An approach to high availability in high-transaction-rate systems

by R. C. Brooks

In business enterprises, it is important that high availability be maintained in the computer systems used by the enterprises, particularly in systems that have high transaction rates. A way of maintaining high availability is discussed, including the implementation that should be undertaken and the design issues involved. Some additional steps for further improvements are also offered.

The requirement for very high availability is a business need of most, if not all, on-line systems. The addition of high transaction rates adds to the complexity and the urgency for managing availability.

The experience gained in working with a client toward achieving an increase in availability is the basis for this paper. The client's installation serves as the setting for the discussion. The basic requirements for availability which were derived from the work done at this installation are stated as follows:

- The major on-line systems must be 100 percent available, 24 hours per day, seven days a week.
- The data processing department should be viewed as a power utility. It should supply service to users as power is supplied.
- The total system must be able to handle a high transaction rate with consistent performance.
- Availability must be maintained without affecting the implementation schedule.
- Capital and operational costs must be minimized while increasing the productivity of the end users and system owners.
- No business transaction can be lost.
- All business transactions must be completely processed in the correct day's cycle. The batch must not affect operation of the on-line systems.

This paper has two main parts. The first addresses the implementation needed to achieve higher availability for the current systems. The second part reviews the design issues involved as the environment changes, plus some additional steps which could be undertaken to improve availability and lower total operational costs.

As an overall summary, the approach taken covers the removal of unnecessary complexity, the avoidance of all single points of failure, the avoidance of predictable causes of error, increased use of automation within the operation of the system, and building applications for operating and managing the system.

Obviously we need to define availability. Any definition will vary from installation to installation but should include three components: reliability, performance, and usability. Specifically, the services of the system are accessible to the end user during the specified hours. The performance in terms of response time is consistent and within the specified requirement. The system should be seen by the end user as being "usable." This covers such issues as alternate ways to use the service based on skill level, help facilities, and confidence that the results of the transaction will be correct and safe. Such a definition should allow for components to be unavailable, such as the end user's terminal, but require that the affected end user have access through other access points, based on business need.

Implementation steps for high availability on the current system

The environment under consideration. In the installation which we are discussing, the total system is comprised of many applications, few of which are independent. These applications reside on the full range of computer possibilities—central large mainframes to distributed small processors, both local and worldwide. The end users include not only data processing professionals but users ranging from the fully trained to those with no training.

In March 1984, one of the client's IMS-based on-line systems was experiencing low availability. There had been a history of small outages, but two large failures occurred during that month. It took several days to recover from the effects of these major failures and return service to the end users.

Stress uncovers new problems as the load grows.

The design work and resulting implementation steps required to increase the availability of the system flow through to all other applications, regardless of type and location. Therefore, this on-line system is the focal point of the following discussions about work which has already been completed and is the base example used when considering the future.

It is assumed throughout this paper that the minimum base, such as trained personnel, is in place.

Installation management. As stated above, there had been a history of minor outages; the major failures were the tip of an iceberg. There were a large number of problems, of different levels of importance. The systems in place to manage these problems were under strain. They had not been upgraded to cope with greater complexities. As a result, problem resolution was overextended by the many people working on the problems, some of which were interrelated. To further complicate the situation, changes were being made to address the known problems and to provide for growth in new facilities and the number of users.

It was considered mandatory to enhance the "installation management" techniques. Of these, the major ones were problem, situation, and change management. This step resulted in an investment in additional manpower and the dedication of full-time managers to the areas of problem and change management.

This step has provided documentation of all known problems with the system and an indication of the importance of each one. This procedure allows some of the interrelationships to be investigated. The status of all resolution steps can be determined, and the responsible correction groups can be managed. All changes to the system can be tracked from one point of reference. The situation manager is able to determine the action or actions to be taken, control the cascading effect of a failure, and develop an action and resource plan to address a major failure.

Problem avoidance. We began to analyze the basic causes of the small failures. Hardware reliability was not a significant cause; most failures were caused by operation procedures and software. ("Software" here covers the application, system software, and controlling tables involved.) The problem had been aggravated by the requirement for growth, correction to current software, and upgrades to new versions of the current software.

There were also problems caused by stress. Stress uncovers new problems as the load grows. In such situations, for example, parts of the software are used for the first time, or the space of a buffer is exceeded. The requirement was to maintain performance and reliability as the loads increased. Therefore, we needed to avoid problems by predicting and then resolving them before they affected the reliability or performance of the production system.

Formal testing. We concluded that higher availability could be obtained by testing the effect of change, by increasing the quality of the total software package, and by validating the operating procedures. Formal test is a management procedure for the testing of all components in the production system. Test cases are built to cover both normal and abnormal conditions. Testing is completed only when all test cases are completed and a fixed number of faults are found and corrected. Normally two to three weeks

are allocated to each test phase, but elapsed time is not a measure of a successful test phase. It is estimated to take at least 18 months to gain the full benefits.

The methodology divides the testing into five phases. The first two phases are carried out by the formal software supply departments, either system programmers or application developers. They are responsible for phase one, the *program test* (module test). Phase two, the *system test*, is carried out by a separate test group within the application development department. The system programmers do their own system test.

The next three phases are carried out by a specialized group within operations. The third phase is called the *integration test* (or operational test). This is a test of the whole system which is to be the next production system. The objective is to run one full day's transactions in pseudo-production. Operational procedures are included in this test phase, both the normal ones and as many abnormal situations as can be created. An impact analysis is presented to change management on the potential production stability by the test team. The processor and I/O requirements are the same or greater than those of the production system, and the Teleprocessing Network Simulator (TPNS) is used as a driver system. We are beginning to use TPNS to drive the small application machines and include them in our integration test phase.

The fourth and fifth phases are *stress* and *performance*. The system is run under loads and with the network that is expected to be used within the next six months. These tests do not normally stop the new production system from being moved to production status. The intent is to allow time for the load-related problems to be resolved. They require in-depth analysis, followed by a combination of tuning, software changes, or the installation of new hardware. None of these can be achieved quickly.

As a sideline, the TPNS system is being used to address *ad hoc* questions. These come mainly from the application development and capacity planning departments.

The criteria for measuring the success of these formal test teams are the stability demonstrated by their component of the production system, plus the number and type of errors uncovered. Both the system test and integration test teams have uncovered significant software faults. They have been responsible for delaying implementation of new software to allow corrections to be made. In addition, the performance testing drives the tuning of the on-line system.

Operator training. Since the objective is high availability, the production system is no place for trainee operators. We have used the integration test as a training vehicle for the operators. This was added to the justification of this facility.

Effect of formal testing. The advantage of this approach has been an increased stability in the production system. Problems have been found before they occur in production.

There are also disadvantages with formal testing. A built-in delay is present between the time when a change is considered ready to leave the developers and when it successfully passes through the formal test system and is ready for use. The change "on demand" has moved to a scheduled cyclic installation of changes every few weeks, moving out to a monthly basis as we grow the implementation.

There is a need to accommodate emergency changes which may be necessitated by catastrophic failures requiring urgent correction. The use of this opening in the formal testing must be carefully controlled.

The change management has the responsibility of rejecting a change based on the test team's recommendation. Then the change does not happen until the next scheduled production system upgrade. This may cause a delay in the implementation schedule. The impact of the formal test requirement must be reduced to meet the business need to allow the rate of change to continue.

Formal planning. The built-in delays caused by formal testing and change management require that changes to the environment be planned for the next six months. We have allocated a six-week period extending from the exit of the software from its developers to when its availability for production can be planned.

These two time constraints have an effect upon the system owners and within the data processing department. This is basically a cultural change to the management in the area of planning and expectation. The desire for rapid unplanned change needs to be balanced with the need for high availability. The installation of quick changes to address one

need exposes the reliability of the whole system. What is required is an estimate of future needs and a commitment to accept the position that errors in the estimates may mean that a given change cannot be achieved. This covers the whole field from the ship dates of the developers to the number of end users, etc. So, as with testing, we need to move to formal planning.

The formal planning is based upon having Service Level Agreements (SLAS) between the operations department and the system owner. These SLAS are presented in end-user terms, e.g., terminals, business volumes, performance, and availability requirements. The operations department then develops these SLAS into changes to the current environment and obtains commitments from the other groups within the data processing department. The basic management reporting from the operations department, such as service level measurement, is based upon the SLA. If the service levels are not obtained, discussions are on a formal basis between operations and the system owner. If the actual resource usage or rate of change exceeds the estimates agreed on, operations should be able to estimate the effect. This procedure allows negotiation as to whether to proceed with the changes, or whether the increased load can be handled and what the potential impact may be. It is necessary for both the operations department and the system owner to believe that the SLAS are of mutual benefit and that the time invested in their creation, and commitments made, are of real value.

Unnecessary complexity and single points of failure. The reliability of the total system is the product of the reliability of each of the components. If a component is duplicated, the reliability of that component is the sum of the reliability of the two components. Hence, we have removed unnecessary components and added redundancy for the remaining components.

Host system design. The host system design was a contributing factor to low availability. The processor of the on-line system ran as the Job Entry Subsystem 3 (JES3) Global for the total system. Because of the shared direct access storage device (DASD), the Global Resource Serializer (GRS) was used. The Virtual Telecommunications Access Method (VTAM) for the on-line system processor had terminals that did not use its applications, but passed through via a channel-to-channel (CTC)-supported VTAM cross-domain.

We estimated that this unneeded sharing reduced the reliability of the on-line system by more than 1.5 percent. Recovery time was increased, since problem resolution data needed to be gathered to analyze some of these cross-system problems before service could be returned. Some of these interconnections could not be tested by our formal test system. Performance of the on-line system was affected by concurrent access by other jobs, in either system, to the same resources, e.g., disks. Some of the changes to the system required both systems to be upgraded concurrently; some required an initial program load (IPL) of both systems.

Access to data can be lost for a number of reasons.

This analysis led to the removal from the environment of the major on-line system all components or complexities that are not required to service this application. The other requirement identified was to build a level of redundancy for all the remaining components to bypass a failed component.

Current system design is aimed at simplicity and redundancy. The on-line system does not share any unnecessary resources with other systems. Environmental requirements such as air cooling, water, and power are provided so that failure in a component does not affect the on-line system, although other work may be affected. All access to the system is via the network or isolated sequential files for bulk data. No on-line system DASD is shared with other processors, except to allow for a backup processor to access the data of the on-line system. The design has removed all unnecessary CTC connections between processors. The system runs as its own JES3 Global, and the network control function is on a separate system.

Removal of data and software as single points of failure. We saw that data form a single point of failure, and loss of access means the application will not run. In this context, data include the application needs, the driver systems such as MVS, etc., i.e., all the data required by the application that are on DASD and tape.

Access to data can be lost for a number of reasons. We can lose access because of hardware failure. We can damage the data with the application or other support systems. We may not always be able to recover the data after a failure because of failure of the recovery system or operator error. We have had occurrences where the recovery system cannot rebuild the data base from the last backup copy. Also, we cannot wait for this rebuilding and still approach 100 percent availability.

Therefore, we implemented the following DASD subsystem to bypass all single points of failure involved in access to the data. All data for the total on-line system are duplicated on separate DASD. They are set out on DASD so that no failure of a disk drive, IBM 3880 Storage Control, channel, or processor will stop access to one of these copies. To avoid software corruption, only one copy at a time is accessed by the software. As static changes are made to the production DASD, a duplicate copy is also made. Each morning, the volatile data are copied onto the second copy.

A hardware failure of any type can be bypassed (an IPL may be required). On failure of the system, a normal recovery is started on the current production processor, and a cold start on the copied DASD is used on the other processor. If the normal recovery fails, then the cold-started system is used. The transactions of the failed system are recovered by Data Base Administration and loaded on the data base of the running system. The old data base is then discarded. The Information Management System (IMS) 1.3 duplicated data base will be used to reduce the need for restart caused by the failure of the DASD supporting the data base.

With this DASD design, we have also eliminated some of the single points of failure within the software. With one failed IMS system, we have returned service using another IMS system. IMS 2.1 will allow similar support for the application. The program may fail and that transaction may fail, but other transactions will be provided with a new copy of the program. This is very useful, since application programs should only fail in production because of unusual data conditions. With the removal of that transaction, others should continue to work correctly, and application service to these end users should not be stopped.

The Multiple Virtual Storage/Extended Architecture (MVS/XA) Extended Recovery Facility (XRF) will add to the availability, since it normally continues service after major failure of the hardware or software, with little end-user awareness of the failure. XRF is seen as complementary to the above. We have not achieved continuous service, but we have reduced the recovery time from the failure. The situation will improve with IMS 2.1 and XRF and other software/hardware facilities directed at continuous operation.

Removal of processor, network, and site single point of failure. For business survival, certain applications must continue after access loss to the production site. Access loss to the site can be caused by many factors such as fire or major damage to the main network trunks.

We are using multiprocessors, which reduces the likelihood of a total processor failure. Such a failure when running at high resource utilization may be seen as a total processor loss, since the resulting reduction in the processor resources will dramatically affect performance. We may need to move to another processor even though the current one is partially operational.

To address the need to bypass these single points of failure, we have developed the following implementation of the application mix. The applications have been categorized into four types, with one special case. They are

1. A1—quick on-site recovery and disaster site recovery within 2.5 hours
2. A2—quick on-site recovery and disaster site recovery within 24 hours
3. B—quick on-site recovery and no disaster site recovery
4. C—no on-site recovery but disaster site recovery within 24 hours
5. D—neither on-site recovery nor disaster site recovery

On-site recovery is the provision of resource duplication to allow the bypass of a failed resource, for example, the processors.

Disaster site recovery provides for resources in place at the second site to allow for loss of access to the current production site of that application. Service is returned without waiting for replacement facilities to be supplied and installed. For category A1, the data required to support the site switch are already on the DASD of the second site, allowing a 2.5-hour recovery. With category A2, the required data are loaded from stored tapes.

To reduce the costs, the lower-category applications lose their resource to the higher-priority work upon failure of that resource. The intent is to have the end-user management justify the category or categories for each application, so that they are aware of the costs involved in supporting their requirements. This is part of the SLA, built either at application design time or on the yearly review of the SLAS.

Category B at first sight appears as a strange combination. For some applications it is expensive to activate manual systems for a short failure; but a multiple-day activation of these same manual systems is less expensive than the costs involved in the provision of disaster recovery resources.

We have two sites, with the network switchable to either site. Both sites are connected by a network using land and microwave systems. Within each site, we have allocated processors on the basis of application category.

We are reviewing a possible extension to this plan. This extension involves two separate buildings on one site to guard against the threat of fire. They will be close enough that fall-back and disaster recovery resource capability are the same resource. The network can be fed into the site by alternate paths and potentially different means (e.g., land lines, microwave, or satellite).

Avoidance of transaction loss. We can lose business transactions as a result of corruption of the data base, fire in a site, or operator error.

We have defined that, for the end user, we own a transaction when that end user has received some form of acknowledgment that his/her involvement with the transaction is complete. Again, we have categorized the transactions into levels of importance to the business, and hence the level of backup, etc., required.

For important business transactions, the requirement is to have the transaction recorded in multiple ways and in multiple locations. The transactions are retrievable from these locations regardless of the type of failure within the system, and no outside resources or personnel are involved. Other categories of transactions get reduced levels of multiple recording.

Continuous operations. The requirement is for the on-line system to run 24 hours a day, seven days a week. Today there are a number of scheduled stops to the on-line system.

One major scheduled stop is to image-copy the data base each night. We have reduced this from over two hours to less than 20 minutes by an image copy to disk first. With IMS 1.3 this time reduces to zero.

The next major scheduled stop is a weekly re-IPL to allow changes to the system. We have reduced this time using formal testing and planning, but we need to be able to apply changes to the system on line. XRF will allow some maintenance to be done while service continues. However, more facilities are required in this area to produce a total solution.

Review of future trends and potential requirements of our installation

Let us now consider the design issues as we expand the usage of computer technology throughout the client's corporation to support the business needs. In this greater design area, there are major differences in the point of view about what the likely changes will be, what the likely design solutions are, and what implementations should actually be undertaken. In this area of prediction of needs, our requirements will also change as our view of both what is required and what technologies are available also changes.

A view of the potential changes. The major changes can be summarized as increasing the number of components that make up the data processing department system; more on-line systems, some with on-line update; increased interconnections between these applications; a broader range of end-user skills and demands; increased pressure to control the costs of data processing; and increased pressure to use data processing to increase the efficiency of the corporation. In more detail, the client will see potential changes as follows.

The current number of terminals in the corporation is less than 6000; this could grow by a factor of five or more. The variation in types of terminals will increase, ranging from television sets with keypads to other large processors. Their protocol and network intelligence will vary from simple ASCII standards to full function such as Systems Network Architecture/Systems Network Interconnection (SNA/SNI).

The end-user skills with computer applications will also vary over a range from personnel fully trained in data processing to those with little or no experience with computers. The investment by the data processing department in education and applications that are seen as user friendly will be significant.

More of the system owners of applications will demand very high availability, and many will demand availability 24 hours a day, seven days a week. The loads will be end-user driven, with little control of work schedules by the data processing department.

The end user will require more complex applications.

The end user will require more complex applications. There will be a move toward complementary processing, with the end user indifferent to where the task is processed or the data are located so long as performance is reasonable and the desired task is completed.

The performance of the data processing department will be a major contributor to the image of the corporation. Similarly, as knowledge grows, the system owners will demand more from the data processing department.

Today there are separate applications and departments for different services. The direction will be to allow the end users to view these systems as one. There will be increased connections between systems which must be transparent to the end user and without major investment in modifications to the current applications. Similarly, these connections must allow for the restructuring of the corporation at minimal cost.

The cost of the total system will be very high. There will need to be a balance between end-user service and the costs of that service. The method of passing on the true costs of that service will play a major role in the growth of technology usage.

The security of a system will be a major requirement. With its increasing number of components and interconnections, the system will be more complex and open to security problems. The security system must support different levels of security without undue impact on ease of use.

The transaction integrity must be maintained while allowing an increase in the total number of transactions in the total system and an increase in the number of types of transactions. The handling of queries on these transactions must be made more efficient to handle the increased load. A similar situation exists for such activities as the end-user help desks.

The network. High availability and high transaction rates involve network availability and complexity. One requirement is for redundancy within the network: cost control potentially limits this to major trunks only. Another requirement is to produce a single corporate network. The ideal would be one that carried all traffic. To allow cost efficiency, some traffic may be on dedicated and unique networks.

Currently, there is an ongoing discussion on "any-to-any" connection, how this is to be handled, and where it should be allowed. Another area of discussion is how to allow for the application interconnections and the large variety of terminals and services.

Any-to-any connection. A number of solutions exist for providing any-to-any connection. These solutions include private and public X.25 networks, carrying an X.25 network on an SNA base, and potentially with networks as extensions to SNA in a manner similar to those presented in recent Share/Guide papers.[1] The choice considered appropriate will determine the structure of the corporate network, if any-to-any connection is provided.

For end-user ease of use and total system security, any external user should have a restricted path through the network. Our current belief, which may change, is that the external applications associated with these terminals should only be allowed to reach a maximum of two entry points, with the system moving the transactions to the required application servers.

For internal users, there are three considerations. The first is that the end user should be independent of the current location of the source applications. The location needs to be found by the network. The second is that the end user should not be affected by single-server applications. (The analogy is the busy telephone; the phone can only handle one call at a time.) The third is overall costs; e.g., a nice-to-have facility for a group of end users may add to the total cost of the system. Potentially the bulk of the transactions are "many locations to a few application servers." These transactions must be carried by the

network cost-effectively, and these costs are not increased by the any-to-any users.

Boundary nodes. In the summary of potential changes were a number of items that increased the complexity of the total network. One potential method to reduce the impact on the network and the application servers is the creation of a boundary node function. The following is a summary of its functions and potential benefits.

The transport system should work with a single protocol.

- Hiding the network from the application server. To gain high availability of the total system, we need to remove unnecessary complexity from the application servers. To this end, we could hide the real network from the application servers. There is no requirement for the application servers to know in full detail the whole network of potential users. This transparency would reduce the impact of network changes and provide faster recovery from failure. Even the network control center system might not know about all terminals but just about the boundary nodes. The boundary node could pass through to the network control the details of the downstream terminals.
- Response assurance from the application server/timer. One of the requirements is to take some action if the application server is known to be unavailable or response has not been received. We should put a timer in the boundary node on the response of the application server to a transaction. If there is no response when the timer "pops" or if the boundary node is told that the application server has failed, the boundary node should respond, on the basis of terminal type, to the end user. This activity may vary from sending the transaction to another application server, acting itself as off-host, or canceling the transaction.
- Security. In the long term, potentially all transactions within the system, from the simple memo to business transactions, will be electronically signed, and the whole message made secure from unidentified change. Such measures may involve either the Data Encryption System (DES) or public key systems, or both. An end user will gain access to a given facility through the security system. The question is, "Where in the network/hosts is the best place for this to occur?" If we put it in the host application, we have the best security system for that application. If the user is moving between applications or is moved by an application to others, the user may be forced to pass the security check many times. This is not user friendly, and the application is complicated by the security system for each user. We could design into the boundary node the facility to build a security header for the terminal end user as he or she connects to the network. The header would be attached to each transaction from that user with a sunset facility. The application would then be independent of the end user, using this header as the access authorization. Some transactions, because of business impact, might require additional security validation.
- Protocol conversion. The transport system should work with a single protocol. Any conversions should be handled at the boundaries. For cost reasons, the volume application servers should use the transport system protocol. Every protocol conversion requires additional hardware and software, adds to the response time, and potentially reduces availability by adding another component between the end user and the application server.
- Handling of device dependencies. Complexity can be removed from the application servers if some of the device dependencies are removed and processed by the boundary node. As an example, the ASCII television-based Videotex screen uses graphics, data, and continuous image update. Only the data are supplied by the application server. Other intelligent terminals then work with the Videotex system, receiving only the required data and not the graphics or image update. The boundary node has made the application server independent of the terminal requirements.
- Transaction store and forward. Each transaction is given a priority for processing. At certain times, it may be desirable not to process low-priority transactions. The boundary node could be used to store these transactions for later processing and the end user apprised of their receipt.
- Multiple applications. The boundary node could be used to look at the transaction data content and determine the correct application server(s) for this request. This arrangement would allow the

end user to be independent of the location(s) of the application server(s).

- Broadcasts. At times, a number of end users must be informed of changes in the environment. The boundary node could be used to transmit these messages to each desired end user either in-flight or as they log on.
- Transaction integrity. The boundary node is a location that could be used to log certain transactions as they pass through the system. This could include both the inward transaction and the response. This data log could be used to answer queries from the end user, as well as to support transaction storage off site for integrity and accounting.

Transaction manager. We have built this pseudo-design for the boundary node independent of technology and costs, but the node is very intelligent. However, some of the facilities could be handled more cost-effectively by a central transaction manager. The central transaction manager could potentially service the following in a more cost-efficient manner:

- Building of the security header for the terminal type and location
- Interrogation of the transaction data to determine the application server(s)
- Storing and forwarding of transactions
- Concentration of the network for the application servers
- Broadcasting to selected terminals independent of the application servers
- Assurance of a response to end-user timer function
- Logging of transactions and the response of transactions for recovery, accounting, and audit reasons

Not all end users need pass through this transaction manager, but the end users who require a restricted path or multiple application servers would benefit. If the transaction manager uses the same technology as other processors in the main sites, then backup and disaster recovery do not require spare unusable processors. This would leave the protocol conversion, device dependencies, and probably the timer function in the boundary node.

Transaction rates and availability requirements for applications. We need to consider the effect on application design and processor placement caused by the growth of transaction rates. This requirement is sometimes known as horizontal growth. The second consideration is the requirement for up-to-date data and the increased complexity of on-line updating compared to memo-post techniques. The third area is a side issue of the usability of the application.

Horizontal growth. An application grows over time, due to the increasing number of end users, increasing loads, and demand for increased facilities. The application may exceed the size of the processor as it grows. It may exceed the capacity of a processor only during cyclic peaks, such as yearly. Hence, we want to move processors in and out as required (as we could DASD). These spare processors could be used for other application peaks or low-priority applications. The growth in facilities makes the applications more complex and, hence, a target for failures caused by this complexity.

The application may exceed the processor size as it grows.

Four options exist today for moving an application onto two or more processors. One is to build identical applications and split the data base and the transactions into separate workloads on separate processors. Another is to share the data base between two applications on two processors.

The third option is to split the application vertically with the data base. This requires us to undertake a review of the whole application and split the application by transaction types and data base requirements. Such an approach has the advantage of taking one large application and producing two smaller ones. Each smaller application would have a reduced load and less complexity. This would lead towards more stable applications. However, there might be some minor data base duplication.

The fourth option is a variation on the third. The application is split vertically, and the network is used to move the small number of transactions that require access to both data bases between the two applications servers.

Both the IBM 3084Q and the IBM 3090-400 computers can run as one or two processors and can be

combined to handle peak load. This allows use of the capacity during off-peak times for other low-priority applications. Alternatively, XRF can be used to change processors in-flight. With three processors of different sizes, the workload could be moved among them without bringing down the application that requires high availability.

The third and fourth options have many advantages over the first two. Growth in application facilities can be accommodated without increasing complexity within the application. It also allows us to avoid shared DASD or having two similar data bases and having to recombine the data for later processing along with all the implied complexity.

This approach to horizontal growth will allow growth in facilities within the applications and growth in the transaction rates. Total costs are reduced by handling cyclic peaks in high-priority applications by moving the processor capacity for low-priority applications in and out of a high-priority application.

This moving of transactions between applications, which is transparent to the end user, can address two other requirements. One is the restructuring of the corporation without causing major change in applications. The other is to allow the end user to work with the corporation as a single entity and not as individual departments. So, although we need single application servers to support very high transaction rates, we can take steps both to reduce the load to these application servers and to provide the operational flexibility required to control total costs.

On-line update versus memo post. The major on-line system uses the memo-post technique, with an overnight batch operation. The considerations are how to provide up-to-date details and handle the increasing transaction loads and the increasing batch load during the batch window, while maintaining high availability. There are two schools of thought: One is the on-line update; the other is the long-running trickle update to a separate full-detail data base.

On-line updating has the advantage of processing the transaction only once. It brings with it a longer pathlength per transaction and a more complex data base and application. Thus, it requires a larger processor for the on-line load. The application is more likely to fail due to the increased complexity of the application and data base, and potentially take longer to recover from a failure. This technique is of value for the lower-volume application servers.

The trickle update technique involves taking the transactions in large groups from the memo-post data base during the day and running them through the batch to update the separate, larger, more complex data base. This method has the advantage of protecting the investment in the memo-post system and adding little to its complexity. There is a need for enquiry and potentially for update access on line to the batch-updated data base, but failures in this system will not affect the major high-availability on-line transactions. The trickling of old transactions from the memo-post system will be stopped by MVS in periods of resource shortage.

An end user will view usability as part of availability.

In both cases there is still an overnight batch run, but since the majority of the work is done during the day, it is of less concern. The trickle update is superior because of its simplicity in a high-availability environment and the protection of investment.

This is an example of the vertical splitting of the application: Those functions that are not required to be processed together are not processed in the same application when high availability is desired.

Usability. An end user will view usability as part of availability. Many of the considerations have already been covered. The last major item is skill base. The first-time or irregular user should be provided with a friendly step-by-step process through the transaction. Second-level help is also desired. For the experienced user all of this assistance gets in the way, potentially producing frustration and inefficiency. An experienced user needs a user-controlled access with minimal overhead to create the desired transaction. The ability to move between these two modes should be provided by the application.

Balancing. The system in total will be processing a very large number of transactions per day. Not all

will complete successfully. The applications need to support on-line investigation of the status of a transaction and provide documentation as to its history. The applications must continue to operate in an out-of-balance condition and support indicators that show the location of the out-of-balance situation.

Operations. We need to consider the operations department, since its ability to run the complex environment efficiently and effectively has an effect upon availability and operating costs. We should concentrate on the removal of manual systems within operations.

What is covered under operations varies by installation. Let us define operations as including the operation of all central and remote processors and the network and providing the maintenance of all software and hardware, the help facilities to the end users, and the management support systems. These management support systems include problem, change, situation, capacity, and availability management.

The removal of manual systems implies automation and applications for operations. The application server must monitor the data flow from the whole environment. This includes the environmental issues, such as air and water temperature, power supply, and fire protection. The server should monitor the utilization of all components both local and remote, particularly zero or 100 percent: these are indicators of potential problems. It should be involved with the message flow from all components of the system. If it knows the required response, then it should take that action. If the response is unknown, it should highlight the occurrence to an operator. This area may be further enhanced by artificial intelligence or expert systems.

It is necessary for the application servers to provide information to the management of operations and others to allow the system to be run effectively.

Batch automation. Even systems that appear as totally on-line systems to the end user have a batch component. Batch processing involves building a job stream, gathering the needed input, running the jobs in the correct sequence, validating the successful completion, and taking any corrective action. This load tends to be repetitive, dividing into cyclic workloads. The work is often done at night, when it is more expensive and difficult to hire the required staff.

We need to run the batch jobs with a minimum of operator involvement. To achieve this, we need to remove the tape and printer output from the remote systems. Operation is performed from the central site. This includes the ability to remotely initial microprogram load (IMPL), IPL, power up and down these remote systems, and operate their master and user consoles. The operations application should be able to control centrally the batch jobs in these remote sites.

We should be able to have a similar level of control over the two central sites, except that tape and printer output may not be avoided. We should question the need for after-hours printing, and look toward automatic tape loading and unloading to remove operator involvement. Much of the batch control can be achieved by extensions to the operations planning and control (OPC) and by application additions in the remote systems.

We have a network control center processor that knows about the network and handles the network control function. We could add to this processor the operations application of batch job control for all local and remote systems.

Maintenance of software and data on remote systems. The requirement is to maintain the files on a large number and on different types of remote systems, including dial-in systems, that are transparent to the user. The number of systems could be in excess of 10 000 and include both internal and external systems. This application server should be distribution-list driven and be able to inform the central control function of unsuccessful updates. The uses of such a facility are numerous, from updating static data to retrieving data files from the remote system. Such a level of automation requires code in the remote system as well as in the host. The function described is an extension of the Distributed System Executive (DSX) program product.

Remote system and application recovery. We have provided a help desk which the end user may call regarding problems ranging from operator errors on the terminal to application server problems. The help desk personnel have to be able to view the network status to determine whether the user problem is being addressed by the network operators.

For the closed application remote systems, the help desk can use local versions of these systems to assist in addressing end-user requests. For the open appli-

cation systems, the help desk personnel have to access their system as the end user, plus view their files, change their software, and remotely IPL and power up or down the system. In some cases they should be able to view the status of their environment, such as the air-conditioning. Such a requirement is an extension of the Operator Communication Control Facility (OCCF)/Remote Operator Control Facility (ROCF) for IBM 4300 and DSX/HCF (Host Command Facility) for IBM 8100 computer systems.

Central site automation. In the large host complex, the rate at which the system informs the operator of its current status is very high, and will get higher as the processors get faster. We need to reduce the "noise" to the operator to allow the information to be of value. We should not stop the messages, since each may be an indicator of a problem and be required for problem resolution. The traditional approach is to divide the messages into groups and display or print them on separate consoles. Interrogation of these messages is still required, and a search of procedures is required to take action on the message.

There is a need for two types of automation. The first type is to automatically provide a standard response or responses to a system message, in which the response can be predicted from the message. The second type is for use when the automatic system cannot respond to the message. In this case it should supply the message and appropriate documentation on the console to assist the operator decision. This second type is an extension of the Info/Library function, with the automatic system supplying the text to a message or page from the operations procedures along with the system message.

The question is whether this automation should be inboard or outboard of the application processor, or both. Such activities as filtering the messages and printing or storing the information messages could be best done within the system. The same applies to the split of messages into functional areas. The other activities should be outboard. I have suggested that monitoring the environment and watching for indicators such as zero utilization are desirable and can only be performed outboard of the application system.

IBM Japan has produced a product, System Command Centre Facility (SCCF), that addresses many of these requirements. Briefly, it uses a Series/1 computer to act as a console to the application system, and uses an MVS system to reply to messages. It has the support for environmental monitoring and for the requests to customize the hardware to allow power up and IPL/IMPL of the processor and usage reading. The next stage is for the system to analyze the flow of messages and not each one in isolation so as to move from symptoms to causes. This would require an expert or artificial intelligence system.

The on-line systems will normally run without operators for a short period.

Remote operation of central site. There are two major sites, with a large support staff. Costs could be reduced by operating one site as a normal remote system.

The first consideration is the tape handling. Nothing can be done without the full automation of tape handling. The second area is printing, which could be consolidated into a central print shop.

We have the ability, with SCCF-type products, to monitor the environment at the second site. We need to provide local support for hardware maintenance to correct the failures. We can remove low-priority applications from the available processors to allow the others to run on the reduced resource. We should be able to operate the systems from one site by using the network to transmit the outboard system messages.

Now that we have our operators from most of the components of operations in one site, we need to protect ourselves from failures in our automation system. We have made the network available to either site to bypass any failures. We still need all the consoles or backup on the outboard pseudo-console to bypass the failure of the pseudo-console. We still need the resources for the operators, e.g., phones, desk, etc., in case access to the first site is lost.

The on-line systems will normally run without operators for a short period. Errors that occur during this period will remain unanswered. Hence our exposure to loss of the automatic operator for a short period is not high. If the savings in single-site operations are considered, the risk is potentially worth taking.

Effect of automation on operator skill levels. As a result of even limited automation, three types of operators are needed. The first group is the manual printer/tape operators. The second is the help desk group, assisting the end users. The third group consists of the monitors of messages from the automatic system. They are required to act when the automatic system cannot respond. These cases will normally involve the complex conditions generally not covered by procedures. Therefore, the skill level of these operators will need to be at least at a system programmer level. They will need the analytical skill to determine the required action, and the time to react will be very short. Indeed, after the investment in automation, the group of operators that is left in the machine room is both small in number and very skilled.

Management issues. With high transaction rates, rapid growth, and the need to provide high availability, the management system needs to move toward management by targets and exceptions. We need to provide the correct level of detail to each manager to ensure effective resource usage. This implies effective reporting and capacity planning.

The management structure should ensure that a single manager is responsible for availability and is given the authority to address the issues involved. This is similar to the delegation of responsibility for issues such as capacity and resource usage, security, change, problem, disaster capability, and audit management tasks. It is important that all these areas are not delegated by default to the operations manager.

Management reporting. There are a large number of areas on which basic reporting is required. These areas include the reliability of a component, its current spare capacity and predicted loads and peaks, its performance, the status of outstanding problems, action plans to address any of the above, and the personnel requirements to address each area as the environment changes.

No one manager will want to see all this detail. Therefore, we need to have a hierarchical structure of reporting, each level obtaining the detail to drive its scope of control and able to interact with its management team and the system owners.

To achieve this structure, targets for each component must be set, based upon the SLA between operations and the system owners. Graphics will enhance the overall picture. Exceptions to these targets will form the basis of the reports. Systems are required to allow any manager to see all of the detail of a particular problem.

Capacity planning. Capacity planning is divided into three groups: the short term, the tactical, and the strategic. The short term is to view yesterday and determine whether we can process for nine months ahead. The tactical is from nine months to two years, and the strategic is normally two to five years.

The major change in a rapidly changing environment is in the short term. For each component, we need to be sure that the resource can handle the changing workload. Because of the rate of change, this estimate should be done daily. Because of the volume, these estimates must be automated, and reporting made on exception, continuously comparing the actual usage of all resources with targets and predicted growth. Since any corrective action takes time, the estimates of component usage must predict six months ahead and highlight those resources with a potential excess or shortage.

This short-term planning must be the trigger for the tactical plan. Since the rate of growth can quickly change the capacity needs, a cyclic tactical plan may not be effective enough to provide for the efficient usage of components. The need is to provide time to react to major change. The tactical plan will have to be continuous and based on subsets of the total system for each study.

Strategic planning should be done yearly, as it is today.

Application development resource. The availability of the system and the resources used by a transaction are dependent upon the application development. We need to measure the quality of the application and its effect upon the availability as seen by the end user. From a cost viewpoint, we need to monitor the resources used by an application.

Operations needs to measure the failure rate of applications as seen by the end user and recommend,

on the basis of this data, additional work by the application supplier.

The capacity planning teams need to review the resources used and recommend additional application development based upon this study.

Costs of high availability

We must consider the costs involved in high-availability and high-transaction-rate systems. The cost curve is exponential, rising rapidly as we approach 100 percent availability. We cannot change the shape of the curve, but we can lower the curve itself.

Some application areas may justify high availability on business impact only.

All system owners will, if offered, accept 100 percent availability as a requirement. We need to take a pragmatic view of the 100 percent figure, since the cost of supplying this is higher than 99.9 percent availability or any lower amounts. Thus it is necessary to have a methodology that both allows the system owners to understand the costs involved and allows them to justify a given level of availability. This methodology would quickly remove some of the emotion from this subject. Some application areas may justify high availability on business impact only, with no other justification. It is sound business practice to enforce a cost-based reason; otherwise all applications will use the above reason.

Fortunately, many of the actions taken to improve availability for an application actually reduce the total costs of running the installation.

Moving to reporting by exception can lead to reduction in the management time required, as well as ensuring that the people and investments in capital items are directed toward the correct areas. It also can be used to ensure that the high-priority problems are addressed and that the correct level of tuning is done. Therefore, the investment in this large reporting system can be justified for reasons other than availability alone.

There is a need to automate the operations function, not only to increase availability, but also to reduce the cost of operations.

The formal testing of all production systems improves the quality of the production system. It also forces the resolution of problems that are found at a time when the development team is still in place. Corrections to applications become more expensive as the applications become older and the skill and knowledge base for use on an application decrease with time. Urgent corrections to production systems increase the chance that other errors will be added during the correction, thus adding to the cost.

The provision of disaster recovery for some applications is required for the survival of the business after a major failure. We have provided on-site resources as backup for the high-priority applications. Some are required only to provide availability. Throughout, the intent is to use the additional resources during normal operation for some productive work, even if it is of lower priority. Even having an uninterruptible power supply can be partially justified in that it avoids the costs normally involved in repairing components or bypassing them due to the effects on hardware of sudden power loss. We have avoided the normal requirement to use two processors, each running at 50 percent utilization, such that one processor can handle the total load. The design of our processor workload split is slightly less available than the design of two processors sharing the load. With XRF, we would gain the same benefit as with the two shared processors, and the second could be used for useful work. XRF requires approximately 10 percent of the second processor in normal operation.

The operations department is like the old story of the cobbler's children with their shoes in need of repair. Surrounded by computers, it uses manual systems to support its requirements. We have taken the network control methodology and expanded it to an application server for operations. At the same time, we have decreased the complexity of the application servers and increased their availability by having a better-run operations department.

Although the inclusion of a transaction manager and boundary node function in the network increases

the costs of the network, total cost may be reduced. Justification on business grounds in areas such as ease of operation, ease of use for the end user, and the image of the corporation can be used.

Another area, not included above except in passing, is load reduction. This reduction can take many forms, including transaction storage and forwarding, blocking of several transactions into one, bypassing intermediate nodes for high volumes, and encouraging people to work at off-peak times.

In a perfect world, each new investment in availability would be cost-justified. Traditionally, these benefits are the avoidance of the costs of the failures and the resultant recovery, such as human time. Other costs are the productivity loss for the internal end users, the loss of business, and the effect on the corporate image. Indeed, some investment is continually required for business survival.

The recent past has illustrated that some of the intangibles in this cost calculation are the effects of quality and internal end-user satisfaction. Quality, required for high availability, reduces total operational costs, thus supporting the required investment in resources and management time. Similarly, as we drive the usage of systems throughout the corporation, the costs are reduced if we supply a reliable and user-friendly system to end users. We can only gain the full benefits of a system if the end users use the system productively and do not avoid it, actively or passively.

The need for availability is real, to the extent that when a given corporation invests in the required resources, it is a function of the business objectives for the applications.

Cited reference

1. *Any to Any Connection in an SNA Network*, Share and Guide; available as ISC documents CSYS-15-85-01 and CSYS-15-85-02.

Russell C. Brooks *IBM Australia Limited, P.O. Box 1798Q, G.P.O., Melbourne, Victoria, 3001, Australia.* Mr. Brooks is a Senior Systems Engineer in the finance and retail branch office. He joined IBM as a programmer in 1967, after obtaining a B.Sc. in mathematics. Since then he has worked with a range of customers in various industry sectors. He has worked in professional roles that included programming and application design, IMS, and large systems, as well as spending several years in management positions. Mr. Brooks' current assignment is to provide strategic and architectural direction advice to IBM's customers.

Reprint Order No. G321-5254.