EDP ANALYZER

JUNE, 1976
VOL. 14, NO. 6

DISTRIBUTED DATA SYSTEMS

One of the lessons of the first two decades of computer use has been: big systems often mean big troubles. So there has been a lot of attention given to ways to sub-divide large application systems—into sub-systems, into modules, and now into distributed systems. There are, however, some varying concepts being advocated under the banner of distributed systems—distributed processing, distributed computing, and distributed data bases, to name three. In this first of two reports, we will discuss some user experiences with distributed *data* systems. These are systems where large amounts of stored data are involved but the data is not necessarily organized as a data base. Next month, we will consider network structures and protocols for distributed systems.

Distributed data systems are generally those application systems that are relatively large and widespread and that have been partitioned and dispersed over a geographic area. There is a variety of approaches to this partitioning and dispersal. We can best give an idea of this variety by discussing the experiences of several organizations.

Aeroquip Corporation

Aeroquip Corporation, with headquarters in Jackson, Michigan, is a subsidiary of the Libby-Owens-Ford Company. Aeroquip is a leading manufacturer of fluid power components—hose lines, coupling, fittings, and so on. Annual sales are in the order of $240 million and the company employs about 6,000 people. The central data processing equipment at Jackson consists of an IBM 370/158 operating under VS-2, with both IBM 3330-II and Itel 7330-II disk storage.

In 1972, a number of Aeroquip plants in the midwest area were using an on-line query system tied into Jackson. But the company's major division, the Industrial Division, had such widespread operations that management considered an on-line system for this division to be uneconomical. The division headquarters are in Ohio, and "industrial branch plants" (warehouses) are located in Oregon, California, Texas, Minnesota, Georgia, and New Jersey. Typically, only a few people (less than 20) are employed at each branch plant.

The goal at each branch plant of the Industrial Division is to ship a customer's order on the same day it is received. This means processing the order, checking the inventory, checking the customer's credit, preparing the warehouse documents and shipping documents, and then assembling and shipping the order. An on-line system, with inventory and credit information, would be a big help in getting orders processed,

management recognized. In addition, an on-line system would be helpful in quickly answering customer queries about stock availability.

So the data processing staff began investigating a distributed system approach, as an alternative to an on-line system served from Jackson. The system that was eventually selected was built around the Sanders 810 Cluster Programmable Terminal System at the branch sites. A typical configuration consists of 2 to 3 CRT-type terminals, 5 Mbyte disk storage, a Sanders 200 or 600 lpm printer, modem, telephone equipment (including unattended send-receive), Sanders disk management operating system, plus Aeroquip-developed applications and system software.

The system has been programmed to handle some 300 different types of transactions, on a "fill in the blanks" basis. The terminal operator simply indicates the transaction type and the format is then displayed on the screen.

The main data files for this application are maintained at the central site at Jackson. Inventory records are 520 characters in length, for instance. At the branch locations, only selected portions of the data, pertinent for local needs, are stored. In this instance, inventory records are 80 characters in length.

As the system was originally designed and installed (in 1973), orders were entered on the terminal, validated, and stored on disk. The operator would enter customer number and the system would fill in customer name and address. Similarly for parts information; the part number would be entered and the system would supply part status. The outputs for warehouse disbursing and shipping documents were put in an output queue and printed in their turn. The transactions were retained for transmission to Jackson, in compressed form.

At night, the central site would dial each branch plant and automatically request transmission of the accumulated transactions. These were posted to the master files and output reports prepared. The updated records and output reports were then transmitted automatically to the branch plants.

To enhance the system, Aeroquip has recently revised the design somewhat. Now the central computer dials each branch plant (via a WATS line) every five minutes for message switching and queries which cannot be handled at the local site. As before, the terminal batches transactions for transmission to the central site at close of business and receives batches back after processing.

This revision in system design allows for even closer control of inventory. In addition, stock availability queries need not be limited to just the local branch plant; instead, stock at other locations can be considered. Also, all backorder and shipment information, which is available at Jackson, can be taken into account. So, within a relatively few minutes, a representative at a branch plant can give a customer a complete answer to the customer's query about stock availability or order status.

A complete Sanders system is installed at Jackson. If a branch plant system has trouble, the operator calls Jackson to inform them. A specialist at Jackson dials the branch 810 from the Jackson 810 and takes control of the remote 810 for running diagnostics. The specialist can often pinpoint the trouble to a particular part of the machine—so when the Sanders service representative is called, he can be told where the trouble seems to lie. In general, system reliability has been very good, we were told.

If the central computer detects data transmission errors, it first counts those errors as they are detected. If the error rate exceeds a designated threshhold, the central computer disconnects and redials. The remote site, sensing a loss of carrier, automatically restarts. If a remote site detects transmission errors, it disconnects and automatically restarts, to be ready when the host re-establishes the circuit. It automatically initiates diagnostics upon detecting certain types of disk errors.

So a distributed system has given Aeroquip the advantages of an on-line system without the need of providing all processing services from the central site—and at less cost than for a centralized on-line system.

Société Générale

Société Générale, a nationalized bank with headquarters in Paris, France, is the third largest bank in France. It has some 2000 branches throughout France (plus branches in other countries), as well as about 300 automatic cash dispensers, which are almost automatic tellers. The bank has some three million checking accounts

and one-half million loan accounts, and employs about 35,000 people.

Currently, Société Générale has two main data processing centers—one near Paris and the other in Aix-en-Provence, just north of Marseille. All input from the many branches comes to these two centers each working night. The processing is performed and the reports and other documents are delivered back to the branches before 8 AM. For this physical transportation of documents, the bank uses trains, buses, delivery people on motor bikes, and so on.

In 1972, the bank began studying what its next generation of data processing system would need to be. Transaction volume was growing rapidly and it was clear that the existing system would operate dangerously near capacity by 1978. In fact, it was projected that by 1980, transaction volume would reach 100 transactions per second. It was concluded, based upon the capabilities of the then-announced computers which they considered, that one or two centers would not be able to handle this volume of transactions expeditiously and reliably. So the data processing staff began to consider a distributed system. By 1973, they had convinced themselves and bank management that the bank should aim at a distributed system.

The structure of their distributed system is most interesting. It is a hierarchical system. At the top of the hierarchy is the Paris center. At the next level, performing about 30% of the daily batch processing (which we will discuss shortly) and acting as backup for the Paris center, is the Aix-en-Provence center. At the next level are the group (regional) centers. The bank has had over 200 groups, which operate very much like small banks and which have a number of branches under them. Bank management did not want to impose a reorganization of the banking structure on top of a data processing system change, so the distributed system was designed to have a remote processor at each group. Each group, in turn, serves an average of about 10 branches. Each branch will have a terminal controller and an average of about five terminals, for a total of some 10,000 terminals in the system. The terminals will be non-intelligent CRT-type units, with printers attached. Application and control logic will reside in the terminal controllers, for first level control, and in the group processors for the remaining control.

The terminals will be tied into the group processors via the controllers in an on-line manner. During the working day, tellers will operate the keyboards of the terminals—to verify account balances, to record deposits and withdrawals, to perform end-of-day balancing, and so on. Selected information for each customer served by a branch will be stored in disk storage at group level. During the day, these records will be updated on-line—and the same transactions may be forwarded to Paris or Aix.

The complete customer file, located in Paris and Aix, will include a history and full description of the banking products used by each customer; the average record length will be 1500 bytes. During the day, this file will be available for inquiry from all terminals. In addition to these inquiries, some updating of specific centralized files will be done directly from terminals. About 10% to 15% of the total transactions entered on the terminals are expected to use the central files in an on-line basis.

About one-half of the total updating transactions come from outside sources—from other banks, clearing houses, and so on.

In the original design of the system, all "official" updating was to be done by the central systems, at night on a batch basis. This updating would include the transactions received from the terminals as well as those received from external sources. At the completion of the updating, the updated records and reports would be transmitted back to the groups. This approach might be termed "trusting the central system." But we understand that Société Générale is still debating this point and may eventually put more reliance on the group processors, as they assure themselves of the viability of the approach.

About 90% of the transactions originating within the groups apply to records stored at the same group processors. The other 10% apply to records stored at other groups. These inter-group transactions will be transmitted automatically to the appropriate groups. Hence a customer from, say, the south of France can easily cash a check at a branch in northern France. Security rules will be applied for data that can be transmitted for these inter-group transactions.

The bank has paid a lot of attention to reliability and backup within the system. The Paris and Aix centers provide backup for each other. If a

group processor is down, switches can be thrown to route transactions to either Paris or Aix. So these central sites provide backup for the groups. It has been assumed that as many as ten group processors might be down at any one time. One weak point in the system might be the terminal controllers in the branches; if one goes down, it disables all terminals at the branch. The bank does not think that this will prove to be a problem area because of the simple structure of the controller. If it does, dual controllers might be used at each branch.

Société Générale selected Compagnie Honeywell Bull as the prime hardware supplier. There will be two H66/80 processors, two Datanet front-end processors, and 5 billion characters of disk storage at the Paris center. The CPUs and front-ends will be cross-coupled for switching workload in case of failure. A similar complement will be at Aix, except that the CPUs will be H66/60s. Initially, the two centers will be linked by a 9600 bps line, later to be changed to a 72 Kbps line.

The group processors will be H716 mini-computers, each with 10 million characters of disk storage for local files and a 300 lpm printer. At the larger groups, magnetic tape drives will also be used.

The bulk of the equipment investment will be in the branch equipment. So the bank chose to use non-intelligent terminals, with intelligence supplied from the terminal controllers. The bank has not yet made a final selection of the suppliers of terminals or controllers.

A major economic point is that the group installations will not have computer operators. This was a large factor in the selection of equipment—equipment that could operate in an unattended mode. No operator console will be provided; maintenance personnel must bring their own console. Also, no local programming capability is to be provided for the group processors; programs will be controlled and loaded from the central sites. With over 200 group processors, an operations and programming staff for each could add up to a huge expense. The bank checked carefully into the feasibility of this type of operation before finally deciding on the distributed approach and on the Honeywell Bull equipment.

By last fall, Société Générale had begun the testing of the new system. Two Honeywell H60/60s, plus Datanet front-ends, had been installed at Paris on an interim basis. Two group processors and 50 terminals, plus testing equipment, were installed. Conversion of the first test group and four branches began in October 1975 and was completed about five weeks later. This was a limited version of the type of operation described above.

The distributed system represents a major change for Société Générale, so a slow, careful conversion has been planned. All affected programs are being converted to run on the Honeywell equipment; those not already programmed in COBOL are being reprogrammed in that language. File structures are undergoing significant change. Previously these files were account oriented. If a customer had several accounts, there was one record in the files for each account. In the new system, the data is customer oriented. All accounts for the same customer are linked together, using IDS, for both on-line query and sequential batch processing.

In all, a three year schedule has been laid out for converting the branches and groups to the new system. It is planned that the distributed system will have all nodes in operation by 1980.

Société Générale expects that their distributed system will be able to grow with the workload, adjusting to increases in transaction volume and allowing the bank flexibility for opening new branches and groups.

Other companies' approaches

To give an idea of the variations in approaches to distributed systems that we encountered, we will briefly describe what some other companies are doing.

Fireman's Fund Insurance Company

Fireman's Fund American, with headquarters in San Francisco, California, is one of America's seven largest insurance companies. It is approaching the use of a distributed system by way of remote data entry.

In 1972, FFA made its first step in this direction by installing Four Phase IV/70 clustered terminals in six processing centers around the U.S. Operators, working from copies of insurance policies that had been received from branch offices, would enter the data in a "fill in the blanks" manner on CRT terminals. The applications had been "rated" (assigned to a premium rate category) at the

branch offices. But no longer did the staff at the centers have to assign codes or keypunch the applications; keypunching per se was eliminated. Instead data was entered in raw form and transmitted to headquarters. The coding was automatically done on the IBM 370 computers at headquarters. In addition, validation of input occurred during data entry, allowing earlier detection and correction of errors.

Last year, FFA made another step toward distributed systems, by installing Four Phase equipment in the branch offices. For one thing, this speeded up the overall data processing. The operator could now work from the agent's handwritten application form, without the need to rate the application or type up the policy. The information is transmitted to headquarters in a batch mode, where the application is rated, coded, and processed. Policy information is sent back to the branches for the automatic printing of the new policies and endorsements.

The next step toward distributed systems, planned for this year, is to set up local files at the branch offices. These files will be primarily cross-indexes—obtaining name from policy number, obtaining policy number from name, and so on. Further, it is planned that the branches will be tied into the central system in an on-line basis, so that the central data base can be queried, for making changes and for assistance in processing claims. The branch offices will continue to process claims and make out the checks in the payment of claims.

So FFA is moving toward "distributed processing" by moving intelligent terminals to the branch offices. At present, the company does not have plans for storing much policy-holder data in these branch terminal systems.

Lowes Companies, Inc.

Lowes Companies, Inc., with headquarters in N. Wilkesboro, North Carolina, is a chain of some 140 retail/wholesale stores dealing in building materials, appliances, and hardgoods. The stores are located in 15 states of southeastern U.S. Sales are over $340 million annually and the company employs about 3,700 people.

Acree (Reference 8) has discussed the experience of the company in installing a distributed system; it has also been written up by Data General.

From 1963 to 1974, Lowes used IBM 402 card accounting procedures to control inventory and print customer invoices. Card tub files were used for inventory control, with one card for each item in stock. But Lowes had the usual problems of cards misfiled, out of date prices, and so on.

So in 1972, they began to look for a better solution. One proposal was for a large computer installed at the central office and tied to remote intelligent terminals in all of the 140 stores. This system would have handled the current workload and would have speeded customer service. But the costs of the intelligent terminals and the communication lines, plus the lack of backup for the central site, argued against it.

Another alternative that Lowes considered was the distributed system. The more they looked at it, the better they liked it. So in early 1973, they began installing the new system based on Data General NOVA 1200 mini-computers. Each store would have its own stand-alone system, with 5 million bytes of disk storage and up to 16 small CRT terminals. And each store would have unattended data communications capability to the central office.

With the new system, store personnel have on-line access to inventory, pricing, and customer account information. The system prints customer invoices at the time of sale. It saves transactions and at night transmits inventory and sales information to the central office via unattended operation. At the same time, the system receives updating information for its files, such as new price data.

The first store was converted to the new system in early 1973. By the end of 1975, some 42 systems had been installed. They were being added at the rate of one per week. A store installation is done by one person in about 1½ days.

The system has resulted in a 30% increase in sales person efficiency. It provides more accurate and timely inventory and pricing information, and it gives better control over credit sales and accounts receivable.

Kennington Motor Group

The Kennington Motor Group, with headquarters in Chesterfield, England, is a large motor vehicle distributor. The company also has car rental services, auto parts supply stores, filling stations, etc., under a number of corporate entities. There

are 300 self-accounting locations. See Reference 1 for further details.

The company has a central system that uses an IBM 360/40. In addition, they have moved toward a distributed system by installing a number of IBM 3741 Model 2 data entry systems at remote locations. These systems have floppy disk storage capabilities as well as data communications and automatic answering capabilities.

Sales and purchasing transactions are entered on the 3741s during the day. The data is validated upon entry and error corrections made immediately. The transactions are then stored on the floppy disks. At night, the central system dials the remote locations and requests transmission of accumulated transactions.

Night-time data communications has a number of advantages, say the people at Kennington. For one thing, service is less expensive at night. For another, there is less traffic on the network and hence less interference, so transmission is more reliable. Also, when transmission failures do occur, they are not observed by the operating staff, so the staff is less likely to lose confidence in the system.

The structure of distributed systems

Consolidated systems have been a major trend within the computer field almost from the first days of the computers. The early computers were relatively large and costly so that an effort was made to consolidate applications and files. This was followed by the consolidation of decentralized installations into larger centralized ones. Data bases, too, have tended to be centralized. When remote terminals have been used, they have in essence been tied directly into the central system.

But now a trend toward distributed systems is developing; it has been underway for the past three or four years, at the least. There are a variety of ways in which systems are being partitioned, as was illustrated by the several examples discussed above. We see several main variations of partitioned systems:

Distributed processing systems

We use the term "distributed processing systems" to mean those in which the data files are either centralized or in which large data files are not involved, but where application logic is distributed.

Data entry systems. One currently popular approach is to distribute the data entry function so as to locate it close to the source of transactions. Intelligent terminals, or key to disk systems, are used for performing some of the input validation function. This approach seems to be a natural outgrowth of consolidated systems and we would expect a wide following for it.

Remote file access systems. This approach is quite close to the popular on-line systems of today. Remote terminals are connected to a central system, where essentially all of the data is stored. However, the terminals perform some of the processing. Withington (Reference 2) sees the fourth generation of computers in terms of this type of system. This fourth generation will have a central, disciplined data base, transaction (not batch) processing, mini-computers at the point of transaction for performing some input functions, and dynamic interaction with the user. Two or more central processors will work asynchronously, he says, with the next job in line being assigned to the next processor which is free. There will be a hierarchy of file storage devices and a communications network with either a hierarchical or a loop structure. We would expect to see a wide following for this approach, also.

Distributed computing systems. The concept here is quite different from the types described above. The best example is some research work being performed on the ARPANET; see Reference 3. The concept is the simultaneous, coordinated use of two or more (perhaps dissimilar) remote computers for the solution of a computing problem. A part of the problem might be worked on computer A at one location; it transmits its results to computer B at another location, to perform the next portion of the calculations, and so on. Data files tend to be relatively small; they might be transmitted from one computer to another or might be duplicated at each site. A large data base generally is not involved. The concept is intriguing, although it is not yet clear just how it might be applied in a business environment.

Distributed data systems

We use the term "distributed data systems" to mean those in which the application data is lo-

cated near the distributed processors. Again there are variations.

Centralized/distributed data systems. The concept here is that of a central, controlling data base tied in with local, application-oriented files maintained by local processors. The central data base is updated on a batch basis, perhaps once a day, while the local files are updated during the working day in an on-line basis. Synchronism may be established by the transmission of updated records from the central system to the local processors at prescribed intervals.

We see this approach also as being a popular one. It leaves system control in the hands of well-established batch processing procedures, avoids on-line updating of the controlling data base, and yet provides the operating people with on-line access to operating data.

Fully distributed data systems. In this type of system, there is no central data base. Rather, the data base is partitioned and distributed among the nodes of the overall system. This approach is at the opposite end of the spectrum from the consolidated system.

When data is distributed, it is also likely to be duplicated. For instance, in the centralized/distributed data systems, the master records are in the central files and selected portions of those records are stored at the remote sites. The problem of synchronizing the different copies of the same data then arises. We will have more to say about this important problem of synchronism later in this report.

We will not attempt to discuss in this report all of the above types of distributed systems. Instead, we will concentrate on the distributed data systems where at least some of the application data is stored at the remote sites.

It is worth mentioning the variety of types of equipment that we assume may be attached to the nodes of the distributed network. These types include: (a) computers, sometimes called host computers since they might also support local terminals; these may have large, medium, or small logical size, and can include what has been termed "small business computers"; (b) intelligent terminals, including keyboard terminals, remote batch terminals, and graphics terminals; and (c) work stations, which have many characteristics of small business computers but which are tailored to specific types of applications and which are similar to regular office machines in their flexibility, each of use, and reliability of operation. In each case, a node of the network has both processing and data storage capability.

Why partition?

Since the consolidation of data processing functions has been a major trend in the field, the first question to address is: why partition data systems?

Von Simson (Reference 4) provides one clue. His firm performed a study for a client who wished to investigate the reliability of data processing systems and the ability to recover from system failures. The study found that two factors were paramount—system complexity and system criticality. Complexity was defined as a highly inter-related data base, with a multi-level hierarchy or network structure. With such a data base system, it was found that failure was more likely to occur than in more simply structured systems, and that it was harder to recover once failure did occur. Criticality was defined as data systems that were closely coupled to the day-by-day operations of the enterprise. With tight coupling, if the data system goes down, the affected operations of the enterprise stop.

The study covered 59 organizations. Of these, six were found to have quite interesting approaches to critical sub-systems. In each of these six cases, the companies had partitioned these sub-systems and implemented them in fairly simple ways, often on dedicated equipment. With simple design, these sub-systems were less likely to fail and easier to recover when failure did occur. The use of dedicated equipment reduced the complexity as well as the chance of failure.

In previous issues, we have mentioned the fact that more and more data systems are moving into the "main line" functions of enterprises—that is, helping the operating people of the enterprises do their day-by-day jobs. As von Simson's study indicates, these are the systems that need to be partitioned—to reduce the chance of failure and to make recovery easier in case failure does occur.

Helgeson (Reference 5) discusses some of the motivations for distributed systems. User requirements for processing are growing faster than is computing power, he says, and user data bases are growing faster than is storage capacity with reasonable access times. Under such conditions, distributed systems provide a more natural fit to the

organization than do centralized systems. They allow for local control over critical operations, and they allow an organization to reduce the number of paper reports.

The problem with distributed systems, says Helgeson, is how best to answer four questions: (a) what functions should be distributed? (b) where should they be distributed? (c) how to control the operation of the hierarchy of functions? and (d) how to insure data base integrity?

There are other motivations for distributed systems that we have come across in our study of the subject. One, of course, is to enhance reliability. If a large central system goes down, everything tied to that central system stops. If one node of a distributed system goes down, only that node is affected. Related to this question is that of restart and recovery. With one central data base, data base dumping (checkpointing) can take a very large amount of time. If the data base is distributed, the checkpointing job can be divided among the several nodes. Another reason is that processing costs are falling faster than are communications costs; in the trade off between processing and communicating, new computer technology tends to favor the processing. Another reason might be that transaction volume for a given application may grow to the point where it is not feasible to handle it with a centralized system—too many CPUs, too many communications lines, and so on. Still another reason is to put intelligence out at the source of transactions, so that input validation can be performed and errors detected and corrected on the spot.

We suspect that the overriding reason for partitioning will be to reduce system complexity and criticality.

How to partition

We see three main ways of partitioning systems:

- Partitioning of an applications system
- Partitioning by functional area
- Partitioning of the data processing function

It is still too early in the days of distributed systems to talk about a "typical" approach. We would expect to see all three of these ways used, probably in combination. So we will discuss each one briefly.

Partitioning of an application system

In partitioning an application system, the system designer must search for the natural clustering of activities within the system. The partitioning should then be such that most of the activity of a partition occurs within that partition (let us call it "intra-node") and very little of the activity involves other nodes ("inter-node"). For example, in both the Aeroquip and Société Générale cases discussed earlier, most of the transactions entered into a regional processor dealt with customers in that region; a relatively small percentage of the transactions involved customers of other regions.

It appears to us that most of the partitioning of applications systems will be based on *location.* There is, in fact, a hierarchy of locations to be considered—country, region, site, department, and work station. The system designer should consider each level, to see if it is appropriate for a partition.

Perhaps the dominant characteristic is the relative amount of intra-node activity versus the inter-node activity. If the inter-node activity is "high," then partitioning is less desirable. We have seen no analysis for determining the threshhold between "acceptable" and "high" but we suspect that it lies somewhere between one-quarter and one-half of the total activity. If the amount of inter-node activity is above the threshhold, then a centralized system probably is to be preferred.

For instance, in the early days of the airline reservation systems, some were set up on a regional basis. But these did not work out too well and were quickly supplanted with centralized systems. The reason would seem to be too much inter-node activity. The planes themselves cross regional boundaries, and customers in every part of the country buy space in any flight. Since there was little natural geographic clustering of activity, a centralized system worked better.

There could be other factors that override this clustering of activity. For instance, total transaction volume might be growing to a point where a distributed system is required. Also, communications facilities might not support a centralized system. Or the system might use critical information, access to which on a very fast response basis is vital.

As we say, though, we suspect that distributed

application systems will ride or fall mainly on the amount of natural clustering of activity.

Partitioning by functional area

The concept here is that of a functional or departmental system, instead of each department using a central system. Each department probably would have its own mini-computer system, and the several departmental systems might be tied together in some form of network.

Acree (Reference 8) has described what is being considered by Lowes Companies along this line. The company is in the process of replacing their IBM 370/135 and 360/30 central system with a network of seven Data General ECLIPSE C/300 departmental systems.

Lowes has had the usual problems with central batch systems. For instance, manual work had to be scheduled so as to meet the computer schedule. The reports produced by the batch processing were typically days out of date. The increased volume of transactions meant a continually increasing control group—but without providing better information for management. And so on.

In the new system, a mini-computer will be dedicated to each department—marketing, purchasing, personnel, and accounting. In addition, processors will be used for communicating with the stores and for control. Typically, each processor will have 400 million bytes of disk storage, two to four printers, and 20 to 40 terminals.

With the new departmental systems, department managers and staff will have on-line access to files for answering queries; this will replace the days-old batch reports. Each department will be responsible for the timeliness and accuracy of its data. New systems will be designed with the needs of the users given first consideration, rather than the demands of the central system. Further, the new system will be modular, with one or two systems being installed at a time.

In addition to these operating benefits, the company estimates that total system costs will be reduced in the order of 20%.

Departmental systems to replace shared central systems will be a popular approach to distributed systems, we believe.

Partitioning of the data processing function

Some forms of partitioning within the data processing function have already begun to occur. One form is the use of mini-computers in key-to-disk systems and in intelligent terminals, for supporting remote data entry. Another form of partitioning has been the use of mini-computers for front-end processors, to handle data communications functions. Still another that has been discussed but not yet widely used is the idea of a back-end processor for handling the data base management function.

In our April 1973 report, we discussed how the Japanese Racing Association had installed a multi-mini type of system to replace single central computer systems, for the pari-mutuel betting application. By using multiple minis, the size of the system is easily adjusted to the requirements of the race track. Further, reliability is enhanced; if a mini goes down, another can be automatically switched in to take its place.

Comba (Reference 6) discusses yet another form of partitioning within data processing—the partitioning of the data administration function. Centralized control of the data base views all data in terms of one integrated data base, he says. Headquarters then designs the total system and defines integrity and access controls. Decentralized control sees the data base as initially dispersed; the data base is integrated and shared only as the need arises. Comba favors the decentralized control, which is evolutionary in nature and which gets direct user participation on a local basis.

This partitioning of the data administration function for a distributed system would seem to need more study and debate.

Constraints to partitioning

One possible constraint to partitioning is the current state of the art of technology *at each node of the network.* The system designers may find that some vital element is missing at one or more critical nodes of the network—hardware or software support inadequate, data communications services poor, or such.

Another possible constraint is the ability of the organization to accept change. It is important that the day-by-day operations of the enterprise not be thrown into a turmoil by the introduction of a new data system; there has been entirely too much of that in the past. Moreover, as systems become more "main line," any such disruption can

have much more severe consequences than has been true in the past. So the rate of change to a new system should be only as fast as the organization can assimilate it.

Some problems with distributed systems

Distributed systems will bring their own set of problems. Some of these problems have been discussed by Withington (Reference 2), Bolt Beranek and Newman Inc. (Reference 3), Helgeson (Reference 5), and Cashin (Reference 7), and we will draw upon those discussions. Also, some problems have been mentioned to us in our discussions of the subject with others in the field.

Operating considerations

Batch processing makes more efficient use of CPU cycles and storage accesses than does on-line processing. Also, data processing people are more familiar and experienced with batch operations. So conversion from a batch system to an on-line distributed system may involve some loss in operating efficiency.

Data integrity and security procedures are more established for a centralized data base.

If the system uses a central data base as well as local files, the local files must be refreshed periodically. If the central processing is delayed beyond a certain point, there may not be time available for refreshing those files before the next operating cycle starts.

Remote sites may encounter a wide variety of operating difficulties. Field service may be slow, due to the travel time of the service person. The service person's diagnostic routines may be inadequate, meaning that troubles continue to reoccur over a period of time. The equipment and software instruction manuals may be inadequate, due to erroneous information and missing information. Supplies may be hard to get. We have observed one such case in some detail, where the hardware/software came from a major supplier of intelligent terminals. We suspect that such operating difficulties are more the rule than the exception.

Costs of operation may become significantly higher than anticipated. Each site may want to have its own computer operators and programmers "to serve local needs." The budget for the system may not have allowed for such a staff.

Within a given state of the art, economy of scale applies. Minis represent reduced processing costs because they represent a change in the state of the art. So, within a given state of the art, a distributed system might well cost more to operate than a centralized system.

System design considerations

There will often be a need for local "customizing" of the programs and perhaps of some data definitions. Operating needs do vary from site to site in numerous instances. For example, in multinational data processing activities, one expert we talked to estimated that only about one-half of the computer program code written could be used internationally; the remainder had to be written to meet local needs. The differences among domestic sites probably will not be as great, but still differences will exist.

This possibility of local customizing raises the question: how much autonomy should be allowed for each of the several nodes? Will each one be allowed to build up an operations staff and a programming staff? Carried to the extreme, this local autonomy could seriously affect the economics of the distributed system. So a policy and a method of control will be needed.

We think that inter-node communications will be a requirement in a good percentage of future distributed systems—if for no other reason than that of "electronic mail," which we will discuss next month. System design will have to consider whether one node can communicate directly with another, or whether all communications must go through a central site. Also, what security and privacy safeguards will be required and how will they be implemented, if node-to-node communications are allowed?

System failure considerations

If a distributed system has a large number of nodes, the system designers must allow for the case when several node processors are down simultaneously. Will backup be needed, or can nodes operate for several hours with their processors down? If backup is needed, how will it be supplied? Will a central processor be available and will it provide backup? What happens when it is down?

When a node processor goes down, a satisfactory message flow must be maintained. The mes-

sages directed to that node must be directed to the backup site.

In general, the system designers must consider each component in the chain—from the remote terminal to the central processor at the top of the hierarchy (if the network is designed that way). The fall back mode of operation must be spelled out under the assumption that each component fails. That is, what is done when the terminal fails? When the terminal controller fails? When the communications link to the node processor fails? And so on.

Data file considerations

The design of file structures at the nodes must be considered. Will these be simple, single level files, perhaps accessed by an indexed sequential access method? Or, at the other extreme, will they be network-structure data bases? Or will a variety of structures be used? Will different data base management systems be used at the different nodes?

These questions are particularly appropriate when a distributed system is set up by linking existing installations.

If the file designs at the various nodes are different, what sort of user or program interface will be used? Will the user have to learn a different interface for each node? If a common interface can be used, will it be tailored to meet the specific circumstances of each node—hence making it difficult to change the hardware and/or software at a node?

The system designers must also decide upon the degree of redundancy to be used. This would be planned, controlled redundancy, not the almost casual, uncontrolled redundancy that has evolved in past data systems. Redundancy may be needed for two reasons: reliability and fast access. For reliability, it means getting at a backup copy of the data in case of a failure. For fast access, it means getting at a copy of the data quickly, regardless of the traffic in other parts of the network.

One approach, particularly for increased reliability, has been to put the master data base at the central site and selected files at the nodes. The central site would also use normal backup and recovery procedures. The central site can act as backup for the nodes.

Another approach, particularly for fast access, is to duplicate the specific files at the various nodes where fast access is needed.

Where redundancy exists, the question of "up-to-date-ness" of the multiple copies must be considered. There are a variety of possible needs, each with its own set of implementation problems. One situation is where all copies must be current. Another situation is where only the active copy need be current but where the backup copy must be brought up to date before taking over as the active copy. Still another situation is where the backup copy can be used in an out-of-date status for a period of time, until the active copy is available again. In the last two of these situations, it might be assumed that the backup copy is updated once a day, perhaps on a batch basis.

Synchronizing problems. When redundancy exists, as just described, the need for synchronizing the redundant copies arises. We understand that this has been a real problem area for some of the early distributed systems.

Bolt Beranek and Newman Inc. (References 3 and 9) has investigated some of these problems and has implemented solutions on the ARPANET. For instance, in the case of duplicate files, they have investigated and implemented ways to assure that all updates are delivered to all copies, once and only once, and that all updates are processed. They have also investigated how to keep a backup copy of a file synchronized with the active copy. They have investigated another challenging question: after switchover from the active to the backup copy, which copy then becomes the backup copy and how is *it* kept synchronized?

It seems to us that BBN's work on the ARPANET project will provide valuable results for distributed file and distributed data base systems.

Cashin (Reference 7) has observed that if two or more duplicate copies of a file are to be kept in synchronism with each other, a control process is needed to avoid the "deadly embrace" (although BBN has opinions to the contrary on this). "Deadly embrace" is the situation where each update process is waiting on another update process to complete its updating before it starts, with the result that everything stops. His suggestion is a control process that inhibits the flow of incoming transactions (including queries), then locks all copies of the records as soon as all queries in the queue have been processed, then updates all copies, and then unlocks all copies and allows the flow of transactions to resume.

This very brief discussion has perhaps given some indication of the types of complexities that can arise in distributed file and distributed data base systems.

Network structures and protocols

In a distributed network, the problems of line control and device control must be addressed. Also, there is the problem of the distribution of application logic. So the question arises: what control functions should be put where?

This subject of network structure and protocols is a complex one—and still a relatively new one. There are a variety of approaches appearing on the marketplace, including IBM's Systems Network Architecture (SNA).

Next month, we will continue our discussion of distributed systems by describing some of the pioneering views on network structure and protocols. One goal of these efforts has been to make the data communications system independent of the applications—so that new applications can be added or existing applications changed without the need (hopefully) to change the data communications system. Another goal has been to make the data communications networks independent of the computers and/or terminals that are used.

Our discussion next month will be mainly from the viewpoint of suppliers of these networks, since as far as we can tell there has not been much user experience to date.

Conclusion

In this report, we have equated distributed systems with partitioned systems, wherein a partition is one node of the distributed network. Often this network will be geographically dispersed. However, this is not a necessary condition; the network can exist within one building, for example.

If natural clustering of activity exists within the application—say, a large percentage of activity is found to be of the intra-node variety—then distributed systems have something to offer. As the percentage of inter-node activity rises, so do the arguments in favor of a centralized system.

Not much is yet known about distributed systems. On the surface, they look appealing. They offer a number of benefits. At the same time, it is possible to visualize a number of problems that may be connected with them, as we have tried to indicate. But the solutions to those problems seem to be evolving.

We suspect that "distributed systems" will become the dominant trend in the computer field during the remainder of this decade. However, distributed systems are not the best answer for everyone, particularly for those applications with insufficient natural clustering.

And we can be sure that by 1980 or so, some new concept will emerge that will challenge distributed systems. At least, that has been the history of the computer field to date.

REFERENCES

1. "Floppy disc works well for Kenning," *Data Processing* (Dorset House, Stamford Street, London SE1 9LU, U.K.), November-December 1975, p.324-7; price £12.50 per year.
2. Withington, F. G., "Fourth generation computer systems," *Proceedings of Compcon Fall 1974* (IEEE Computer Society, 5855 Naples Plaza, Long Beach, Calif. 90803), p. 111-113; price $20.
3. Bolt, Beranek and Newman Inc. (50 Moulton Street, Cambridge, Mass. (21238), reports on research conducted on ARPANET; including Report 2976, Vol. III, Natural Communication with Computers, and Report 3106, Interface Message Processors for the ARPA Network, Quarterly Report No. 2, June 30, 1975.
4. von Simson, E. M., "System fragility in complex data bases," *Managing the impact of generalized data bases* (AFIPS Press, 210 Summit Avenue, Montvale, N.J. 07645), 1973, price $6.
5. Helgeson, W. B., "Networks bring dp to source of data," *Data Communications User* (60 Austin Street, Newtonville, Mass. 02160) December 1975, p. 39-40; price $2.
6. Comba, P. G., "Needed: distributed control," *Proceedings of the International Conference on Very Large Data Bases* (ACM, 1133 Avenue of the Americas, New York, N.Y. 10036), 1975, p. 364-373; price $15.
7. Cashin, P. G., "Data base interworking," *Network Systems and Software* (Infotech Information Ltd., Nicholson House, Maidenhead, Berkshire, England), Report #24, 1975, price £50.
8. Acree, J., "Putting the principle into practice," *Data Systems* (Embankment House, 3 Caledonian Road, London N1 9DX, U.K.), February 1976, p. 10-12; price £1.25.
9. Cosell, B. P. et al, "An operational system for computer resource sharing," *Proceedings of Fifth ACM Symposium on Operating System Principles* (ACM, 1133 Avenue of the Americas, New York, N.Y. 10036) November 1975, price $16 prepaid.

SUBJECTS COVERED BY EDP ANALYZER IN PRIOR YEARS

1973 (Volume 11)

Number

1. The Emerging Computer Networks
2. Distributed Intelligence in Data Communications
3. Developments in Data Transmission
4. Computer Progress in Japan
5. A Structure for EDP Projects
6. The Cautious Path to a Data Base
7. Long Term Data Retention
8. In Your Future: Distributed Systems?
9. Computer Fraud and Embezzlement
10. The Psychology of Mixed Installations
11. The Effects of Charge-Back Policies
12. Protecting Valuable Data—Part 1

1974 (Volume 12)

Number

1. Protecting Valuable Data—Part 2
2. The Current Status of Data Management
3. Problem Areas in Data Management
4. Issues in Programming Management
5. The Search for Software Reliability
6. The Advent of Structured Programming
7. Charging for Computer Services
8. Structures for Future Systems
9. The Upgrading of Computer Operators
10. What's Happening with CODASYL-type DBMS?
11. The Data Dictionary/Directory Function
12. Improve the System Building Process

1975 (Volume 13)

Number

1. Progress Toward International Data Networks
2. Soon: Public Packet Switched Networks
3. The Internal Auditor and the Computer
4. Improvements in Man/Machine Interfacing
5. "Are We Doing the Right Things?"
6. "Are We Doing Things Right?"
7. "Do We Have the Right Resources?"
8. The Benefits of Standard Practices
9. Progress Toward Easier Programming
10. The New Interactive Search Systems
11. The Debate on Information Privacy: Part 1
12. The Debate on Information Privacy: Part 2

1976 (Volume 14)

Number

1. Planning for Multi-national Data Processing
2. Staff Training on the Multi-national Scene
3. Professionalism: Coming or Not?
4. Integrity and Security of Personal Data
5. APL and Decision Support Systems
6. Distributed Data Systems

(List of subjects prior to 1973 sent upon request)

PRICE SCHEDULE

The annual subscription price for EDP ANALYZER is $48. The two year price is $88 and the three year price is $120; postpaid surface delivery to the U.S., Canada, and Mexico. (Optional air mail delivery to Canada and Mexico available at extra cost.)

Subscriptions to other countries are: One year $60, two years, $112, and three years $156. These prices include AIR MAIL postage. All prices in U.S. dollars.

Attractive binders for holding 12 issues of EDP ANALYZER are available at $4.75. Californians please add 29¢ sales tax.

Because of the continuing demand for back issues, all previous reports are available. Price: $6 each (for U.S., Canada, and Mexico), and $7 elsewhere; includes air mail postage.

Reduced rates are in effect for multiple subscriptions and for multiple copies of back issues. Please write for rates.

Subscription agency orders limited to single copy, one-, two-, and three-year subscriptions only.

Send your order and check to:
EDP ANALYZER
Subscription Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-3233

Send editorial correspondence to:
EDP ANALYZER
Editorial Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-5900

Name_____

Company_____

Address_____

City, State, ZIP Code_____