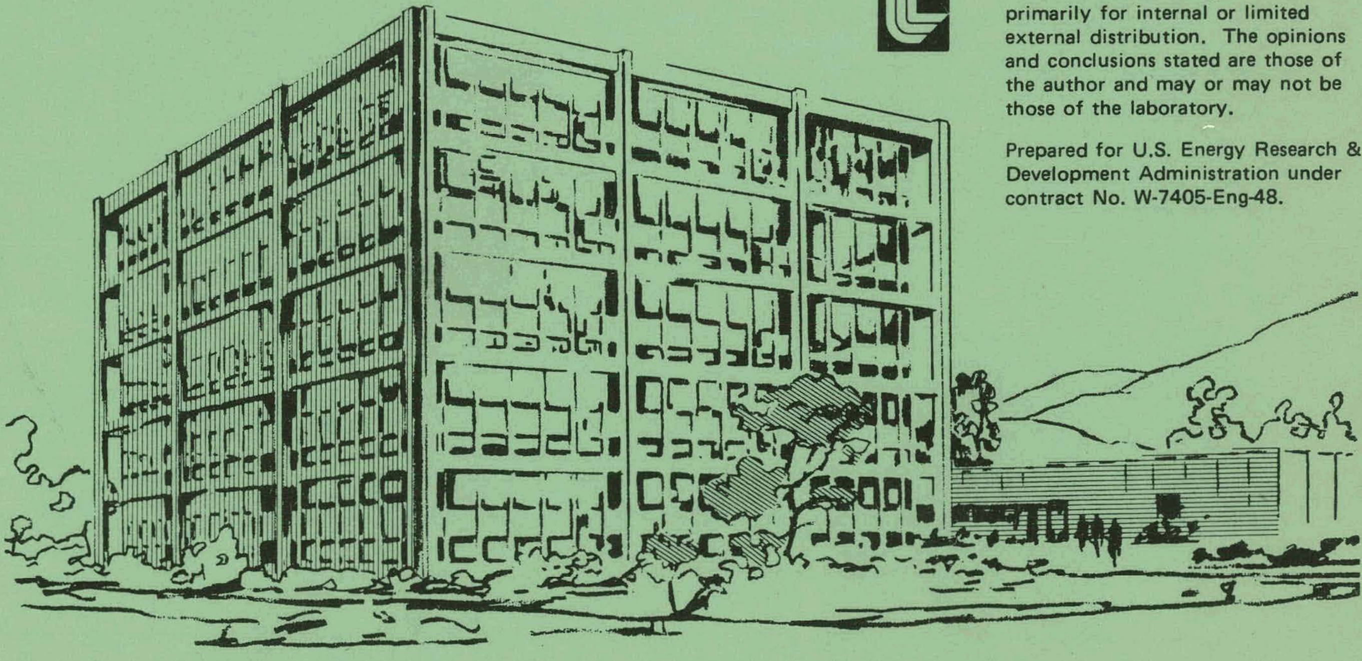
UCID- 17314 Part 2

Lawrence Livermore Laboratory

A MONITOR OF DISTRIBUTED DATA SYSTEMS (MODDS): PART 2, DETAILED FUNCTIONAL SPECIFICATIONS

E. W. Birss, J. E. Donnelley, and J. W. Yeh

November 15, 1976



Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

PAGES __i__ to __ii__

WERE INTENTIONALLY

LEFT BLANK

FOREWORD

The Data Management Research Project at Lawrence Livermore Laboratory (LLL) prepared the functional specifications for a Monitor of Distributed Data Systems (MODDS) under contract [RA] 76-12 with the U.S. Department of Transportation/Transportation Systems Center (DOT/TSC). LLL will submit Part 2, the detailed specifications, to the contract monitor (Alan Kaprelian, Information Division, DOT/TSC, Cambridge, Massachusetts). Part 1 contains a digest of these specifications.

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

CONTENTS

A MONITOR OF DISTRIBUTED DATA SYSTEMS (MODDS): PART 2, DETAILED FUNCTIONAL SPECIFICATIONS

ABSTRACT

Lawrence Livermore Laboratory (LLL) investigated a new approach for improving the use of geographically distributed computer services used by the U.S. Department of Transportation/Transportation Systems Center (DOT/TSC). The interactive use of these distributed services (a major portion of which are data bases) consumes increasing amounts of the DOT user-analyst's time. The new approach increases accountability, selection capability, usability, and control of distributed services use. LLL's approach centers these four elements in MODDS, which is an intermediary between the user-analyst and the distributed computer services.

This Part 2 report presents a detailed description of the current operational problems. It then proposes a solution that addresses each of the problems. LLL performed the work as part of an ongoing contract with the Information Division of DOT/TSC.

INTRODUCTION AND PURPOSE

The U.S. Department of Transportation (DOT) uses vast quantities of transportation-related data to assess current transportation facilities and to plan for future transportation needs. DOT established a Transportation Systems Center (TSC) to provide a statistical and analytical foundation for transportation assessment, planning, and engineering. This center, which collects, assembles, coordinates, and disseminates statistics on current transportation activities, also conducts studies and reports the meaning and significance of its statistics.

DOT/TSC, a relatively new organization, has faced many problems in accomplishing the tasks in its charter. One of its large problems concerns the handling and manipulating of transportation data.

Unlike many other data processing installations that maintain their own data exclusively inhouse, DOT/TSC must make use of data collected by other agencies and organizations such as the Federal Aviation Administration, Civil Aeronautics Board, Federal Highway Administration, National Bureau of Economic Research, etc. The requirement to use these data resources, which are distributed throughout the nation, creates difficulties in use and in accounting for the use of distributed data.

This report intends to identify the problems of use and of accounting for the use of distributed data. A solution using a Monitor of Distributed Data Systems (MODDS) will be proposed and the MODDS function will be described.

BACKGROUND AND USE OF DISTRIBUTED COMPUTER SERVICES

Distributed Data: A Historical Perspective

From the time of the first computer, its use has had to adapt to the rapid evolution of its technology. Early use centered around the organization's computer. Usually the organization had only one, and all computer work was accomplished on that machine. As computers became more economical, organizations could afford several, and typically, certain applications were developed to run on specific machines. The minicomputer boom has further accelerated this evolution. Many organizations have acquired a large array of minicomputers to address specific applications.

Other than growing computer power within an organization, competition from computer-service vendors began to increase. As computer technology developed, commercial vendors were able to decrease their costs to such a degree that it was frequently more cost-effective to go outside the organization for certain computer services.

Over the years, computer services diversified as well. Time sharing became one of the first types of service available. Time-sharing service installations proliferated, and competition forced each one to offer services different from any of the others. Through this process of evolution, vendors of computer services began to offer specialized programs and data bases.

Emerging Data Services

Various companies offer a repertoire of specialized data bases along with program tools to analyze them. These data services often provide data in a very cost-effective manner. Data is usually inexpensive because the vendor can amortize costs over many customers. By availing itself of various data services, an organization may use better data at a fraction of the cost of doing its own collecting, installing, maintaining, and processing of data.

Some of the emerging data services used by the Department of Transportation follow:

- Applied Programming Languages, Inc. (APL). APL supplies the Civil Aeronautics Board (CAB) Form 41 data base by means of an online computer system called AAIMS.
- Data Resources, Inc. (DRI). DRI markets econometric models of national and regional scope. It also supports detailed time-series analyses of such subjects as the Consumer Price Index, population statistics, wage rates, and energy consumption.
- National Bureau of Economic Research (NBER). NBER provides services very similar to those of the DRI service. Its TROLL package supports statistical analyses of various econometric data.
- Computer Science Corporation (CSC). CSC (INFODATA) is coordinating with the Federal Aviation Administration (FAA) to establish online requirements for data distribution of vital statistics to FAA/DOT.
- Boeing Computer System (BCS). BCS provides services to DOT to process data gathered from the air- and ground-transportation industries.
- Transportation Systems Center. TSC provides data to DOT. Its systems 1022 and DBMS-10 provide data typically not available to the DOT community from commercial sources because of restrictions on the distribution and use of proprietary data.

Commercial Computer Services

Besides these data services, DOT/TSC purchases time-sharing and batch services from such vendors as

First Data Corp., Tymshare, USS Engineers & Consultants, General Electric Co., Bowne Time Sharing, United Computing Systems, Control Data Corporation, Informatics Inc., Computer Network Corp., Grumman Data Systems, and McDonnell-Douglas Automation Co. These services are offered on a variety of hardware from such suppliers as Digital Equipment Corp., Honeywell, Control Data Corp., and International Business Machines.

With such hardware comes an even greater variety of software systems. The software operating systems for the same hardware sometimes differ among vendors. Methods for using the operating systems, files, and data bases also vary across the vendor services.

Coping with these vendor services requires the DOT analyst to know how to

1. establish a connection to a computer,

2. log on to each system,
3. be able to process stipulated tasks,
4. know how to recover from an unexpected error or system message,
5. log off the system, and
6. disconnect from it.

Frequently, the processing tasks mentioned in item 3 above include:

- creating files,
- editing files,
- listing files,
- writing, compiling, and executing data base application programs.

Current Use of Computer Services within DOT/TSC

To use these various data and time-sharing services, the DOT/TSC users and analysts must be versatile. Basically, the users and analysts fit into two categories: those that directly use the computer and those that do not.

The analysts that do not directly use the computer convey a description of their problem and a method of solution to a computer programmer. Use of computers at DOT/TSC, however, is not as centralized as this discussion might imply. Members of the DOT/TSC user-analyst community use a variety of data and time-sharing services as mentioned previously (Fig. 1).

The user-analyst community accesses these services primarily with dial-up telecommunications and terminals equipped with acoustic couplers, although remote job entry terminals are sometimes used. Since the majority of outside computer use is with interactive terminals, this report focuses on problems that occur with interactive use of these remote computers.

A typical terminal session involving two remote computers might be as follows:

- Determine what source data is required and where a computerized version of the data exists.
- Dial up computer "A", which has the required data.
- Log on to the computer.
- Locate the data and analysis files to be used.
- Extract data from one of the data bases on the "A" computer.
- Put the resultant data on a tape cassette.
- Log off the "A" computer.
- Hang up the phone.
- Rewind the tape cassette.
- Dial up computer "B".

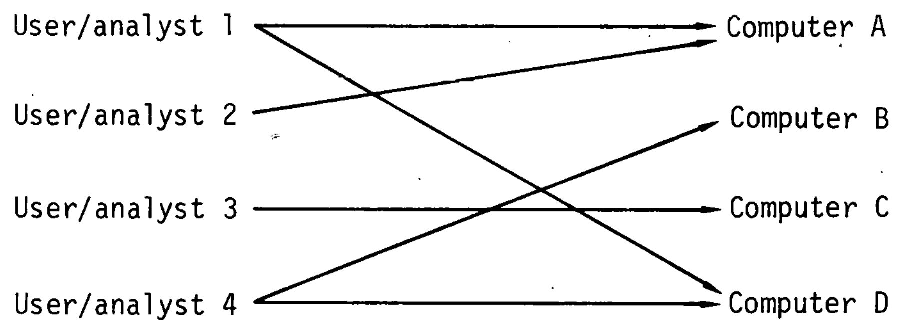


Fig. 1. Typical use of distributed computing services.

- Dial up computer "B".
- Log on to computer "B".
- Construct a file on computer "B" from the tape cassette.
- Reformat data to make it compatible with the input format for the plotting and analysis routines.
- Locate plotting and analysis packages.
- Analyze and plot data.
- Log off computer "B".
- Hang up the phone.

The simple task of extracting data from a data base and analyzing and plotting the results requires at least the above 17 steps using the current mode of operation. Some of these steps are potentially involved and cumbersome. For example, copying the data to a cassette or constructing a file from a cassette is a complex procedure in some computer systems, while it is very simple in others. As can be seen from the above scenario, the current mechanisms are quite cumbersome. They raise problems for computer resource management and progress reporting to management. The next section discusses these problems in more detail.

PROBLEMS IN USING DISTRIBUTED SYSTEMS

Accountability

Besides the use problems associated with distributed resources, accounting for the use of resources is difficult. When all computer resources are local, accounting for the resources and accountability for such use is easily maintained with programs and administrative support from the operations staff of the computer center. However, when resources are procured from outside vendors, the operations staff can no longer generate or accumulate such statistics.

Outside vendors, like an in-house computer facility, maintain statistics on customer computer use. Management generally receives bills for these outside services at month's end. The billed services are from telecommunications vendors, data systems vendors, and time-sharing vendors. The charges, itemized by each user, often break down into such categories as processing, disk storage, and connect time.

Several problems exist with this type of billing arrangement. First, the managers no longer can ask intermediate questions such as how much have we spent so far? Second, each vendor has different charging schemes that are nonuniform. Third, there is no check on the vendor's claim as to the amount of money spent. Finally, bills are only itemized by user identifier. As such, they are very coarse accounting statistics and do not include accounting for all resources used.

Determining What Resources Exist; Sharing Resources

With a plethora of services available today, the DOT/TSC user-analyst is, to a large extent, unaware of which vendor has the best-suited software or hardware for the task. Documentation of these resources is often incomplete, and the analyst may have to rely on word-of-mouth instruction from others. The analyst could greatly benefit from a catalog of available resources and a rating scheme for these services to make an intelligent choice of facilities.

Users are, to a great degree, unaware of the overall facilities available to them. Thus, users tend to duplicate similar work of others, and there is minimal sharing of programs, data bases, or results.

Ability to Use

Since DOT/TSC uses various data services to accomplish detailed analyses of transportation-related data, the efficient use of the various services is less than if the central computer at DOT/TSC was used. The reason for this situation stems from the distributed nature of computer resources.

Armed only with a terminal and a telephone, the analyst must use the various resources from a diverse set of data vendors. To use these data resources, the analyst must remember

1. the telephone numbers of the system,
2. telephone and computer-network access mechanisms,
3. terminal-identification protocols,
4. log-in procedures, including identifier, password, and account number,
5. executive-language commands that allow the analyst to compile and execute programs, edit files, access data bases, etc., and
6. what to do if something goes wrong.

All these procedures are usually learned through experience, and for tasks like 6 above, the learning procedure continues as systems and software evolve.

The distributed nature of data resources complicates the learning procedure because all the idiosyncrasies for each system must be learned. If an analyst is to use six systems, he must learn six each of the following: telephone numbers, network-access mechanisms, terminal-identification protocols, log-in procedures, executive languages, error-recovery procedures, etc.

The time needed to gain familiarity with six dissimilar systems is at least six times as great as learning any one system. Not only is the learning effort time consuming, the analyst must also be able to keep instructions for each system separate and must readily switch among systems without confusion.

The problem is not learning all these aspects of a system; it's learning and maintaining a level of expertise for several or all these systems. Very few users, if any, can be expected to be intimately familiar with all fourteen systems used by DOT/TSC today.

To date, when coping with the intricacies required to access and use the various distributed resources, analysts have either

- concentrated on only a few systems,
- channeled most requests for computer analysis through computer specialists, or
- worked without computers.

Concentrating on just a few systems allows the analyst to maintain a reasonable familiarity with procedures to process data from any one vendor of data resources. Thus, an analyst is effectively limited to two or three systems, which eliminates the advantages offered by the diversity of systems and the communications facilities.

Channeling requests for analysis through computer specialists allows noninteractive access to many systems, but the analyst is far removed from the actual analysis and does not use its full potential. Communicating the method of analysis to a computer specialist is difficult and time consuming for both computer specialist and analyst. Another disadvantage of this approach is the added need for computer specialists to remember and perform computer-related work.

Because of the present cumbersome access to distributed resources, several difficulties arise:

- Programmer/analysts are inefficiently used because the available system is cumbersome.

- Resources are not fully exploited because of the lack of awareness of the most appropriate facility and insufficiently powerful data-handling facilities.
- Effort is duplicated throughout DOT/TSC because of poor communication among users and the difficulty of sharing programs and data.
- Evaluating the significance of transportation statistics, as mandated by the DOT/TSC charter, is difficult. Far-ranging policy decisions necessarily require data gathered by several agencies. Aggregating this data into a form that can be assimilated and then evaluated is tedious when data from several computer centers is involved.

Control

Current operational mechanisms permit any user with a valid identifier, password, and account number to access a remote system. The organization relies on the vendor of the computer services to invalidate or validate users and passwords. Control of resources at the vendor's computer also relies on vendor software to permit and restrict access to files, data bases, and programs.

Relying solely on vendors to provide all the control is not desirable. Vendors do not have ultimate responsibility for the misuse or illicit dissemination of data. A vendor's primary responsibility is to supply service. Thus, it is desirable to establish high-level control over data and analysis programs within DOT/TSC.

REQUIRED IMPROVEMENTS IN USING DISTRIBUTED SYSTEMS

The following four major areas require improvement:

- Accountability.
- Determination of availability and sharing of resources.
- Ease-of-use of distributed data systems.
- Control over the use of distributed resources.

Accountability

Accountability; i.e., keeping track of allocated funds and all resources (including manpower, hardware, and software) and who has or has had custody, is an important management function. The system should provide early notice whenever resources are misused, underused, or overused. The accountability mechanism should be designed to identify the cause of abuse and pinpoint the responsibility. With this information, recovery can be facilitated and unauthorized penetration of the system detected early enough to prevent major loss.

To achieve this accountability, the following four points must be enforced.

Accounting for Physical Resources — This portion of the accounting mechanism keeps track of the physical resources used, such as central processing unit (CPU) time, peripheral storage, communication channels, and terminal activities.

This detailed information has three uses:

- One can determine when to increase or decrease telecommunications facilities. Communications vendors provide a spectrum of communications resources; each service is specialized, each has its tradeoffs. Selection of the optimum service requires detailed use statistics. Sophisticated facilities can be included to automatically and dynamically configure the available communications facilities to minimize costs.
- One can determine whether to locate data locally or remotely. Data accessed frequently should probably be located at the local computer installation; data accessed infrequently should be located at remote sites.
- One can plan for future needs. Trends in computer use can, to a certain extent, be predicted based on use statistics. Such statistics can be helpful in planning for future communications needs, data needs, software needs, etc.

Accounting for Human Resources — This portion accounts for the human effort in task execution. Data can be collected on connect time, on frequency of access to resources, and on file use to give an accurate picture of effort. The accumulation of costs by an individual analyst and by project not only provides a measurement for individual performance but also offers a way to distribute costs directly to the projects.

Accounting for Resource Access — Access to all physical computer-related resources should be a matter of record, and management should be able to identify custody and access at all times.

Reporting Regularly — Regular reporting should be provided to assure the identification of irregularities. Dynamic reporting should automatically generate exception reports for management (e.g., production of a report after $1000 has been spent).

Determining Resource Sharing; Online Assistance

Technical feasibility of resource sharing has been demonstrated by time-sharing computer systems and geographically distributed computer networks. There are a number of reasons that make resource sharing beneficial:

- Sharing eliminates redundancy. Theoretically, sharing will allow a data base and application programs to be developed and stored only once in the system.
- Sharing aggregates specialized demands into specialized services. Resources are better used because they address specific needs.
- Sharing will promote access to a wider range of resources. Since the unique resources and specialized packages become profitable through sharing, a wide range of resources are available.

For broad sharing of resources, compatibility, common access, and online assistance should be included.

Compatibility — With various resources available on different computer systems, a proliferation of possibly confusing or conflicting uses of the systems and their access methods may arise. This could inhibit the sharing of all resources. Full use of all available resources requires compatible resources. This includes command-language, communication-protocol, system-message and common-terminology, file-structure, and programming-language compatibility. Command languages must be logically compatible so that a user can work with a number of systems simultaneously. Communication protocols should be compatible to make access and transmission transparent to users. The format of messages from various systems and terminologies used in the messages should be as uniform as possible to express the messages unambiguously. Data bases and programs are the most important resources to be shared; therefore, compatibility among data bases and programs is the most important component of compatibility.

Increased use of high-level programming languages and data base management systems has made the compatibility problem among data bases and programs a bit more manageable. Research on high-level language translation, data translation, and migration has received considerable attention in recent years. These research results and efforts should apply to the MODDS efforts and provide a foundation for further compatibility.

Common Access Method — A simple and common access method should be provided to assist a user in accessing and sharing data bases and programs. To achieve this goal, it would be desirable to have a "single-point contact," where a user may make arrangements for services, obtain the latest information on different services, or even obtain advice and consultation on using the service more effectively. This "single-point contact" would provide easy access mechanisms. Through the use of the central contact, with all the resources being accessed through it, this method assists users in accessing and sharing data bases.

On-line Assistance — Once a user has "logged on" to the system, the operation should be continuous. This requires online user assistance. Typical assistance may be a message about system status and availability of services, a short tutorial regarding system operation and applications, a library of online documentation, an online news bulletin, and a list of consultants who provide assistance.

One of the major components that the online assistance facility must include is a directory of available resources. This online directory should be constructed from the results of the bibliographic-data file project. The online directory would assist user/analysts at DOT/TSC in locating data bases and analysis programs germane to their tasks.

Ability to Use

Ease of Use — The inability to use computer resources in a distributed environment is one of the main hindrances of distributed computing. Whereas economic benefits are often reaped in purchasing services from outside vendors, this benefit is, to a certain extent, artificial if the complexity of the distributed resources renders them cumbersome to use. The cost in terms of "people time" increases dramatically, and in some cases, easily overshadows any cost decrease realized from the distributed architecture.

A more usable system requires an increase in user efficiency and a unified view of distributed resources. For this view, the user should request a resource such as a program or a data base. The user need not know where the resource exists and should not have to perform any connection to the computer that has the resource. This view would look to the user as if there was a central computer with all the resources available but would actually be an architecture where the resources are distributed among dissimilar computers.

To provide the unified view of resources distributed across dissimilar computers requires many automatic mechanisms. Some of the most important mechanisms must dynamically handle the selection and placement of resources among the computers for efficient response. Mechanisms must also be supplied for reliable access.

Efficiency and Reliability — Efficiency and reliability are two main concerns for all computer systems. These problems range from the individual component performance to overall system throughput. Interconnecting dissimilar computer systems to a network further complicates efficiency and reliability. For the sake of this discussion, only the following three subproblems pertinent to distributed data bases will be discussed.

Directory Organization — A directory catalogs resources available to all users and assists users in locating resources they require. There are a number of ways to organize a directory that may be centralized at a site or distributed among many sites (decentralized). [There is no master directory, and every site has its own local directory.] The tradeoff among different organizations greatly depends on network topology and communications, storage, and translation costs (while the actual locations and topologies are transparent to the user).

Data Base and Program Allocation — When a given data base or program is required by a number of users, one copy may be stored at a site in the network for access by all users. The overall operating cost related to data base or program use consists of transmission, translation, and storage costs. The major problem is the automatic allocation of copies of a data base or program to sites in a network so that overall operating cost is minimized while turnaround time remains within acceptable bounds. These procedures must be accomplished heuristically and automatically, since the user is unaware that the resources are distributed.

Deadlock and Recovery — In a distributed data base environment, which must provide simultaneous access to all data bases, the problems of access synchronization and deadlock are of special importance. Access to data bases must be synchronized carefully to prevent one process from disturbing the actions of another. A common access synchronization solution is a lock-unlock mechanism. An important consideration for this mechanism will be the choice of appropriately sized lockable objects to maximize concurrency and minimize overhead.

The other major problem is deadlock, which occurs whenever two or more processes vie for the same resources and reach an impasse. Deadlock has been thoroughly studied in recent years, and a number of prevention and detection algorithms have been published. Recovery is required if a deadlock is detected or a software or hardware crash occurs. "Rollback and restart" is the most common solution. However, it is too time-consuming and may not be acceptable in a distributed environment, since it may involve a chain of rollbacks and restarts throughout the network.

Control

In a truly resource-sharing environment, although data bases and programs are common nonredundant assets to the community, users still have the preroga-tive to restrict their files and programs only to authorized users. For example, a personal medical or psychological record needs restricted access only by authorized physicians. A system with controlled access is required for a resource-sharing environment to assure that only authorized users can gain access to specific data bases and programs.

To develop such a secured system, the following three functions must be provided.

Access Controls — Access control protects the anonymity of private information and regulates the access to shareable resources. Access control has four criteria:

- The "objects" to be protected must be specified. They may be a logical record of a data base, program, event, or physical memory cell.
- The "rights of access" to objects must be defined. Some typical access rights for files are: read only, read/write, write only, or execution only.
- Access rights among users must be regulated so they can gain and grant rights from and to others. A number of models have been developed or proposed in recent years. Among them, the capability list (or C-List) model is one of the most prominent.
- A mechanism must be created to enforce access by authorization only. Note that the "enforcer" itself must be a secured system. It must be a small kernel, which is tamper-proof and certifiable.

Data Cryptography — Access control protects access to containers but not to their contents. As a consequence, everything inside the container is subject to exposure. For example, if a container is lost accidentally, its contents may be exposed. For data security in stored media, cryptographic techniques have been widely used. The encoding/decoding and scrambling/unscrambling of data have been used to protect private, stored information. Techniques for high-level cryptography of data bases and file structures have been developed and should be included in a resource-sharing environment.

Secure Transmission — In a distributed environment, information must be transmitted between computers; therefore, it may be exposed during transmission. In this area, encryption has been extremely successful in preserving the security of private message traffic. The most obvious way to provide transmission encryption is to install an encryption/decryption device on each end of a transmission line. As a result, communication processors must be equipped with some of these cryptographic capabilities.

AN APPROACH TO SOLVING DISTRIBUTED DATA PROBLEMS

Introduction to MODDS

A simple central monitoring facility, called MODDS, would offer accountability, resource sharing, and a usable secure facility that provides transparent access to distributed data systems. MODDS would simplify the use of distributed data while monitoring the access and use of computerized data systems.

The basic concept for MODDS is that of a central hardware/software entity situated as an intermediary between the user-analyst's terminal and the remote computers (Fig. 2).

The central architecture of MODDS permits accountability, since all requests for resources are channeled through it and can be noted and logged for subsequent analysis. To log resource requests, MODDS must distinguish between a resource request and data input to a program. The knowledge base used to make this determination is a directory of resources available by means of MODDS. This online directory will be derived from the bibliographic data-file project at DOT/TSC.

Besides permitting recognition of resource requests for logging accountability data, the directory provides another useful user service. By intelligently querying this directory, a user can locate a previously unknown resource (either a program or data base). The directory will reduce duplication of programs and data bases because its ability to locate data and use established programs permits increased efficiency for support programmers and casual users.

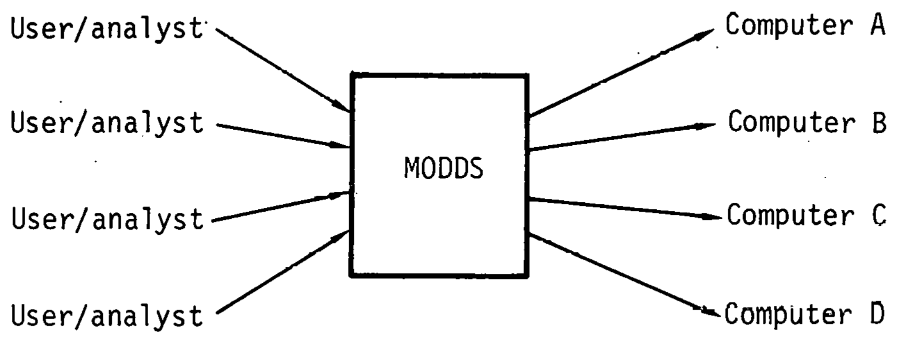


Fig. 2. MODDS as intermediary between user/analyst and remote computer.

By aiding the user in locating already existing resources, the directory promotes resource sharing. Such sharing decreases the cost of providing resources by eliminating redundancies. Also, it promotes the use of a wider range of available resources and aggregates specialized demands into specialized services.

Once the user connects directly to MODDS, it assumes responsibility for connecting the user-analyst's terminal to the various computers maintaining communications between terminals and computers.

Although the centralized architecture of MODDS is appropriate for accountability and resource sharing, this may not necessarily mean that MODDS could simplify the use of distributed resources without more functional capabilities. After all, an added level of computers at the peak of the data-systems pyramid could conceivably complicate the user's view of his problem.

However, MODDS has an intelligent software component which simplifies the user's view of distributed resources. The problem with a distributed view is that the resources are distributed; if they were located centrally, no problem would exist. Thus, the software component of MODDS makes distributed resources appear local.

Figures 3 and 4 diagram the user/analyst views of programs and data bases without and with MODDS. Figure 3 depicts the user/analyst view without MODDS; the user/analyst sees individual resources within an array of computer systems. Figure 4 contrasts this current view with one that would be presented with MODDS (where the user/analyst's view is that of a "virtual computer"). The virtual computer is an aggregation of available computers; all resources available to the user/analyst on any of the computers is available on the virtual computer.

Thus, MODDS, by aggregating all the computers into a "virtual computer," lets the user/analyst see a unified, consistent view of all resources (programs and data). This facilitates the use of distributed resources by making them appear local.

MODDS Operation

The use of MODDS, while simple, is difficult to describe; thus a scenario (Table 1) outlines its operation. The left half of Table 1 shows what the user does, the right half describes the actions of MODDS.

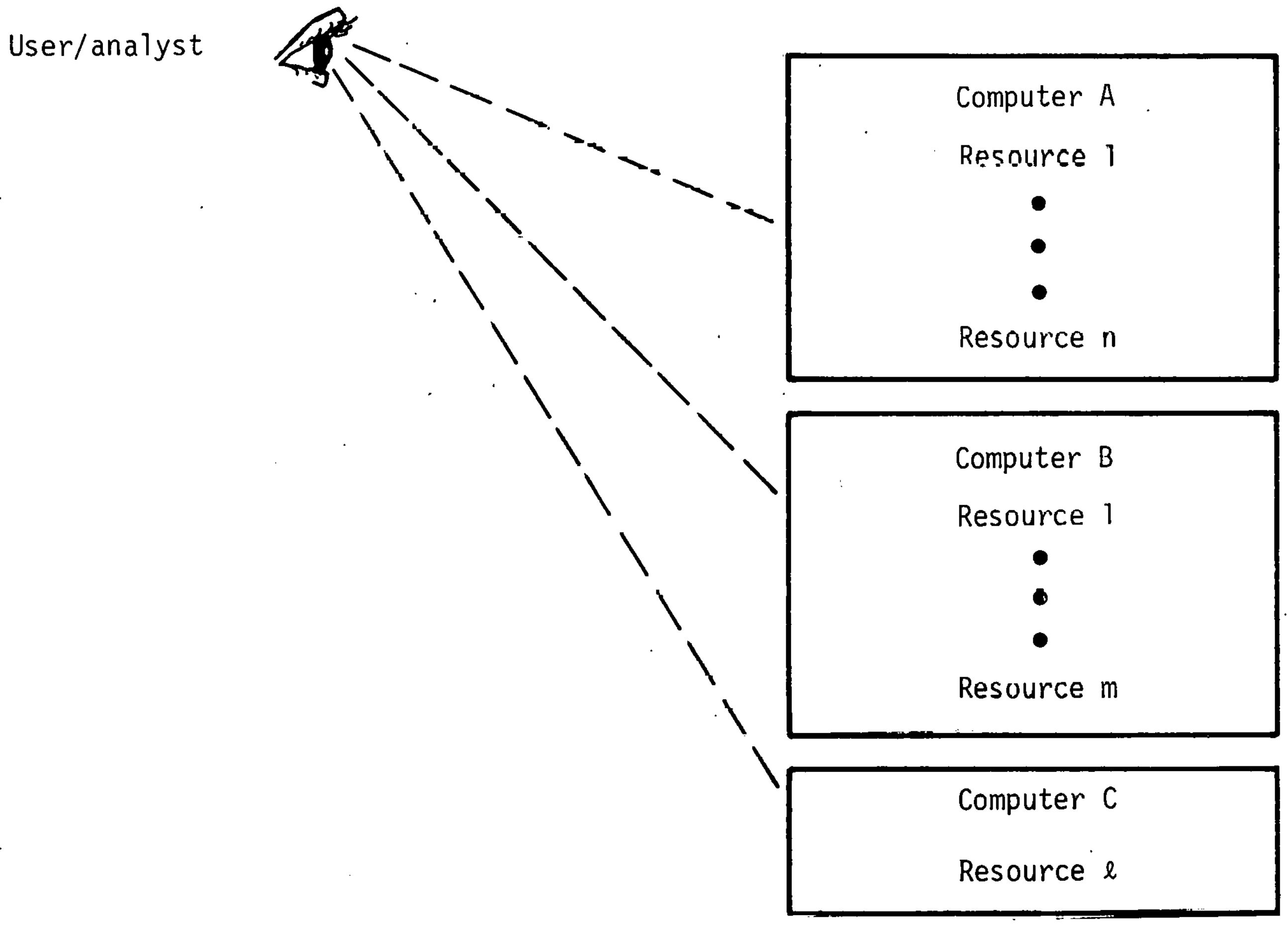


Fig. 3. User's view of computer resources without MODDS.

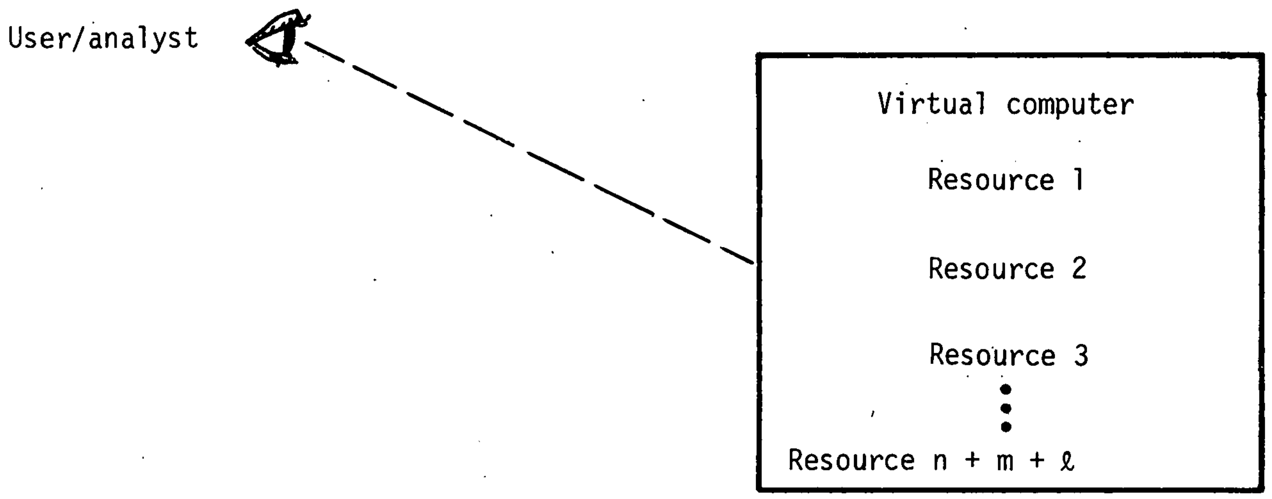
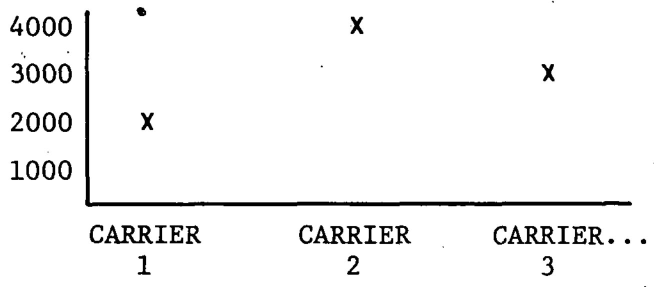


Fig. 4. User's view of computer resources with MODDS.

Table 1. User/MODDS activities for typical transportation problem.

USER INPUT COMMAND	*MODDS actions;* COMPUTER'S TYPED RESPONSES
1. CONNECT TO MODDS EITHER BY TYPING A SPECIAL CHARACTER OR DIALING-UP	
2. LOG IN USER ACCT. # PASSWORD	*Verify user's identification, password, account #.*
3. QUERY DIRECTORY-OF-RESOURCES	*Locate directory, establish communication with required host computer, log the user into the host, initiate the query package.*
4. FIND ALL DATA BASES WITH AIRLINE DATA	2 DATA BASES ARE FOUND [Note: Assume they reside at two different locations. There is no requirement to inform the user of the physical location since he is only interested in the logical content of the data.]
5. DISPLAY DATA BASE NAME, ABSTRACT	KEY: 1 NAME: CAB41 ABSTRACT: CONTAINS BASIC DATA COLLECTED BY CAB KEY: 2 NAME: CAB586 ABSTRACT: CONTAINS ORIGIN-DESTINATION DATA FOR COMMERCIAL AIRLINE PASSENGER TRAFFIC INCLUDING ENPLANEMENTS AND DEPLANEMENTS.
6. DISPLAY FIELD-DATA FOR CAB 586	FIELDS ARE: BASE REFERENCE CODE CHARACTER NUMERIC LOCATION CODE NUMERIC ALPHA AIRPORT CODE CHARACTER
7. QUIT	*Disconnect from host that has Query system.*

Table 1 (cont.)

USER INPUT COMMAND	*MODDS actions;* COMPUTER'S TYPED RESPONSES
8. CAB586	*Locate CAB586. Connect to host, log user onto host, initiate DBMS.*
9. EXTRACT REVENUE-PASSENGERS FOR ORIGIN = LAX AND DESTINATION = ORD OUTPUT = FYLE	
10. QUIT	*Disconnect from host.*
11. PLOT REVENUE-PASSENGERS VERSUS CARRIER FROM FYLE	*Locate Plot Rooting.* *Connect to host.* *Log user into host.* *Ship FYLE to host.* *Initiate plot package.* 
12. QUIT	*Disconnect from host.*
13. LOGOUT	*Note logout time.* *Disconnect user.*

As can be seen by the scenario, the user is unaware that the resources viewed were from distributed sources. One can also see that MODDS performs many functions in response to a single user's command. The scenario, while quite simplified, did not include the activities involving communications media selection, data translation, query translation, locking, unlocking, access control, and accountability mechanisms. They are an adjunct to the user's request. The user normally is not interested in the specific activities performed automatically by MODDS.

Interface Considerations

MODDS interfaces with users and computers on opposite sides. User interfaces exclusively consist of terminal-oriented facilities physically visible to the user. To address user needs, a computer specialist is needed to maintain MODDS and support its interfaces.

User Interface — The computer specialist maintains and keeps the MODDS knowledge-base current. His function is similar to that of a data base administrator; he validates and invalidates user identifiers, determines what resources should be used by whom, etc. His function actually includes some data base administration functions (since the resources being managed include data bases). The computer specialist must also keep the directory of resources current so users have all the advantages of new software as it is developed.

Computer Interface — Since many of the functions performed by MODDS are made possible through a unified view of resources, part of the system must provide the conversion from real system to uniform view. For certain resources, the conversion must take place in the remote computer. The computer specialist also maintains the existing remote computer interface programs and adds new interface programs as more facilities are made available to the user community.

MODDS Functions

This section describes the most prominent functions in MODDS and gives an overview of the module that implements each function. The major modules described include the Kernel plus the hardware and high-level resource handlers.

Kernel — This is the heart of MODDS. It supplies the primitives from which all other MODDS functions are built. Specifically, it does the following:

- It maintains the core-resident portions of the tables describing the state and domain of each process in the system. The domain information maintained by the Kernel allows enforced separation of responsibility between the other modules in MODDS. The importance of separation of responsibility has long been recognized in administrative circles but has just recently begun to be regarded as equally important for reliability and security in computer systems.

- It provides an interprocess-communication mechanism based on a generalized model of a computer resource. The flexibility of this mechanism is vital for the extension of MODDS to distributed resources. This needed flexibility is a major constraint on the Kernel design. See Appendix A for more details.
- It makes the hardware resources of the computer only available to those processes responsible for their proper use. On a PDP-11 prototype system, this includes the CPU, memory, and various input/output devices. The next paragraph describes the operation of the major hardware resource handlers.

Hardware Resource Handlers — These modules handle all devices in the system. Specifically, they reserve hardware resources such as the CPU, memory, disk, terminals, telephones, and value-added networks.

The CPU Handler insures that the right processes are run at the right time. In the prototype MODDS, this function has two roles:

- It creates a priority-interrupt system appropriate to the handling processes.
- It schedules an appropriate process when there is more than one requesting the CPU at the currently active hardware priority. Section II of Appendix B describes the algorithm currently used for this classic scheduling operation.

The Two Memory Handlers are responsible for the system's direct access memory: the resource and information pagers.

- The Resource Pager handles the portion of the memory allocated for the Kernel's resource tables. When a request for a new resource supported by the Kernel tables overflows the tables, the resource paper does the following:
  1. It saves the information needed to reconstruct the least recently used resource in the online disk. A trusted data storage device must be used. If this information were altered, the reconstruction of the resource, when it was needed as in step 4 below, could produce erroneous results.

2. It destroys the resident version of the saved resource, freeing some Kernel table space. The resource is destroyed so requests for the no-longer-resident resource will be detected by the resource pagers.
3. It allows the original resource request to proceed.
4. If a request is made for a resource no longer resident, the resource pager is nullified so it can reconstruct the resource.

- The Information Pager performs what essentially amounts to the classic least-recently-used paging algorithm for allocating the system's direct access memory to processes. One very important difference, however, is mandated by MODDS' design to allow use of distributed data systems. Proper paging of distributed data requires cooperation with the handler (which might be a remote system) so a remote update does not invalidate the data in MODDS' direct access memory. For this reason, MODDS' information pager must perform the appropriate write and read/write locks on the source of the paged information and must request updated copies of the data when needed.

The Disk Handler subdivides online disk storage into resource units (files) that can be individually allocated to processes. To insure that processes (which may be remote) can properly manage local copies of the data, the disk handler also services write and read-write lock requests and performs the appropriate notification when necessary.

The Terminal Handler supports individual terminal resources by properly handling various terminal and multiplexor devices. It also supports various option negotiations to insure a match between a requesting program and the terminal on issues such as echoing, character set, graphic or editing capabilities, page size, etc. For more details on these mechanisms, see the Interactive Terminal Resources (ITR) section in Appendix C.

The Telephone Handler allows processes to reserve telephones. These processes are given complete freedom to manipulate the telephone with operations like dial a digit, send a character, hang up, etc.

Value Added Network (VAN) Handlers allow a host to communicate with many others simply by naming the remote host to a local VAN node. This fact makes communication with all systems on VAN appear the same to MODDS, greatly reducing the number of communications protocols MODDS must handle. The single protocol that MODDS uses to communicate with VAN tends to be somewhat sophisticated to insure that communications are as flexible and reliable as possible. It includes error control, flow control, multiplexing features, etc. to insure efficiency and integrity of data transmitted and received.

Some network protocols that may be served by MODDS are ARPANET, DECNET, and X25 (a proposed international standard). Added protocols such as those described in Appendix D can also be supported.

High-Level Resource Handlers — These handlers support resources not closely tied to the hardware resources of the system. They more directly support the needs of DOT/TSC users. Since a user only can interact with MODDS through a terminal device, most of these resources (except the ITRs) are largely invisible to the user. High-level resource handlers specifically discussed include the Distributed-Capability Computing System (DCCS), the log handler, the user handler, the account handler, the Data Storage Resource (DSR) handler, and the various ITR handlers.

The DCCS protocol is the primary resource-sharing mechanism in MODDS. It represents a significant breakthrough in computer-resource sharing. For the first time, a single resource-sharing mechanism can share any type of computer resource supported by a variety of systems and not be changed as new types of resources are added. Appendix A more fully describes DCCS.

Basically, the support mechanism has two parts:

- DCCS user — This allows MODDS to use any standardized resource in the computer systems MODDS can communicate with.
- DCCS Server — This allows controlled access by remote systems to any resources existing in MODDS. The need to allow a uniform DCCS source implementation was a major design constraint on Kernel and other basic support functions of MODDS.

The Log Handler services requests from various other handlers to log services they have performed. Each entry in the log generally contains information like the time the service was requested, the requesting user, the handler servicing the request, the type of request, etc. Handlers can make log entries whenever they perform a service for which statistics are desired.

The User Handler identifies users for the other handlers so they can identify log entries by user name.

The Account Handler maintains MODDS currency that can be used to determine cost of services and charge for various functions. Each handler charges for its services by transferring currency between accounts. MODDS currency will be kept at parity with real dollars whenever possible.

The DSR handler presents a uniform view of both local and remote data bases to requesting MODDS processes. This portion of the MODDS development is the most research oriented. Several DSR servers will be implemented as the required translation mechanisms are developed. More than one distinct form of DSR may result: one representing tape-oriented data bases, one representing relational data bases, and possibly others. Two implementation mechanisms will be attempted:

- Query Level — This approach starts much like the connection procedure described in the ITR section below. MODDS connects a query language front end to the data base and interacts with the data base as if it were a terminal. This mechanism's advantage is that no new software need be written for the host system (although it lacks effective data transmission, error control, and synchronization).
- Subroutine Level — This approach requires some software to be written for the host system on which the data base resides. The software interacts with the data base on one side at the subroutine or system call level and interacts on the other side with MODDS over the communications facilities.

This approach has the advantage of allowing flexible data transmission, error control, and synchronization to be handled directly at the data base host.

Further discussion of various DSR implementation issues can be found in the DSR section of Appendix C.

ITRs compose the basic interactive language support within MODDS. An ITR consists of a socket into which a user may figuratively plug his terminal to interact with a program that will respond to his commands. Most of the ITRs available within MODDS actually connect the user to a remote system that supports the desired command language. For each of these remote ITRs, MODDS must negotiate terminal parameters on behalf of the remote program, perform the connection to the remote site, log in for the user, start up the program that performs the desired language service, and translate between the data format expected by the terminal and that expected by the remote program. For more details on the remote ITR mechanisms, see Appendix C.

There are a few ITRs implemented directly on MODDS itself because they either cannot or are not implemented remotely:

- Identifier — The main function of this ITR is to identify users at terminals and to initiate services for them when they are recognized.
- EXEC — Until more specialized command processes are needed, a single command processor called EXEC will be used. EXEC allows the user to plug his terminal into the various ITRs available in his private directory and to manipulate multiple ITRs at will.
- File Transfer Program (FTP) — This allows explicit transfer of simple data files to and from MODDS. It models and extends the FTP available on hosts of the ARPA network. Such an explicit user-visible transfer mechanism is, unfortunately, currently necessary because most existing operating systems do not allow their programs to transparently reference remote files in the same way as local files. This problem does not exist with terminals where a transparent remote-access facility is available. Appendix C further discusses the tradeoff between the FTP and the DSRs.
- DSR Query Language — This will supply MODDS terminal users with a highly interactive means for interrogating DSRs available within MODDS.
- Other Local Utility ITRs — There are many other local ITRs for performing utility functions such as directory listing, file copying, system status listing, explicit telephone calling, etc.

General Performance Specifications

MODDS is being designed to meet the anticipated needs of DOT/TSC for access to distributed data and computational resources. To do so, the following will have to happen:

- It must support 50 to 100 interactive terminals of various types capable of responding to requests from between 10 and 20 concurrently active users within one second.
- It must support the full bandwidth of each of its attached communications devices. This support may extend to include
  1. up to 15 low-speed (300-bits/sec) telephones,
  2. up to four 9600-bit/sec remote job-entry facilities, and
  3. up to four high-speed (to 250K bits/sec) communications lines attached to other computers (e.g., the local PDP-10 system or local VAN nodes).

- It must support about 10^9 bits of rapid-access (about 10^7 bits/sec) online storage for interactive interrogation.
- It must be highly reliable. If it is to be the only window to data outside DOT/TSC for the DOT analysts, then it must be available when needed. A total average downtime of about 1% (including less than one unanticipated failure/week) is expected to be satisfactory.

The factors involved in defining and meeting a set of performance specifications for a complex system like MODDS are very involved. For example, the requirement of one-second response to a user request may be impossible to meet if the request requires service from a remote system taking longer than one second to respond. The only ways to effectively determine if such a complex system will meet a set of performance specifications are through elaborate simulation or actual testing. The testing of a MODDS prototype system is required to satisfy this need.

MODDS IN A DISTRIBUTED ENVIRONMENT

MODDS, as previously explained, is established between a user/analyst's terminal and local and remote computers (Fig. 5). Its central nature permits accountability, while its intelligence provides a nondistributed view to users (Fig. 6).

The flexibility of this design also allows users to explicitly transfer files between computers, communicate between processes on different machines, communicate between users, etc. Thus, the level of user sophistication determines the level of access mechanisms available.

Figure 5 also shows another site with MODDS. An added site can be interfaced several ways: to another MODDS, for example, to minimize the software development required. However, this creates a bottleneck at the central MODDS. A better solution is to locate all resources near a MODDS and communicate with another MODDS for resources near it (thus effectively sharing resources). After a period of use, accountability information can be reviewed and appropriate added direct links chosen. Little-used links could be distributed among all the MODDS.

Multiple MODDS provide a consistent, uniform view to all users. The primary advantage of multiple MODDS occurs when they are interconnected. The interconnection is simpler than from a MODDS to a host, because no translation or simulation is required. Initially, only a connection with another MODDS

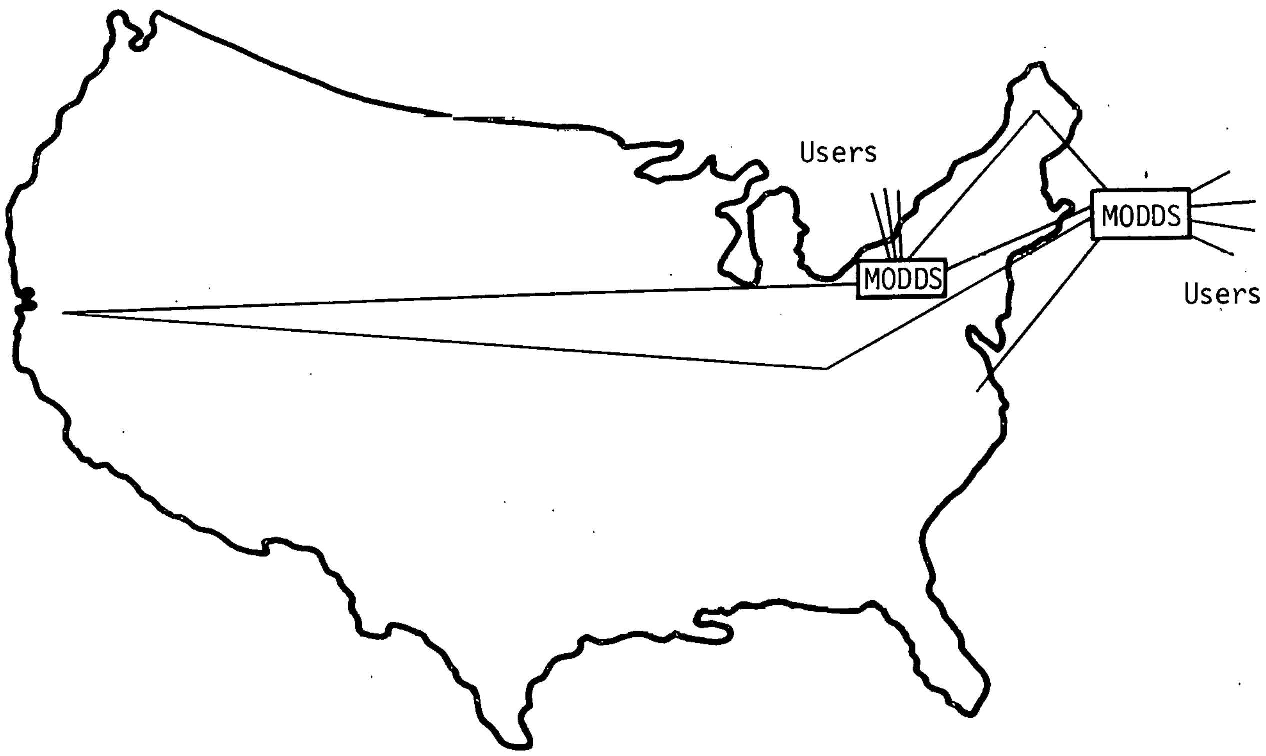


Fig. 5. MODDS in a distributed environment.

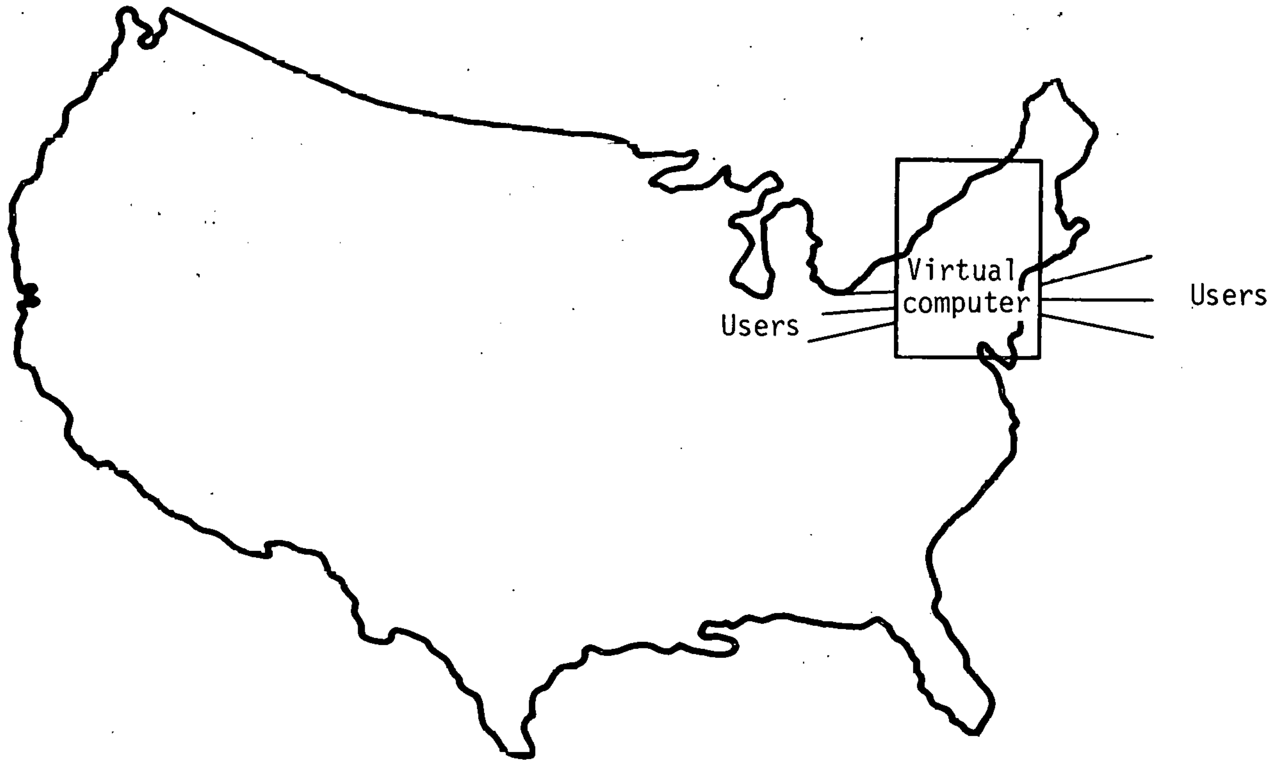


Fig. 6. User's view of MODDS in a distributed environment.

would be required (if the desired hosts were accessible from an existing MODDS). Added connections can then be interfaced, extending the capabilities of all MODDS users.

CONCLUSIONS

MODDS improves accountability, resource sharing, usability, and control of distributed data systems. Improved accountability allows an organization to accurately charge accounts for resources used and to optimize resource allocation. Intelligent choices of communications media and location of data bases can minimize organization cost.

MODDS' resource-sharing capabilities permit the sharing of a wider range of resources among a broader user community. Users can query an online directory of resources to locate, share, and ultimately use existing resources. Sharing resources to a greater degree eliminates duplication of effort and decreases costs.

One of the biggest problems with distributed data systems is that they are difficult to use. A lot of time is dedicated to tasks such as learning the systems and transferring data to and from the systems. Assisting the users of distributed systems by making all resources appear local will increase the user effectiveness and decrease organization cost. Another advantage of the uniform user view is that certain analysts who previously relied on programmer support can directly use computer resources and therefore decrease the organization cost. This is possible because of the simplified user view presented by MODDS.

By far the most important advantage of MODDS is that it facilitates data assessment. Data available to every analyst in a uniform manner eliminates time-consuming formatting and manipulations. Analysts can devote more time to data analysis and assessment. Thus, MODDS aids DOT/TSC in its task of determining the meaning and significance of transportation statistics.

NOTES ON APPENDIXES

The paper in Appendix C contains some initial investigations into implementation approaches for a transaction controller (another name for MODDS). Appendixes C and D discuss mechanisms made possible through the DCCS protocols discussed in the Appendix A paper. The DCCS mechanism is based on RATS (Appendix B).

APPENDIX A

A DISTRIBUTED-CAPABILITY COMPUTING SYSTEM

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

See

PREPRINT UCRL- 77800

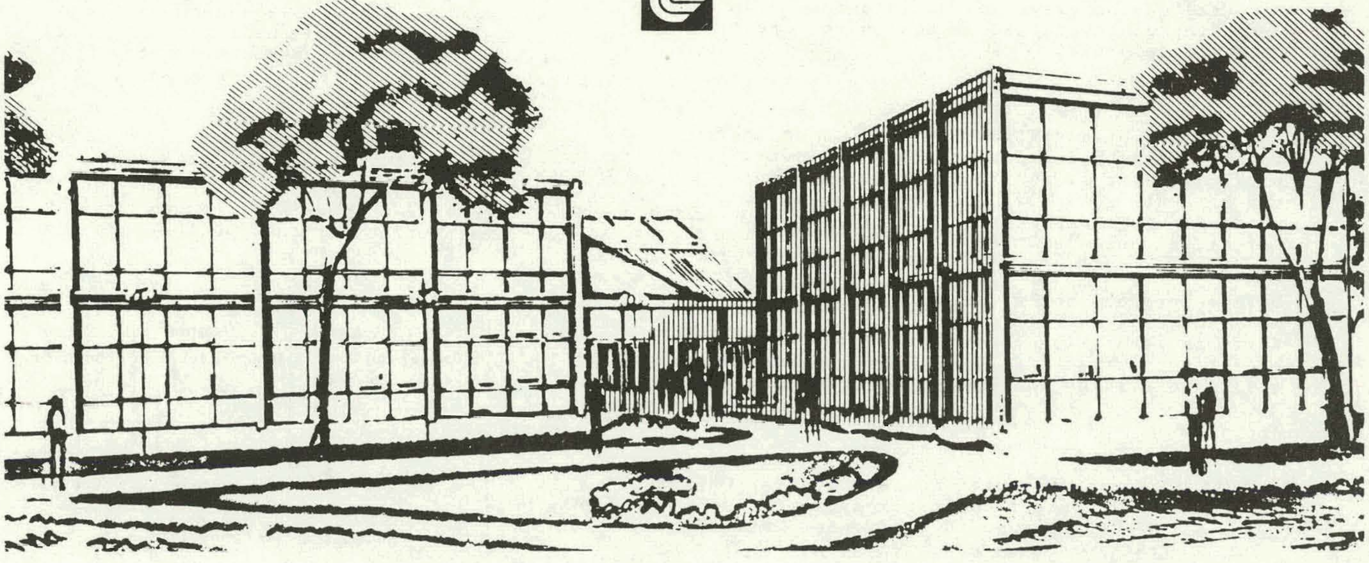
Lawrence Livermore Laboratory

A DISTRIBUTED CAPABILITY COMPUTING SYSTEM

J. E. Donnelley

April 15, 1976

This paper was prepared for submittal to the Third International Conference on Computer Communication, August 3, 1976, Toronto, Canada.



A DISTRIBUTED CAPABILITY COMPUTING SYSTEM*

J. E. Donnelley
Lawrence Livermore Laboratory
Livermore, California

ABSTRACT

One component of the resource sharing problem, that of designing protocols and mechanisms to access standardized resources, is solved. The first step in the solution is the definition of a standardized computer resource based on a set of interprocess communication primitives that are introduced. Building upon these primitives, this paper presents a mechanism for sharing the standardized resources that is independent of the resource types. Applications are presented which provide transparent network resource optimization and an automated money exchange.

Keywords: Distributed, Capability, Network, Protocol, Process, Port, Communication, Synchronization

INTRODUCTION

The motivation for the Distributed Capability Computing System (DCCS) is the advancement attainable through resource sharing. There are many components of the problem of resource sharing that will always be with us: creation of effectively standardized resources, optimization of the available communication channels, minimization of the effects of hardware errors, translation between semantically equivalent resources to increase their availability, etc. The tenet of this paper is that one portion of the computer resource sharing problem, that of designing protocols and mechanisms to allow access to standardized resources through standardized communication channels, can be isolated and solved.

Solving this portion of the resource sharing problem involves two components: defining a standard resource and constructing a mechanism for sharing such a resource across system boundaries. The definition of the standard resource can be viewed as an integration and extension of several ideas that have appeared in the literature.

PORT

The concept of a software port[1,2] is analogous to that of a hardware port (Fig. 1). Just as physical ports allow processors to communicate via hardware channels, logical ports allow processes to communicate via connections which may or may not span processors.

Unlike hardware ports, however, where access is granted by an engineer, access to software ports is generally granted by the supporting operating system. The function of maintaining a list describing the ports and other resources available to a process is common to all operating systems. By the time the generally-accepted name of such a list emerged as capability list or C-list,[3] capability-based interprocess communication had extended beyond the synchronization and communication primitives usually associated with ports.

CAPABILITIES

As motivation for extending the port concept to that of a capability, consider the communication between a process and its supporting operating system. The operating system can not only communicate information to the process, but can also grant the process other ports. If we are to model the communication between a process and its supporting operating system with ports, we must allow ports as well as information to pass over the connection. When so extended, a port becomes a capability.[3-7]

THE CAPABILITY COMPUTING SYSTEM (CCS)

To introduce the interprocess communication primitives needed to implement the DCCS mechanism in a type-independent manner, a simplified capability-based operating system, CCS, is described below.

A CCS process consists of two parts: its memory and its C-list. When a process is started, it executes instructions from its memory much like a commercial processor. The analogs to input/output instructions are termed

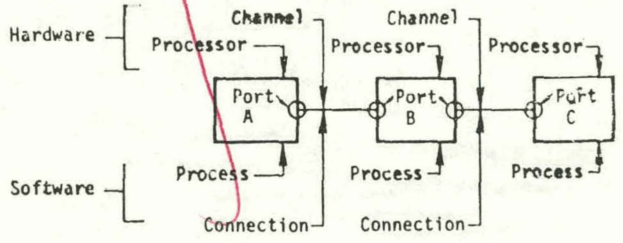


Fig. 1. Hardware/software port analogy.

*This work was supported in part under Environmental Protection Agency contract #EPA-IAG-D5-E681-DB and in part under Department of Transportation contract #[RA] 76-12. The work was performed under the auspices of the U.S. Energy Research and Development Administration under contract No. W-7405-Eng-48.

APPENDIX B

THE RATS OPERATING SYSTEM

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

see

UCRL - 77378
PREPRINT

LAWRENCE LIVERMORE LABORATORY
University of California/Livermore, California

THE "RATS" OPERATING SYSTEM

Charles Landau

September, 1975

THE RATS OPERATING SYSTEM

BY CHARLES LANDAU

ABSTRACT

RATS is a general-purpose timesharing system for the DEC PDP-11/45 computer. Although written for a small computer, it incorporates a number of advanced features, including virtual memory, directory hierarchies, capabilities, and C-lists. It provides for complete extensibility by means of a type of capability called an entry. These features were found to permit software designs conducive to security and reliability.

Key words: Operating system, protection, security, reliability, capabilities, PDP-11, extensibility.

APPENDIX C

CONTROLLING TRANSACTIONS BETWEEN DISTRIBUTED COMPUTER RESOURCES

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

See

PREPRINT UCRL- 78282 Rev. 1

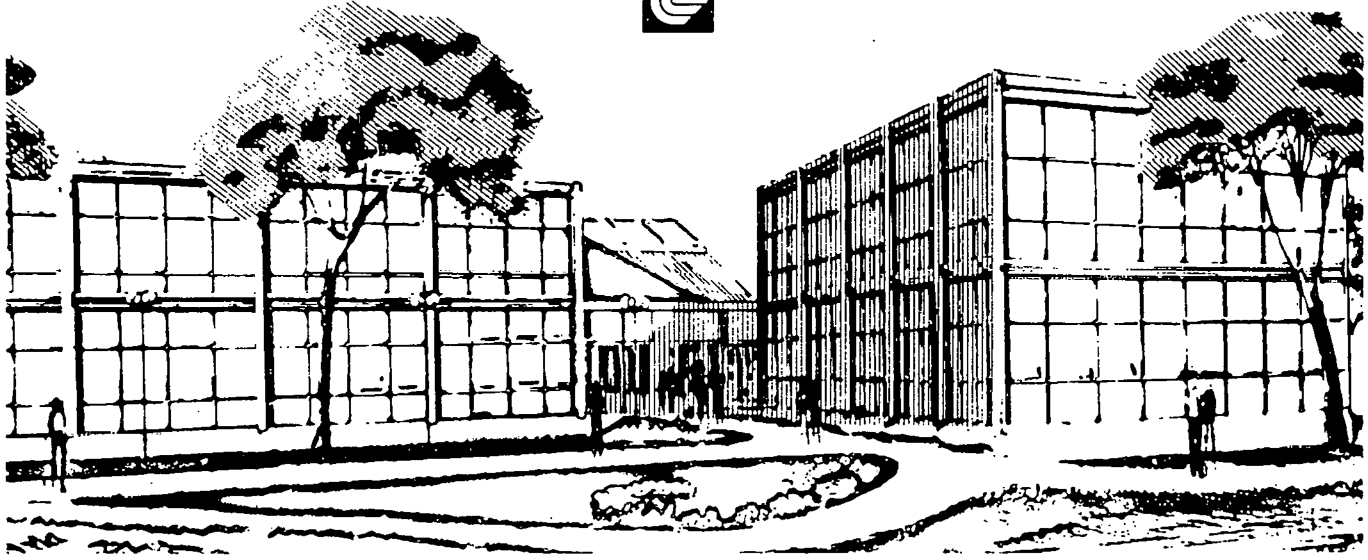
Lawrence Livermore Laboratory

CONTROLLING TRANSACTIONS BETWEEN DISTRIBUTED COMPUTER RESOURCES

J. E. Donnelley

June 4, 1976

This paper was prepared for the Fifth Texas Conference on Computing Systems, October 18, 1976, Austin, Texas



CONTROLLING TRANSACTIONS BETWEEN DISTRIBUTED COMPUTER RESOURCES*

James E. Donnelley
Data Management Research Project
Lawrence Livermore Laboratory
Livermore, California

ABSTRACT

Computer resources available today could be applied cost-effectively to the solution of many information processing problems if convenient access to the resources could be supplied. One way to improve significantly resource access is to automate the tedious manual procedures currently required to access distributed resources. Toward this end, the Data Management Research Group at the Lawrence Livermore Laboratory is under contract to develop a prototype Transaction Controller system to provide analysts with direct access to distributed computer resources by making external resources internally available in a unified manner. The Transaction Controller differs from other systems with similar objectives in that its capability-list operating system kernel supports an extendable set of uniformly processed internal objects and an enforced separation of internal responsibility which can be extended readily to distributed resources. The type-independent resource sharing mechanism built upon these facilities allows most of the Transaction Controller software to concentrate on the difficult task of translating external resources which are physically different but semantically similar into identical internal resources

Keywords: Distributed, Computer, Resource, Access, Analyst, Transaction, Terminal, Communication

INTRODUCTION

For the purposes of this paper, we define an analyst as a person who uses computers in his work. The pace of modern society and the speed of the digital computer are encouraging more and more of us to join this class of computer users from areas as diverse as business administration, politics, the sciences, econometrics, etc.. Analysts must know what data they need and what kind of analysis they wish to perform, but they usually are not computer scientists. They generally are not interested in computer terminal characteristics, communication networks, remote access techniques, login procedures, passwords, job submission procedures, system data formats, programming languages, and all the other implementation details that are the specialties of the computer people (an exception, of course, is the computer analyst).

Due to a natural process of evolution, computer facilities have become widely distributed. One factor prompting this distribution is the emergence of companies offering diversified program and data base services. Also, computer centers that support analysts in one application area often find that groups working in related areas have developed programs or collected data needed by others as well. The economic ineffectiveness of reprogramming or locally adapting all needed programs, and of copying and locally storing all needed data, is forcing formerly self-sufficient centers to consider alternatives.

Any alternative to a self-sufficient computer center requires communication. The greatest benefits from such communication are derived by using remotely located resources in place. This strong economic motivation has prompted the emergence of a considerable number of digital communication networks. What do all of these communication networks do for the analyst?

Contemporary Access to Distributed Computer Resources

Too often the analyst is left to face the expanding and rapidly changing technology of digital computer communication armed with only a terminal and a telephone. To make full use of the available facilities, the analyst has to use an array of telephone numbers, terminal parameter settings, network access procedures, executive language commands, host login procedures, job submission procedures, etc.. These have to be used every time remote facilities are to be accessed. Learning and keeping up-to-date with these access procedures, in addition to knowing the latest application algorithms and available data, is a considerable burden. To date, most analysts have had to either:

1. Concentrate on only a very few systems.

2. Channel most requests for computer analysis through computer specialists.

3. Work without the aid of computers.

*This work was supported by Department of Transportation Contract [RA] 76-12. The work was performed under the auspices of the U.S. Energy Research and Development Administration under Contract No. W-7405-Eng-48.

APPENDIX D

EXTENDABLE INFORMATION FORMATS

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

See

PREPRINT UCRL- 78211

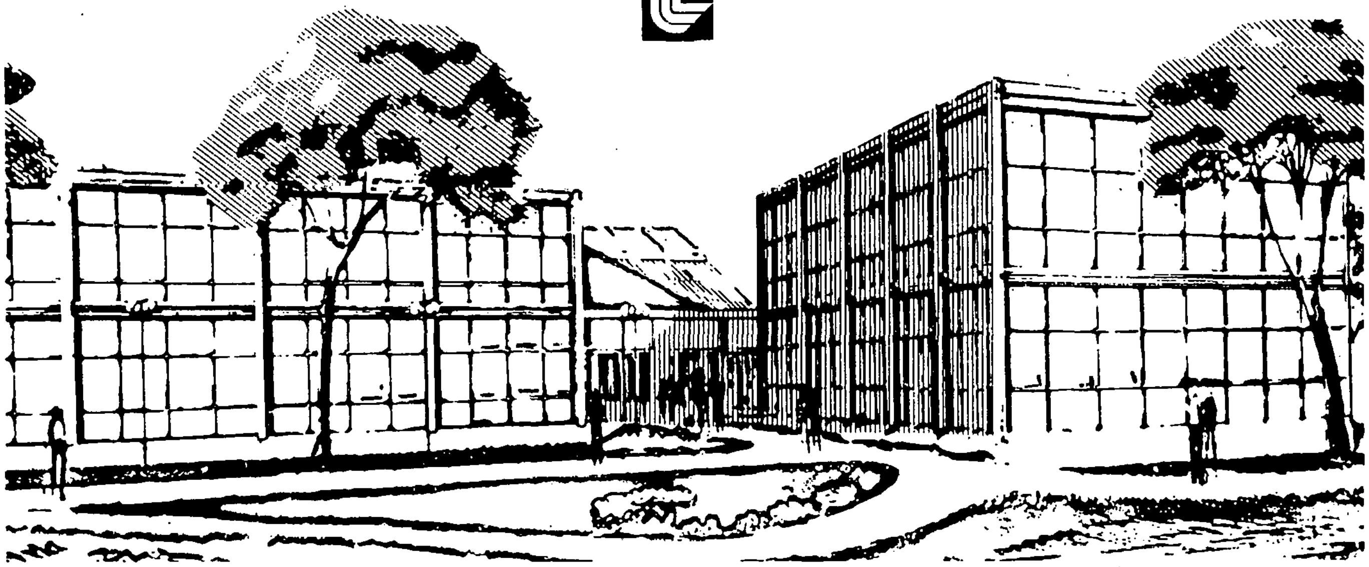
Lawrence Livermore Laboratory

EXTENDABLE INFORMATION FORMATS

James E. Donnelley

May 12, 1976

This paper was prepared for submission to the Berkeley Workshop on Distributed Data Management, May 25-26, 1976, Berkeley, California



Extendable Information Formats*

James E. Donnelley
Lawrence Livermore Laboratory
Livermore, California

Abstract

Information formats can often be adapted for use by processes different than those for which they were initially designed. The extent to which a format can be so adapted without semantic modification is a measure of its extendability. There are many information formats used today in communication protocols, data storage structures, etc., that are severely limited in scope because of their lack of extendability. By examining some of these formats and determining modifications which can make them more extendable, a set of design principles can be extracted which may be used to generate more extendable formats.

Keywords: information, format, communication, protocol, data, storage, structure, address

---

* This work was supported in part under Environmental Protection Agency contract #EPA-IAG-D5-E681-DB and in part under Department of Transportation contract #[RA] 76-12. The work was performed under the auspices of the U.S. Energy Research and Development Administration under contract No. W-7405-Eng-48.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $3.00

Page Range	Domestic Price	Page Range	Domestic Price
001–025	$ 3.50	326–350	10.00
026–050	4.00	351–375	10.50
051–075	4.50	376–400	10.75
076–100	5.00	401–425	11.00
101–125	5.50	426–450	11.75
126–150	6.00	451–475	12.00
151–175	6.75	476–500	12.50
176–200	7.50	501–525	12.75
201–225	7.75	526–550	13.00
226–250	8.00	551–575	13.50
251–275	9.00	576–600	13.75
276–300	9.25	601–up	*
301–325	9.75		

*Add $2.50 for each additional 100 page increment from 601 to 1,000 pages; add $4.50 for each additional 100 page increment over 1,000 pages.

LG/gw/vt/gw