Modeling is a useful method to aid a planner in designing the interconnection of a number of systems for distributed data processing. In the implementation in this paper, a computer-based model for sites using CICS/VS is discussed. The model permits the system definition to be adjusted, taking into account such aspects as the number of sites, their interconnections, and workloads, so that a satisfactory configuration can be obtained.

Modeling distributed processing across multiple CICS/VS sites

by R. D. Acker and P. H. Seaman

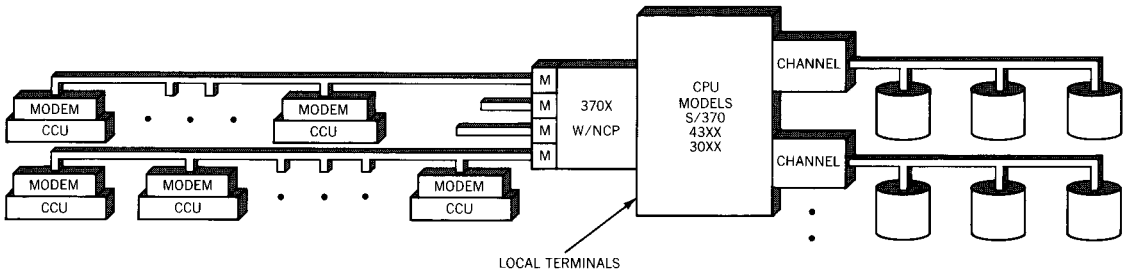
The subject of distributed data processing is one of intense current interest.[1] Many organizations are examining the idea to understand what benefits its implementation may provide in their circumstances. General discussions of the concept and possible benefits are found in References 2, 3, and 4.

One implementation of this concept occurs in CICS/VS (Customer Information Control System/Virtual Storage),[5] where several sites geographically distributed from one another can interact across communication links. A site in this context represents a separate processing system with its own CICS/VS region (or regions) and associated lines and terminals.

To assist in the design of such an assemblage of systems, a computer-based model has been developed. By use of this model, a planner can define a number of sites and their interconnections and assign workloads to the various sites. Upon execution of the model, resource utilizations and system responsiveness are estimated. The planner can then adjust the system definition in an iterative study until a satisfactory configuration is obtained.

This paper begins by discussing the model of a single-site CICS system. Following the initial discussion, the new features in CICS/VS that support multisite operation are introduced. The addition of representations of these features into the basic model is then

Figure 1 Single-site schematic



LINE CONTROL
SDLC/BSC/STSP

MODEMS
3863/3864/3865

CLUSTER CONTROL UNITS/TERMINALS
3600/3650/3790/2770/3767/3276

PROGRAMMING SYSTEMS
VTAM/VTAME/BTAM
DOS/VS, VS1, MVS
CICS/VS
VSAM/BDAM/ISAM
DL/1
BATCH BACKGROUND

DASD
3310
3330
3340/3344
3350
3370/3375
3380

described, after which the use of the enhanced model to study networks of CICS sites is covered. Finally, a usability feature of the model is discussed, which is intended to reduce input redundancy and facilitate changes in multiple run studies.

Overview of single-site model

The basic CICS/VS model, called ANCICSVS,[6] is an analytic design tool which is used by IBM account teams to estimate the performance of high-response, terminal-oriented systems based on the CICS/VS program product. As indicated by the descriptor "analytic," the model employs equations that are analytically solvable, rather than the technique of discrete simulation. This leads to solutions in a few seconds of execution time per model run. ANCICSVS is written in APL for use in an interactive environment. A number of simplifying assumptions, such as CICS operating in a dedicated environment, are not covered here, so users are advised to obtain further details to understand whether or not their particular environment may be represented.

The model encompasses both hardware and software considerations. (See Figure 1.) Internal tables contain average path lengths and operational characteristics that define numerous options and features of CICS and the several operating systems that support it. Queuing representations of the central processing unit (CPU), the direct access storage device subsystem (DASD), and the communication line subsystem (Lines) are included. Based on the system configuration and the workload description specified by the user, the model incorporates these major components into an integrated whole, reporting unit utilizations and system response times for specified actions.

The user input to the model may be classified in the following categories, which are listed in the order of entry:

- CPU parameters
- DASD configuration
- File definitions
- Transaction operations
- Line configuration
- Traffic rates

Each category except the CPU is repetitive, allowing multiple entities of each kind to be defined. The input is entered manually, an entity (record) at a time, for each category. The model provides range and consistency checks upon each record as it is entered, permitting immediate correction. Range checks relate to IBM hardware and CICS characteristics. Consistency checks relate to a proper association of the components of the system configuration.

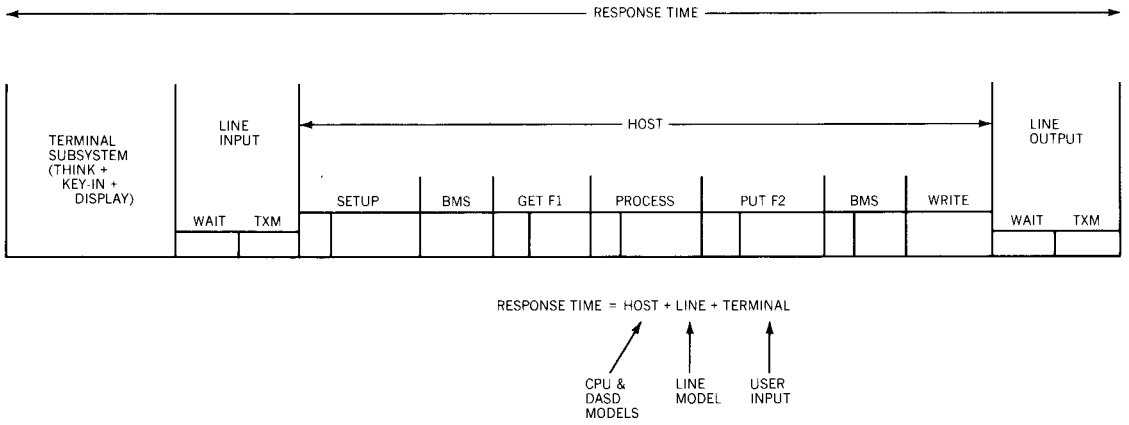
To understand the interaction among these model components, the transaction category is examined in more detail. Each type of transaction is defined, in addition to a set of parameters, by a sequence of operations called "macros." These operations resemble actual CICS macro-instructions, whether macro-level or command-level, as much as possible, so that a model transaction has a high correspondence to actual CICS programs that process transactions in a real system. The macro sequence for a typical model transaction might look like this:

BMSI		basic mapping services input
GUPD	MASTER	get data (w/update intent) from MASTER file
G	PAYMENT	get data from PAYMENT file
PROC	10000	process 10 000 instructions of application code
TSP	AUX	put data to auxiliary temporary storage file
PUPD	MASTER	update MASTER file
BMSO		basic mapping services output
TCW	150	write 150 characters to terminal
END		end transaction

The initial message that invokes this series of operations is defined in the basic transaction parameters and is not part of the series above. The final write macro sends a response back to the initiating terminal. The files referenced in this series have previously been defined in the File definition section, where all such concerns as access method, location, and size are specified. The rate at which the transaction is generated from various terminals in the system is specified later in the Traffic rate section.

One of the principal model outputs is transaction response time. It includes the interval of time from initial entry of the message at a

Figure 2 Transaction response time



terminal until the final response is received back at the terminal, and may be diagrammed as shown in Figure 2.

Note that the part marked "Host" corresponds to the elapsed time of a series of macros such as that specified for the transaction above. The additional intervals surrounding this host time represent time spent in actual line transmissions and terminal delays.

Two additional points should be emphasized here. First, although transactions with only one input and one output are shown, the model supports conversational transactions with multiple terminal interactions. Second, the model does not depend on the sequence in which the user specifies the macros. Usually the macros are specified in execution order for clarity but any order of the same macros produces the same answers. This is a result of the analytic nature of ANCICSVS, which represents the bulk activity but glosses over much of the detailed structure. This insensitivity to macro ordering presents a problem for modeling distributed systems which, as we shall see, is overcome by the development of transaction subroutines.

To understand how transaction response time is estimated, the model calculation scheme is next examined. The calculation scheme consists of the following steps:

- Decomposition macro scan
- DASD model
- CPU model
- Recomposition macro scan for host response
- Line model
- Output reports

During the decomposition scan, all the macros are examined in conjunction with the specified traffic rates of their associated trans-

actions, and tables of elementary actions are built up pertaining to the three submodels for DASD, CPU, and Lines. The DASD and CPU submodels are then exercised, producing raw waiting times and unit utilizations.

A second scan of the macros then reconstitutes all the individual pieces, including internally stored path lengths, calculated service times, and queuing delays, into a host response time for each transaction. The line model is computed last because there are some minor dependencies on the host response.

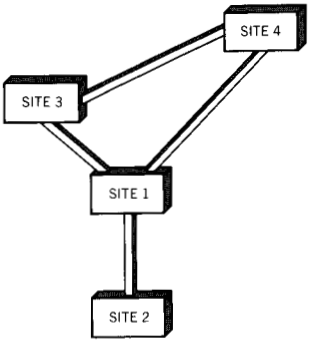
The output section gathers all the results and reports the estimated transaction response times and the underlying unit performances. An analysis is made of these unit operations, and any that exceed specified limits, such as DASD utilization over 70 percent, are highlighted for further investigation.

Description of CICS/VS support for multiple sites

Using this model of a single CICS site as a starting point, we expanded it to include multiple interacting CICS sites. To explain how this expansion was done, it is first necessary to understand the CICS/VS facility of Intersystem Communication (ISC). The following is a simplified view of ISC operation, touching only briefly on many of the necessary details that permit it to work successfully, such as file integrity and recovery features. Other features are ignored completely, such as the "NOCHECK" option. For a more complete description, see Reference 5.

The ISC facility implements communication between two or more independent CICS sites and may be diagrammed as shown in Figure 3. This indicates that the sites have a peer relationship with one another.[7] Although one site may be designated as the central control site, there is nothing inherent in the structure of CICS to require this. Further, there is normally a single level of communication. For example, Site 2 in Figure 3 can talk to Site 4 by passing through Site 1, but if there is to be regular traffic between the two, a dedicated link between them is preferred. (This preference is reflected in the model by not representing "pass-through.")

Figure 3 Multisite network



The communication between CICS sites is accomplished by means of two new entities—the session and the mirror transaction.

A *session* is a concept defined in Systems Network Architecture[8] and implemented in ACF/VTAM (Advanced Communications Function for the Virtual Telecommunications Access Method).[9] It is the logical means by which VTAM in one "node" (or CPU) establishes contact with VTAM in another node within its environment and maintains control of the information flowing between those two

nodes. There may be several sessions between any two VTAM nodes, but a single user employs only one of them at a time. In our CICS modeling world, a VTAM node is synonymous with a CICS site, and the session user is a CICS transaction.

Underlying the session is the physical communication link by which the required messages are actually transmitted back and forth. This link is precisely the same as the normal communication link between terminals and any site processor, except the receiver is now another CPU rather than a terminal.

A *mirror transaction* is the vehicle by which CICS interprets and acts upon the remote requests that are presented to it by VTAM. After a session is established between two CICS systems, the invoking transaction in the "source" site requests operations to be carried out on its behalf in the "target" site. The target CICS system may then create a special transaction, called a "mirror," to carry out the request locally and report the results. Although these mirrors produce activity against local resources like any other transaction, they receive their driving impetus from outside the local system. Therefore, the performance of an interacting network of systems must be considered as a whole and cannot be determined by examining each local system by itself.

Types of ISC operation

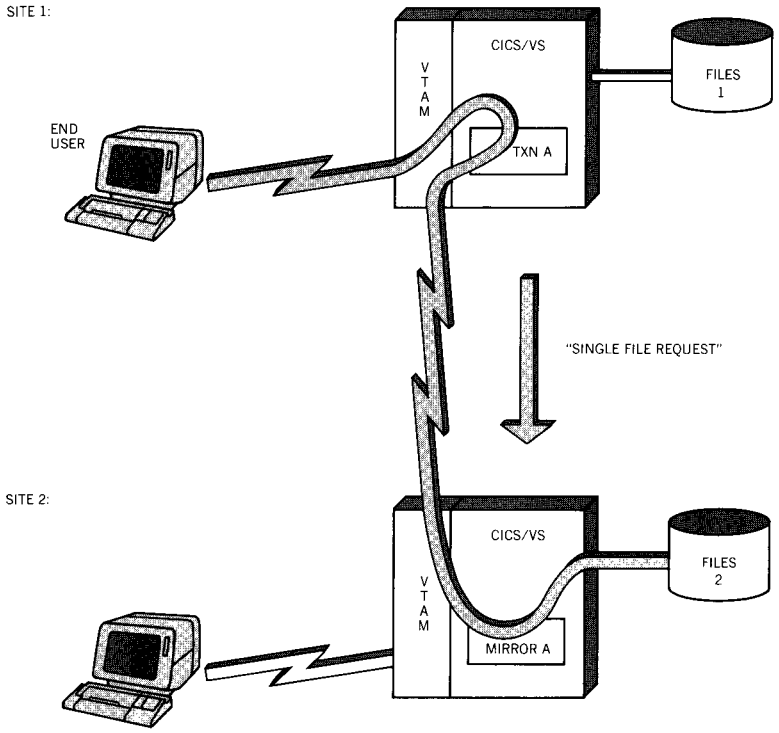
In the use of the ISC facility, there are three basic ways in which a CICS transaction in one site can interact with a remote CICS site: user function shipping, asynchronous transaction processing, and distributed transaction processing. We shall define these three operations here and later examine their impact on system performance using the extended modeling capabilities.

user function shipping

User function shipping involves a single data access to a remote file as shown in Figure 4. If a transaction in Site 1 requires data from a file resident at Site 2, it can be obtained via function shipping. The main requirement is that the existence of the file at Site 2 must be defined in the system file tables at Site 1.

Actual CICS code is not affected by a file's location. Thus, if a file that was at Site 1 is moved to Site 2 and this fact is duly recorded in the system file tables, actual transaction code remains unchanged. When a file request is executed in Site 1 requiring access to this file, CICS will recognize the relocation, and an ISC operation will be initiated instead of a simple file access. This operation will proceed as follows: First a session is obtained from VTAM, establishing connection to the proper CICS system, or target site. The file request is transmitted to the target and incorporated into a mirror transaction assigned to handle the request. The mirror is then dispatched as a normal

Figure 4 User function shipping



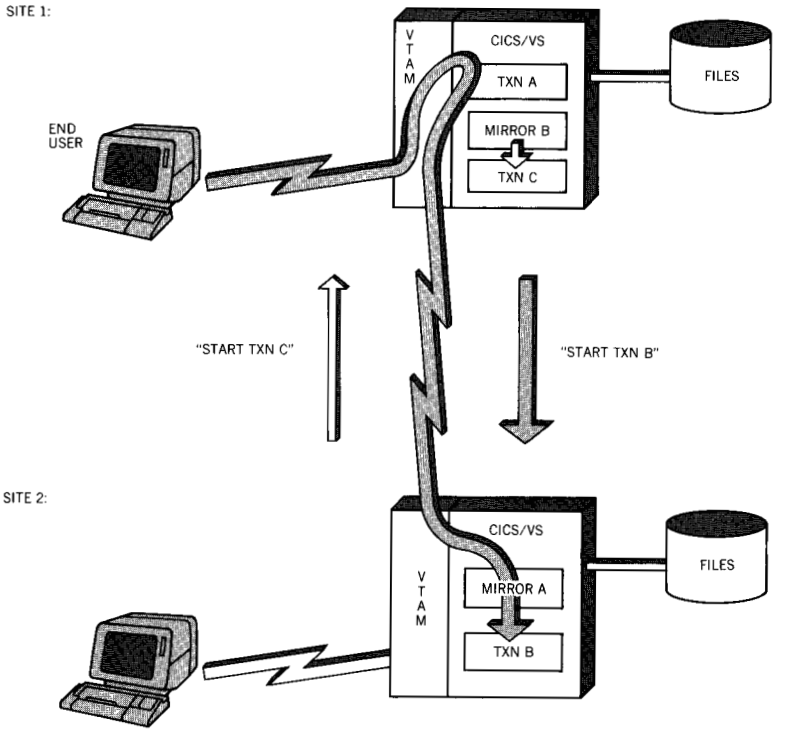
transaction accessing the required file residing in Site 2. The requested data is then packaged by VTAM and transmitted back to Site 1. If all error checks are satisfied, the mirror in Site 2 may be dropped and the session terminated.

However, to maintain file integrity when updating occurs, the session is often held until a synchronization point is reached. At this time, all file updates since the last "sync point" are definitely committed and the operation logged. This function prolongs the session time considerably and must be accounted for in any model.

asynchronous transaction processing

Asynchronous transaction processing starts a named transaction in a remote site from a local transaction, as shown in Figure 5. Once initiated, the new transaction proceeds asynchronously relative to its initiator. This is useful when a number of operations need to be carried out at the remote site, and the response time requirement is not of primary concern. The complete remote transaction operation must be defined in the target system. Only the transaction name need be transmitted with the start operation to initiate the remote transaction, although additional data is usually included to particularize the operation. A session is acquired to transmit the request, and a mirror

Figure 5 Asynchronous transaction processing



is set up to handle the operation. However, once the specified transaction is started in the target system, both the mirror and the session are dropped, and the original transaction proceeds independently of the remote transaction.

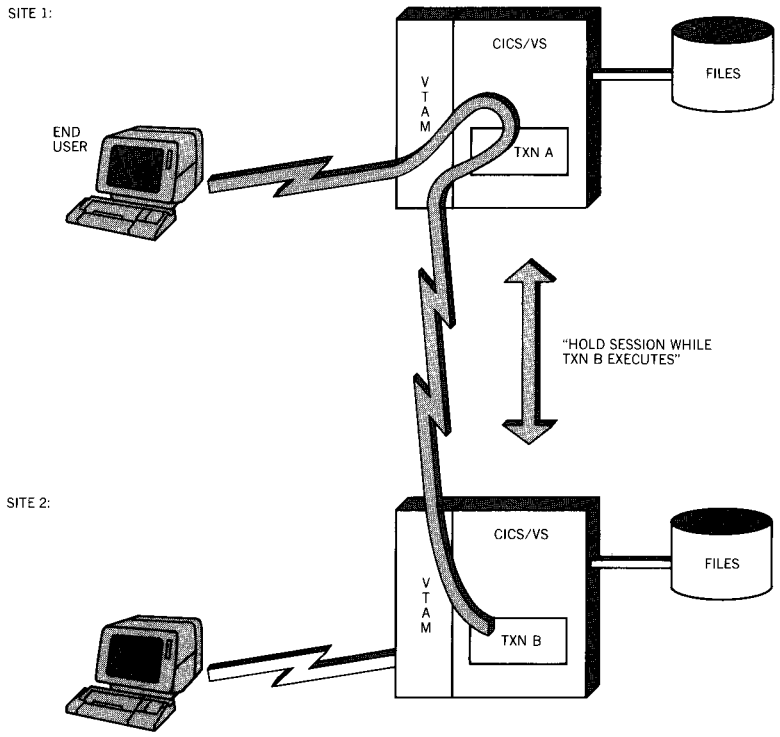
Very often the remote transaction, having completed its operation, will start a third transaction at the original site. This procedure is indicated in Figure 5 by TXN B starting TXN C via a mirror and passing it final results. The third transaction, in turn, will transmit the results to the terminal originating the request, thus completing the request turnaround. Such an operation is very efficient in using session and mirror resources. However, system integrity is loosely controlled, which may not be tolerable in many cases.

distributed transaction processing

Distributed transaction processing (DTP) is the third method of interaction between CICS systems and is designed to avoid the inefficiencies of function shipping, when employed for multiple operations, as well as to alleviate the integrity problems in asynchronous processing. It is illustrated in Figure 6. It should be noted that no mirrors are associated with DTP.

As in the case of asynchronous starts, a transaction in Site 1 calls a second transaction in Site 2 to carry out some work. With DTP,

Figure 6 Distributed transaction processing



however, the session is held, and the delegated work is carried out synchronously while the calling transaction waits. There may be several transmissions back and forth across the session link while the coupled transactions resolve the total problem. Although the session may be held for a relatively long time and large amounts of data may be transmitted over the link, DTP is often the most efficient method of using ISC. The major cost lies not in resource consumption but in application programming.

Representation of ISC in the model

Now let us see how these ISC operations have been incorporated into the basic CICS model. A primary objective of this effort was to disturb the original model structure as little as possible. Thus, the model user not employing ISC was to experience minimal impact. Also, migration of an existing pre-ISC model to one utilizing ISC would be a simple exercise.

The new model features include the following:

- Multiple sites
- Mirror transactions

Table 1 Ways to organize input

Input Method A	*Input Method B*
Define input categories,	For each site successively,
CPUs, all sites	CPU, for this site
DASD, all sites	DASD, for this site
Files, all sites	Files, for this site
Transactions, all sites	Transactions, for this site
Lines, all sites	Lines, for this site
Traffic, all sites	Traffic, for this site
If multiple sites,	If multiple sites,
Sessions	Sessions

- Concurrent solution for all sites
- Session/link model
- CALL and START macros

multiple sites

Multiple sites are the first consideration. When the input categories were examined, there appeared to be two ways to organize the input—(A) add a site identifier explicitly to each category definition and define all entities in each category together, or (B) define complete sites, one at a time, with the site identifier specified once at the beginning of each site grouping and thereafter implied for all categories within the site. The two methods are compared in Table 1.

In either case the same array structure results internally. The ISC arrays will differ from the basic CICS model arrays only in the extra field per record specifying the site to which it belongs. Internal calculations and output reporting require this extra field for purposes of description and grouping. Input Method B was implemented on the premise that it would be the more natural order for the user. Also, Method B has less impact on users with one site only, who constitute a majority.

After the multiple site definitions are complete, a new data category, called Session Configuration, is included to define the sessions available between the sites. This category encompasses the characteristics of the sessions and their associated communication links.

mirror transactions

Mirror transactions are the next feature to be considered. Since these are created internally by CICS, it seems reasonable to have them automatically generated by the model. It does this whenever a reference is encountered in the transaction macros of one site referring to a file or transaction in another site. The proper path lengths associated with the mirror operation, along with the requested operation itself, are inserted in the mirror, and the rate of the invoking macro is applied. Following this, the mirror appears to the model like any other transaction.

In the case of asynchronous starts, however, the mirror transaction plays a distinctly transient role, and for DTP one is not set up at all. For purposes of the model in these cases, the overheads associated with the ISC operation were merged together with the called transaction, so that a single remote operation, including mirror support if any, is reported. The rate for each called transaction is the sum of all of the rates of the transactions which call it, wherever the calling transactions reside in the system.

concurrent solution

After the mirror operations are analyzed and the rates of occurrence are determined in their respective sites in the network, the sites are effectively decoupled as far as calculation is concerned. The transaction response times for each site may then be solved independently of the other sites. The calculation scheme discussed earlier for the single-site case has been extended to cover the multiple-site case as follows:

- Global decomposition macro scan
- DASD models
- CPU models
- Global recomposition macro scan for host response without session times
- Session/link model
- Line models
- Output reports per site

The decomposition macro scan was made global, examining the macros for all transactions at all sites. This produces input for the multiple DASD and CPU queuing models which are solved independently, one at a time. A global recomposition scan then sums together the pieces of each transaction to obtain a host response time, excluding at this point any session times. A session/link model, described below, is then invoked to compute the delay times across the defined sessions in the system. Finally, the individual Line models are calculated, and the output reports are generated for each site, combining and summarizing the various calculated components.

session/link model

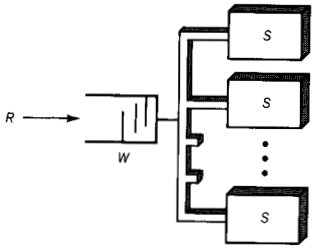
The session/link model is an essential element of the multiple-site calculation scheme. Like the three existing queuing models—for CPU, DASD, and Lines, the session/link model derives its basic input from the global decomposition scan of the input macros which now, in addition to summarizing input to the other queuing models, accumulates all the session activity between sites.

The link portion of the model employs the same algorithm as the basic line model, using appropriate data lengths and protocol overheads to represent the ISC transmissions.

The session portion of the model is represented by a simple multiserver queue, indicated in Figure 7. The actual system is much more

complex than this, but with all the many variables possibly affecting the result and very little measurement data available, a more detailed representation did not appear possible at this time. The utilization limits are accurately shown in any case.

Figure 7 Multiqueue model of a session



The multiple sessions between any two sites are represented by M parallel servers with a single waiting line. With S standing for average session hold (or service) time, and R for the rate of session access, the average waiting time W for a session is given by the standard formula for the multiserver queue:[10]

$$W = f(R, S, M)$$

If this expression is applied to each pair of sites between which sessions are defined, the session delay times, $W + S$, are obtained for all ISC activity. These session delays are then accumulated by transaction and added to the previously calculated session-free transaction responses, as noted above.

The central calculation in the session model is session hold time, that is, the average duration that each session is held to service an ISC request. It consists of three pieces, expressed by

S = link transmissions + mirror response + recovery support

The first piece comprises the transmission times of the request back and forth over the link, which are computed in the link portion of the model mentioned above. The second piece represents the time required by the transaction initiated by the session, whether a mirror or DTP type. This response time is available from the session-free transaction calculations. The third piece accounts for the additional time the session is held to maintain system integrity.

For example, when updating a "protected" file, the session is held until a "sync point" is reached, usually involving several operations in the calling transaction beyond the original ISC request. (See discussion below on CALL macro for a technique to calculate this time.) Often, the total span for the calling transaction is used to obtain a rough but useful estimate of this time.

The extended session time may also include several ISC calls to the same remote site. This is possible in the model because the calls, by invoking the "protect" option, all use the same session as if it were one long service with no intermediate session waits to be calculated. In fact, this mode of operation is often exploited in reality. If there are several ISC calls in the transaction, the session is acquired once only at the beginning and held for the duration of the transaction, thus eliminating separate session acquisitions and their associated waits.

Using the resulting session hold times to calculate the session delays, we may complete the response time calculations for transactions including session activity. Fortunately, at this time, there is no

requirement to model multilevel mirror operations, where mirrors themselves generate ISC calls and further mirrors. Time-consuming calculations involving iterative convergence would be required to handle such a recursive environment. Although the ISC facility handles multilevel mirrors, a situation sometimes referred to as a "daisy-chain," system designers are discouraged from using the feature; hence, the model does not include it.

CALL and START macros

The CALL and START macros, which model entities representing similar operations in the real system, are major contributors to activity generated for the mirror transactions and the session model. By embedding these macros in a modeled transaction, DTP and asynchronous processing may be represented directly. The principal argument of both types of macro is the name of the transaction to be invoked. The rate associated with the calling macro is accumulated by the called transaction, and session activity is set up. From that point on, the invoked transactions operate like any other transaction.

During the second scan reconstituting transaction response times, a significant difference between the START and CALL macros shows up.

The START macro represents asynchronous operation. Thus, the total time for the macro is the time required to perform an ISC start operation, and no more. The timing of whatever is started has no effect on the initiating transaction. This is illustrated in the following example:

Site 1:	TXN A	—	
		START TXN B	(response time for TXN A,
		—	*not* including TXN B)
Site 2:	TXN B	—	
		—	(response time for TXN B)

In the case of the CALL macro, however, the total macro time includes not only the ISC overhead for the call operation but the total response time for the called transaction as well. And if the called transaction contains CALLs, the whole operation is recursive, as shown:

TXN A	—	
	—	
	—	(response time for TXN A,
	CALL TXN B	including TXN B + TXN C)
TXN B	—	
	—	(response time for TXN B,
	CALL TXN C	including TXN C)
TXN C	—	(response time for TXN C)
	—	

This inclusive relationship is incorporated in the model. It should not be confused with the lack of recursion in the session model. Although CALL recursion is employed in the session model, session recursion stops at a single level. The reason is that higher-level embedded sessions involve a complex queuing calculation at each step, whereas call recursion is a simple linear accumulation of partial sums that are already calculated.

When two transactions are in the same site, the CALL macro may be used in a "subroutine" mode to isolate portions of one transaction and report the duration of the activity independently in a second subtransaction. A typical application of this mode is to estimate session hold times elongated by recovery operations—from request to "sync point" release. This latter portion of the operation may be defined separately as a subtransaction and called from the main transaction. When used in this mode, a special transaction indicator is set to inhibit the accumulation of otherwise normal CICS dispatching overheads.

Use of the ISC features in the model

Given these model features, we can examine the three types of ISC operation and see what the performance implications are.

user function shipping

User function shipping may be represented by Transaction A in Site 1, which occasionally (once out of 100 executions) references a file located in Site 2:

```
Site 1:    TXN A    —
                    —
                    G FF2, .01
                    —
                    —

Site 2:    FF2 file definition
```

The model confirms experience with actual systems that this type of operation is quite costly both in terms of resource usage and individual responsiveness. Although the path length to support the access now occurs at the remote site, the ISC path length, including VTAM operation, added to control the transfer of data, is roughly twice this amount, incurred at both sites. And whereas a local access may respond in 0.02 second, a remote access using ISC may take upwards of 2 seconds. Function shipping should be limited to low-volume, exceptional cases. Frequent reference to data dispersed in scattered locations is impractical using this approach.

The major advantage to this mode of operation is that no additional application code is required in the target system, such as is required

for asynchronous processing and DTP. The operation of locating the file and handling the ISC request is carried out automatically by CICS.

asynchronous transaction processing

Asynchronous transaction processing is a possibility where a number of functions are to be performed at the remote site and the immediacy of response is not critical. Policy updating for insurance company branch offices is a typical application shown below.

Site 1 represents the home office, with all the policy files and the program for TXN B, which performs the required updating. Site 2 represents a branch office and includes the programs for two transaction types, A and C. TXN A generates new data, initiates the update operation, and transmits the data to be incorporated in the update. TXN C, started by the successful conclusion of TXN B, serves as confirmation to the branch that the update has been completed and updates the branch office records.

```
HOME (Site 1): Policy files
    TXN B:   —
             —                      (update policy files)
             —
             START TXN C, 500       (transmit results back)
BRANCH (Site 2):
    TXN A:   —                      (gather data for updating)
             —
             START TXN B, 250       (transmit data for updating)
    TXN C:   —                      (update branch files)
             —                      (confirm completion)
```

Traffic rate: TXN A–300/hr

This mode of operation is shown by the model to be quite efficient. The ISC/VTAM path length employed is more or less equivalent to the overhead of a single function shipping request, but in this case it is charged only once against the activity of the whole transaction started, which may include a number of file references. Session usage is low because only one interaction is required to initiate a series of actions. This supposes the proper error recovery procedures are included; for example, what happens when TXN B fails and TXN C is never started?

Further studies are possible to determine the proper functions and amount of work to be processed by the branches versus the home office, based on the sizes and capabilities of their respective processors.

distributed transaction processing

Distributed transaction processing is a prime candidate when on-line processing is required with full control ensuring that all of several actions are completed, and the whole operation must be expedited in

the most efficient manner. An application might occur in commercial banking where a branch refreshes active records periodically from a central master file.

```
CENTRAL (Site 1): Master files
    TXN B:   —                          (extract data from master
             —                          files and transmit to
                                        branch)
             —

BRANCH (Site 2):
    TXN A:   —
             CALL TXN B, 500
             —
             —

    Traffic rate: TXN A–200/hr
```

Here the response time for TXN A at Site 2 includes the time required to execute TXN B, which in turn depends on how responsive the processing is in Site 1, considering all the other activity taking place there. As with function shipping, this type of operation is expensive unless prorated over a number of file references. For example, if three or more remote accesses are required, it is probably more efficient to use DTP rather than user function shipping. However, the critical resource in this form of intercommunication is likely to be the session itself, which is tied up during the whole operation of the remote transaction.

With the model, alternate possibilities can be studied to compare performance. This will permit planners to choose among the options with greater knowledge of the consequences and confidence that the final design will perform as specified.

Repetitive data handling in the model

Computer systems that are distributed over multiple sites tend to be repetitive by their very nature. For example, the processors that support a group of branch offices tend to be copies of one another, with the same transaction types being processed against similar files, differing mainly in specific record content or in frequency of use. Because of this similarity between sites, much of the input data required to represent such a system is repetitive.

Another related consequence of modeling a distributed system is the need to move entities about, attempting to balance workloads with resources available. Because of the site-oriented input structure (Input Method B discussed above), this move operation is not as simple as it might have been for Method A.

To ease the input and editing burden in these cases, the concept of "generic input" is required. This means that the user enters the basic

data into the model only once, and thereafter, by means of simple commands, is able to move the data about, reproduce it, and edit it as required.

The categories of input that can usefully employ generic data are

- File definitions
- Transaction operations
- Entire sites

Because they may reference generically defined entities, macros referring to files, as well as START and CALL macros, must be included in the processing of the generic data.

An important objective of generic processing is the automatic generation of distinct names for entities of a given type, without undue user involvement. To this end, four types of names are recognized—global, particular, local, and distributed. These name types may be defined as follows:

Global—complete, remaining unchanged as the entity is moved about.

Particular—complete but defined by the move command, not an unalterable part of the entity.

Local—a prefix only, completed by the automatic appendage of the site identifier, which is changed as the entity is moved from site to site.

Distributed—a prefix only, similar to a local name, with an appendage specifying a distribution of site identifiers.

A global name might be used for a specific file that is to be moved from one site to another while comparing the effect of its location on overall system performance. A particular name could be useful when copying a transaction, with all its attendant macros, from one site to another and renaming it. A local name is extremely useful when a similar entity occurs at nearly every site, such as a Customer file. A distributed name is used in macro references to address an array of local names: for instance, a reference at Site 1 to Customer file, resulting in 90 percent of the accesses to the file residing at Site 1 and 10 percent to the file at Site 2, with a comparable distribution at Site 2.

The copy and move commands, utilizing these new varieties of names, comprise a new input category, embracing the previous categories in a meta-language concerning the model itself.

ANCICSVS requires that each entity within an input category have a unique name even though it may be similar to a name at a different

site. Therefore, the replication of any entity requires that a new name be specified or that the name be designated as a local name. This type of name implies that there is a root part of the name subscripted by the site identifier. Replication of an entity with a local name implies automatic change of the site identifier.

Macros referencing data files or transactions, whether or not included in a transaction being copied from one site to another, require a further consideration. The names of the entity referenced in the macro can be global, local, or distributed. The third type of name implies that the macro employing it will be repeated within its transaction so that the referenced name can take on all specific local names for which sites have been defined. This type of replication is carried out automatically after the site-to-site copying of entities has been accomplished.

The generic technique minimizes original data entry and expands it across sites to meet a total input data requirement. This technique includes the flexibility of editing the data after expansion, so that small amounts of nonrepetitive information may be incorporated.

Conclusion

The systems planner can now represent a total CICS solution, incorporating multiple sites, using the new features of ANCICSVS that were described. With the ease of changing resource allocations and the rapid execution of the model, many variations can be studied in a short time. The results of these runs can then be weighed with other business considerations to provide the basis for informed executive decisions.

CITED REFERENCES

1. *Proceedings: The 2nd International Conference of Distributed Computing Systems,* sponsored by INRIA (France), IEEE (New York), *et al.,* Paris, France (April 8–10, 1981).
2. A. L. Scherr, "Distributed data processing," *IBM Systems Journal* **17,** No. 4, 324–343 (1978).
3. H. Lorin, "Distributed processing: An assessment," *IBM Systems Journal* **18,** No. 4, 582–603 (1979).
4. J. R. Buchanan and R. G. Linowes, "Understanding distributed data processing," *Harvard Business Review,* 143–153 (July–August 1980).
5. *Customer Information Control System/Virtual Storage: General Information Manual,* GC33-0155, IBM Corporation; available through IBM branch offices.
6. P. H. Seaman, "Modeling considerations for predicting performance of CICS/VS systems," *IBM Systems Journal* **19,** No. 1, 68–80 (1980).
7. R. J. Cypser, *Communications Architecture for Distributed Systems,* Addison-Wesley Publishing Co., Reading, MA (1978).
8. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15,** No. 1, 4–23 (1976).
9. *Advanced Communications Function for VTAM: General Information Manual,* GC38-0254, IBM Corporation; available through IBM branch offices.

10. L. Kleinrock, *Queueing Systems,* Vol. I, John Wiley & Sons, Inc., New York (1975), p. 102.

The authors are located at the IBM Aids Development Center, South Road, Poughkeepsie, NY 12602.

Reprint Order No. G321-5177.