Customer Information Control System—An evolving system facility

by B. M. Yelavich

Presented is an overview of the present CICS architecture. Discussed is the evolution of that original design as a transaction management system that accommodates data base management, operating systems, and input and output devices as well as hardware of increasing numbers and complexity. User needs past and present are analyzed with a view toward understanding how CICS might evolve in the future.

As of June 1985, the Customer Information Control System (CICS) entered its seventeenth year as an IBM program product. This paper examines from a systems perspective the CICS program product as it has evolved during much of that period. This paper is dedicated to the users of data processing who determined the systems requirements, to the developers of systems software who fulfilled those requirements, and to the systems engineers who designed and implemented the system. Systems engineering and such products as CICS have evolved together, adapting to the needs of information systems and applying hardware and software technologies as they became available.

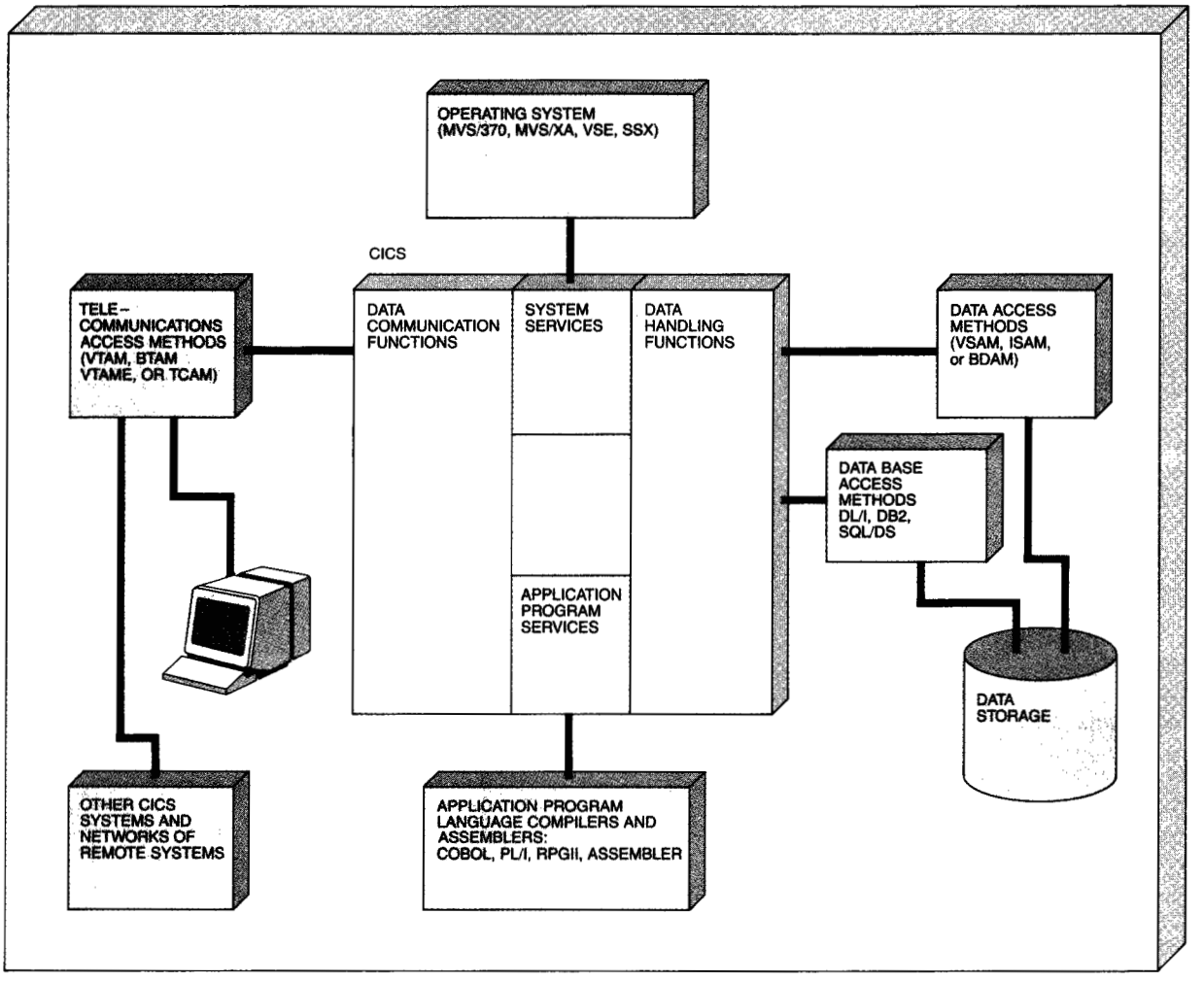
CICS in the mid-1980s

Before discussing the evolution of CICS, we first present the product as it exists today. CICS is a general-purpose transaction management system and is also thought of as a Data Base/Data Communication (DB/DC) program product that can be used as an application enabler by the user. CICS provides support for data communications access methods, such as VTAM, BTAM, and TCAM, with strategic emphasis on VTAM and the IBM Systems Network Architecture (SNA). Applications may be designed to use one of several data management facilities supported via CICS, including such data base management systems (DBMS) as the Information Management System (IMS), IBM's Database 2 (DB2), SQL/DS, and standard file access methods, such as VSAM and BDAM. (See Figure 1.)

CICS provides a transaction control system that the user can implement for use in tightly coupled systems such as the IBM 3090-200, the 3084Q, and the 3081K. A single CICS system, running in an MVS/XA environment and executing COBOL applications using VSAM files, has been measured sustaining workloads of greater than 80 transactions per second at approximately 70 percent utilization of an IBM 3090-200. Other measurements have shown CICS Multiple Region Operation (MRO) configurations executing comparable workloads and achieving transaction rates in excess of 120 per second while fully utilizing an IBM 3090-200.

When running in tightly coupled environments, CICS is designed to enable the user to apply the multiple processors against the systems workload. In that environment, VTAM would be the preferred data communications access method, executing as a separate MVS unit of work and managing the physical terminal network for CICS. The application workload can be distributed across multiple CICS address spaces to achieve any one of a number of desirable user objectives.

Figure 1 The software environment that surrounds CICS



To achieve higher availability, a user may configure a CICS MRO system so that one CICS system acts as a terminal-owning region, one or more address spaces might contain file or data base systems, and still other address spaces contain application programs. This type of configuration not only allows the use of all processors on behalf of the machine's DB/DC workload, but it also has the potential to improve system and application integrity and availability. In the event of any failure within a single application address space, the failure is limited to that one application group only, and restart can be done quickly, because the terminal-management and data-management systems have not been affected.

With a single MVS system, the distribution of a CICS workload is not limited only to multiple address spaces. Like MRO, CICS also supports Intersystem Communication (ISC) to connect CICS systems that reside in different MVS systems. Using either facility, the system designer can configure a global system, so that applications and data resources can be physically located in any of the connected systems. CICS supports transaction routing, thus allowing terminal input to one CICS system to be delivered to another CICS application-owning system for execution. The data accessed by an application can reside on any other CICS-connected system.

While CICS is architected to enable the user to distribute his workload or optionally to exploit the expanded environments of MVS/XA (i.e., tightly coupled and/or network-connected systems), CICS also is able to support smaller DOS/VSE systems. Most

CICS functions are available to the VSE user, thereby allowing upward compatibility and the participation of VSE systems in a network perhaps also containing MVS-based systems. Transaction routing or remote data access can be performed transparently among CICS VSE and MVS systems. CICS/DOS/VS supports MRO and ISC in a manner similar to CICS/OS/VS. The CICS command level (CL) application programming interface (API) provides a consistent, functionally rich environment to both VSE and MVS users.

Few commercial users of data processing had considered transaction processing systems.

The beginnings of CICS

To a person who is not familiar with CICS, the preceding discussion may appear to be a description of a currently available program product, and yet its origin and its evolution are far from obvious. In the relatively short history of data processing, it should be noted that very few software products have had the architectural soundness to continue to exist in their original forms. In contrast, the CICS introduced as a program product in 1969 is architecturally very similar to today's product.

During the period from 1955 to 1960, computers were being introduced to the scientific and academic communities. Only the very largest commercial enterprises had made a beginning with computers and data processing. As we entered the 1960s, computers became more commonplace, with the IBM 7000 series being available for large users and the newly introduced 1400 series bringing data processing to many medium-sized organizations. At this point, operating systems were hardly known, and users performed most work as a batch process. Data communications among computers was almost unheard of.

With the advent of the IBM System/360 in 1964, the new hardware and software products offered new capabilities and opportunities. Except for airline reservations and military systems, few commercial users of data processing had seriously considered designing and implementing transaction processing systems. Some computers were being used as high-volume message switching systems, but few were used as central processors containing data and applications for use by remote terminals.

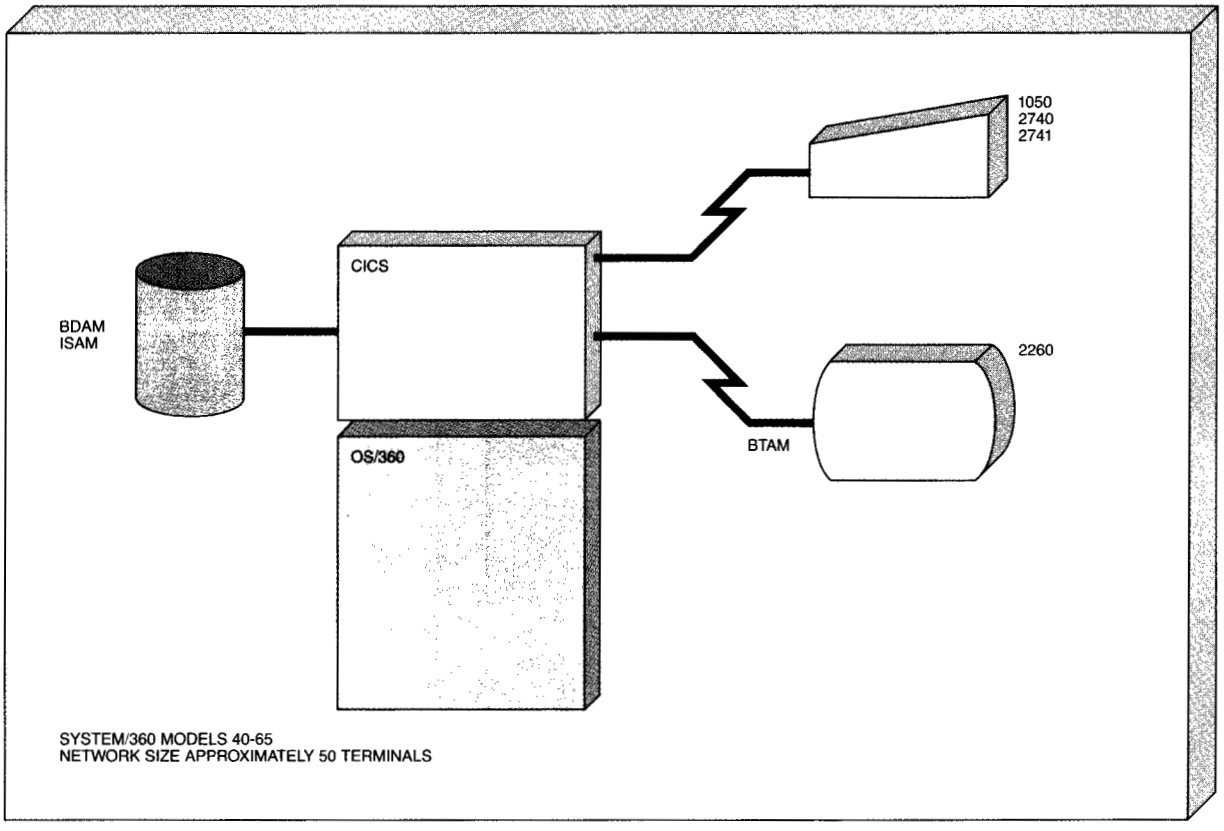
From 1965 to 1970, a variety of attempts were made to design and implement terminal-oriented transaction processing systems. The concepts were clear, but the skills and/or general-purpose solutions were in short supply. Although many program offerings became available during those years, they were often limited in scope. For example, some programs supported local terminals but not remote terminals. The nature of application programming was varied, and, in many cases, such programming was felt to be difficult. The operating systems, data management, and data communications access methods, although substantially more powerful than facilities of the early 1960s, were deemed large, costly, and/or difficult to use.

Evolution of CICS as a transaction management system

Figure 2 shows CICS as it first made its appearance in 1968. It was produced by a group of IBM workers in an effort to create a general-purpose transaction management system that would allow IBM customers to implement terminal-oriented systems more quickly. The initial interest at that time was for customers in the utilities industry. Many electric, gas, and water supply companies wanted to implement systems by which their customer service representatives could make inquiries to central files or initiate some customer-related transactions such as bill payment and service change. Despite the high interest, the prospective users were in need of assistance regarding the implementation of such systems.

The first versions of CICS supported only OS/360, assembler language application programs, BTAM for data communications, and ISAM and BDAM as standard file access methods. CICS also provided many environmental services using facilities designed especially for it, rather than comparable operating system services. This was done primarily to conserve processor cycles and/or real main storage. To recall briefly, the operating systems at that time included the Primary Control Program (PCP), Multiple Fixed Tasks (MFT), and Multiple Variable Tasks (MVT).

Figure 2 CICS in 1968



None of these systems was particularly suited to support the potential execution of multiple, concurrent applications for 20 to 100 terminal users. PCP supported only one application in the entire machine. Although MFT and MVT supported up to fifteen separate jobs, tasks, or programs, the operating systems' subtasking capabilities were felt to be very costly in terms of overhead.

To support the various operating systems conservatively, CICS provided its own multitasking, program, and storage services. CICS did not support QTAM, because of its message-switching orientation, nor did it support EXCP, because of its low-level coding. For data communications, CICS provided a terminal control function that polled terminals to invite new input. CICS also serviced application requests to read or write terminal data, and it handled error situations without directly involving each application program.

As new transactions were initiated, CICS could load the application program, if it was not already in main storage. CICS provided its own program fetch on an asynchronous basis so as to dispatch other transactions concurrently. CICS managed main storage within its partition or region, not only to reduce the overhead of comparable operating system facilities but also to manage storage use by individual transaction. CICS freed main storage upon explicit request or transaction termination, or programs not being used while at the same time storage is needed.

CICS also provided a file control function as a service to its applications, thereby removing the need for each application to be concerned with the open/close function or storage allocation for file-related areas. This type of facility management was intended to limit the application concern only to the function to be performed.

Additionally, the first version of CICS introduced functional components to provide generalized queuing services. Transient data and temporary storage services allowed applications to route data to

other applications or destinations, and, as for the file-control component, the using application needed only to be concerned with the functional read/write request and not with the environmental concerns of storage management or other media concerns.

During 1968–1970, users who were considering implementing on-line systems had a number of decisions to make. Among them was evaluating the available general-purpose software and comparing those possibilities with what would be involved in designing and implementing their own systems.

In 1969, IBM began producing licensed program products. CICS was among the first of these and was considered to be a primary DB/DC offering. In order to avoid the unnecessary production of software with similar capability, IBM identified selected program products as the primary product to address certain functional or application requirements.

As potential users began to focus their attention on products such as CICS, other requirements became more apparent. If CICS were to be considered viable by a broader spectrum of users, it would have to support other environments or functions. Foremost among these at the time was the function of CICS support of COBOL and/or PL/I in addition to assembler language. It was acknowledged that an application could be produced using assembler language that might be more efficient in its use of such environmental resources as real storage and processor cycles than a higher-level language. However, because of the ease of use of higher-level languages and the availability of programmers skilled in COBOL or PL/I, such support was requested.

To incorporate high-level language support into CICS posed several problems for its developers. First, COBOL did not produce re-entrant or reusable code. Both COBOL and PL/I produced executable code which assumed that it was in control of its execution time environment. That is, if a program had to wait for a certain event, the compiler inserted an operating system wait until that function had been performed. If main storage were needed, the compiler issued an operating system request. If an error were to occur, the compiler simply assumed that that task should be abnormally terminated. None of these compiler actions was deemed appropriate for a multitasking, on-line application system.

In 1970, CICS delivered its first support of COBOL and PL/I. Of primary interest, architecturally, was that for COBOL, CICS made a copy of an application's working storage, if more than one terminal caused execution of the same program. For PL/I, CICS supplied a storage management program that could satisfy the compiler's request for storage but in a manner consistent with on-line CICS execution.

Also of major importance was the decision regarding the CICS application programming interface (API). Assembler programmers were accustomed to issuing macro statements. COBOL and PL/I programmers expected high-level language statements, but they might use CALL statements on an exception basis. CICS developers were interested in a common interface that could be used by all programming languages. Therefore, they chose to offer its macro-level interface for use by all languages. In the COBOL and PL/I case, a preprocessor was furnished to convert CICS macro statements in the source program to high-level language call statements that were then compiled.

Regardless of the application program and its language, internally CICS had a consistent interface.

The architectural significance here is that regardless of the application program and its language, internally CICS had a consistent interface. Application requests were quickly routed to the internal component that provided the required service. This CICS macro-level interface continued in the product until the CICS command-level interface was introduced in 1977 and became the preferred application program interface.

Influence of the Disk Operating System on CICS design

The need for on-line transaction processing was not limited to OS users. Soon after the introduction of CICS, Disk Operating System (DOS) users began asking for comparable facilities for their environment.

Because of its functional modularity, CICS was relatively easily adapted to a different operating system environment. The architectural decisions to be made if CICS were to support DOS revolved around making the specific environment transparent to the using applications and permitting a CICS-like system to function in a small systems environment.

The first requirement was fairly easily met. In the CICS/OS case, the application programs did not communicate OS requests directly. Granted, in many cases the CICS functional requests may have looked like their OS counterparts, but they were still part of the CICS API. Internally, CICS would use the OS facilities of its own choosing to carry out the external request of its application. To do something similar in DOS became straightforward. Although DOS did not have a storage management facility comparable to that of OS, the developers found that CICS could easily emulate that facility. The same was also true for file- and data-communication services.

A major requirement for CICS was that it fit into the typically smaller configurations common to DOS users. Primarily, that meant being able to function in systems with less real storage available. In this regard, CICS developers created the following two DOS offerings: CICS/DOS Standard (CICS/DOSS), which was virtually identical to its OS counterpart, and CICS/DOS Entry (CICS/DOSE), which was a new offering that preserved the CICS API but limited internal execution to only one application at a time. It did this by using a roll-in/roll-out technique, whereby only one terminal's application was in main storage at a time. When the current application finished or reached a wait state, it was rolled out to auxiliary storage, and another unit of work was rolled into main storage and given control of CICS.

CICS continues to offer DOS-based system support today. However, the Entry version was discontinued in the late 1970s, owing to the introduction of virtual systems as discussed later in this paper.

By 1971, CICS was gaining the attention of both OS and DOS users and offered support for assembler, COBOL, and PL/I. As more users evaluated CICS as a possible system to control their terminal networks, still other requirements were becoming known. New terminal devices were being introduced, and there was an expectation that if CICS were to be used as the base control system it should be supportive of these new terminals. In particular, one such new terminal was the IBM 3270. The 3270 was a marked change from the keyboard/printer or line-by-line display terminals of the late 1960s. The 3270 was a buffered terminal device that could deal with formatted data streams or send and/or receive individual fields of data, instead of an entire display. The field orientation allowed applications to intensify data or suppress the display of data.

CICS provided not only native 3270 data stream support but also a data mapping facility.

CICS provided not only native 3270 data stream support but also a data mapping facility, called Basic Mapping Support (BMS). BMS enhanced application programming by providing data and device independence not found in previous terminal-oriented systems.

Another requirement that gained wide support was that CICS should support the IBM data base management system. This, the Information Management System (IMS), was introduced in the mid- and late 1960s. IMS offered its users a base product that provided data base (DB) services, called Data Language/I (DL/I) and, optionally, a data communications (DC) feature. IMS provided hierarchical data base services via DL/I, but it did not support standard data management, such as ISAM or BDAM files. It was in this light that some CICS users—not wanting to commit all data to a DL/I organization—began looking for a system that would allow them a choice of DL/I or standard data management. The first CICS DL/I interface was offered in 1972. This interface was updated in 1975 and was significantly enhanced in the late 1970s and early 1980s. In contrast to the OS subtask architecture of the 1972 interface, the CICS DL/I interface today offers the user support of IMS data sharing, multitasking of up to 255 concurrent users, and other features to be discussed later in this paper.

CICS transition to virtual storage systems

In addition to the requirement to provide terminal support for the 3270 and to extend data management support to include data base systems, CICS, in 1972, was on the forefront of still another major architectural change—virtual storage systems. IBM announced virtual storage systems together with new operating systems and such new components for those environments as the Virtual Telecommunications Access Method (VTAM) and the Virtual Storage Access Method (VSAM). Thus CICS advanced together with the other related hardware and software technologies.

The developers of CICS were asked to provide a virtual storage (VS) version.

The developers of CICS were asked to provide a virtual storage (VS) version of its product and its capability. The expectation was that users had made a beginning and a commitment to an IBM primary DB/DC offering and that such users should be able to move their non-VS, System/360 real storage systems to the new VS systems of the 1970s. An examination of this requirement began with determining how CICS would function in a virtual storage environment. In much the same way as CICS had shielded its users from the physical environment in the past, it seemed appropriate that CICS should not allow the nature of VS systems to show through to individual applications under its control. Yet the developers had to support and exploit any characteristic of the new environment that might benefit the user and the user's applications. Thus the CICS storage management facilities were redesigned, with the introduction of new internal storage algorithms for long- and short-running transactions. Thus CICS allocated main storage by the type of request and use to improve locality of reference, reference patterns, working set, and other theoretically desirable goals.

Changes were made in program loading and task management, in order to deal with the possibility of page faults in a virtual storage environment. CICS had previously not required application programs to be pure re-entrant programs and could not now expect that from its applications as a means of redispatching one transaction in the event another encountered a page fault.

CICS terminal management facilities were enhanced to support VTAM, and file control facilities were enhanced to support VSAM. Of the two, the new support for VTAM was the major change. In contrast to BTAM, which executed as an integral part of CICS, VTAM was somewhat asynchronous and ran in its own environment. CICS support of VTAM had to be added in a manner consistent with previous BTAM support but also enabling the user to benefit from new VTAM support.

The new CICS VTAM support introduced a sharp contrast to the previous BTAM support. For instance, with BTAM, CICS was required to initiate line events such as polling and addressing, and to handle error conditions when they occurred. CICS was responsible for the frequency with which it examined the BTAM network, either looking for completed events that had to be acted upon or new events that had to be initiated. With VTAM, network events were more asynchronous. VTAM assumed responsibility for the management of the physical environment and its interface to a using subsystem or application. Thus CICS was on a more specific-unit-of-work basis. CICS no longer had to look for network-related work by doing terminal control table scans. Rather, it could deal directly with elements on a work queue that were posted there by VTAM.

CICS support of VSAM required provision for coexistence as well as migration from ISAM or BDAM. CICS provided initial support for base clusters and alternate indexes, ISAM/VSAM compatibility, relative record data sets, and other VSAM facilities. With each CICS release since the introduction of that initial support, CICS has enhanced its support of VSAM.

The early years of virtual storage systems

Virtual storage was announced in 1972, and by 1974 the products, including CICS/VS, that were to support VS had begun to be delivered to the field. CICS/OS/VS and CICS/DOS/VS were delivered in 1974 and enhanced in 1975. By 1976 four releases had been delivered. Some of these releases delivered new function and all delivered refinements to the initial support of virtual storage systems.

Figure 3 CICS/VS in 1977

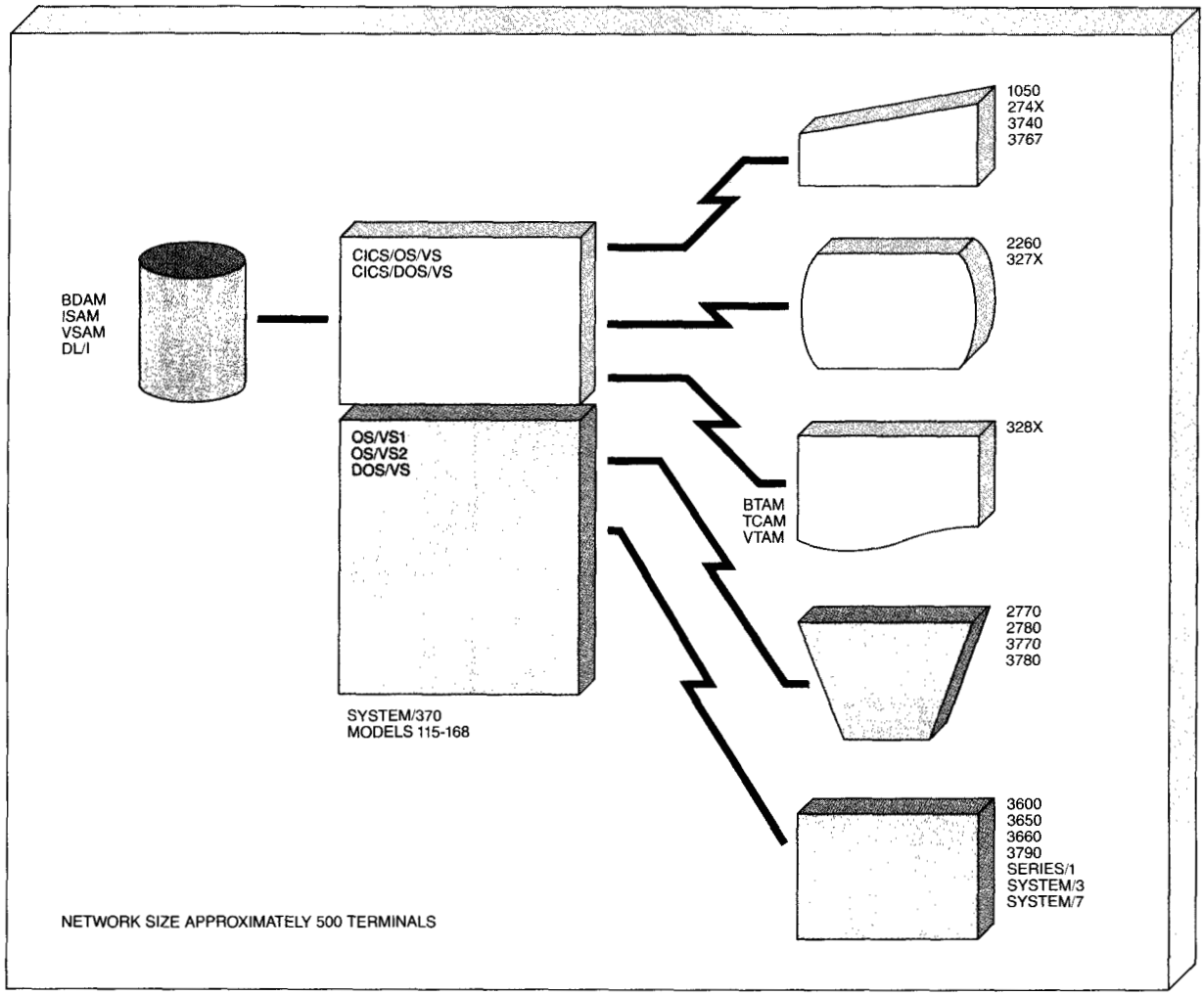
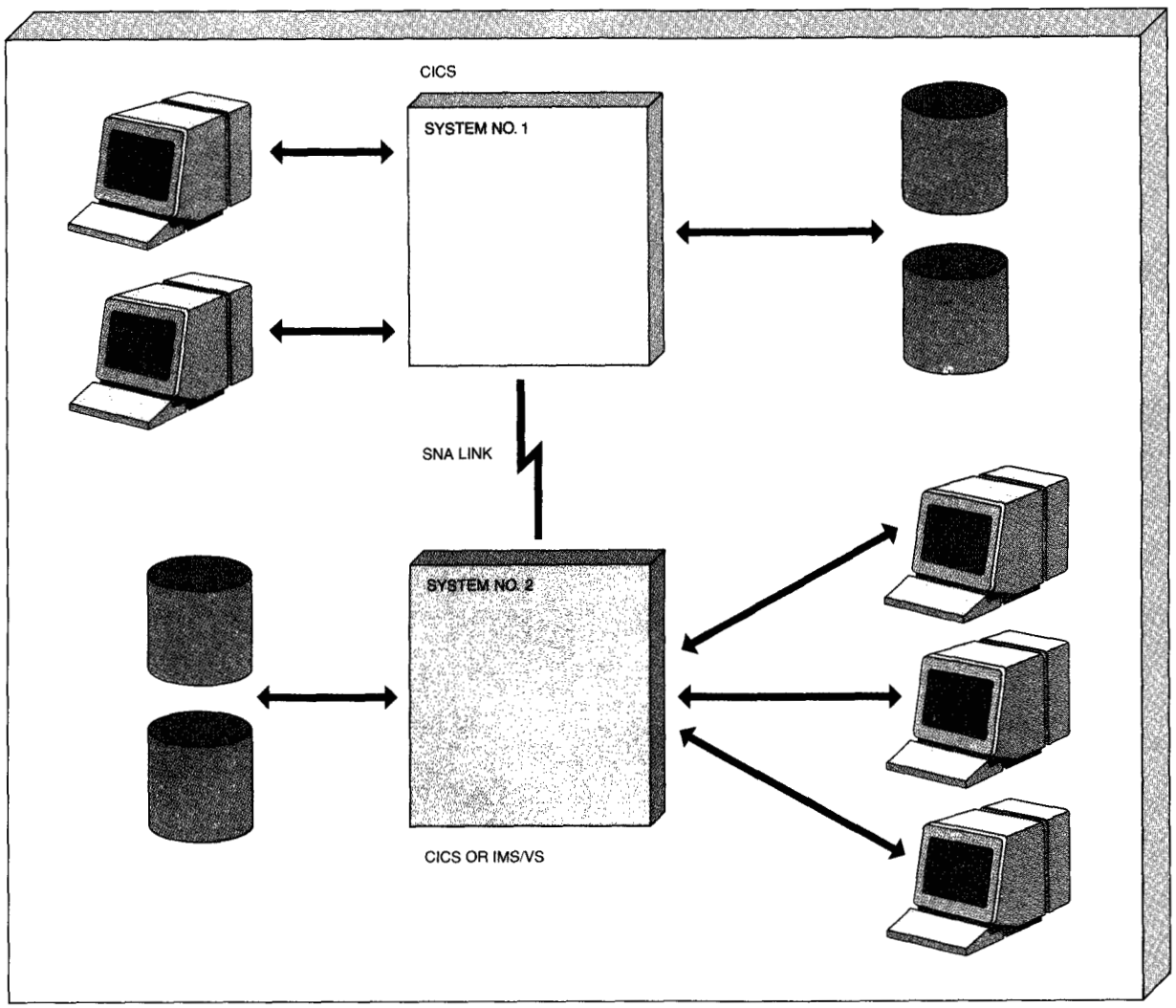


Figure 3 shows CICS during the 1974–1977 period, when many users were beginning to install virtual storage systems. By 1978 virtual storage had established itself. The products had made the necessary adjustments, and a new confidence existed in systems that supported considerably more virtual storage than real storage. Thus many users who had been constrained by the limitations of real storage systems began to move to and expand their use of virtual storage systems. By this time, components such as VTAM and VSAM had also demonstrated their merits. VTAM had proved to many users that it could produce better performance characteristics than its predecessors, BTAM and TCAM. About this time, IBM demonstrated its strategic intent to support Systems Network Architecture (SNA) through implementations such as VTAM. Thus CICS also adopted VTAM as its strategic data communications access method.

VSAM also demonstrated its ability to perform as well as or better than previous access methods, while also providing considerably more function and service to the user and his applications. CICS further enhanced the use of VSAM in 1977 by supporting Local Shared Resources (LSR) and Alternate Indexes.

Users began redirecting their attention to new requirements. No longer constrained by real storage, and with adequate basic function, more users indi-

Figure 4 An Intersystem Communication configuration



cated a readiness to expand their applications and terminal networks. This new set of requirements placed an emphasis or priority on the ability to distribute the workload and its resources. Users began addressing their application backlog with the new facilities that met the needs of application development. To handle the expected growth in systems and applications, users exploited the new hardware and software environments.

Further enhancements of CICS

CICS began a series of significant releases and enhancements, beginning with Version 1 Release 3.0 in 1977. In that release, the command-level interface was introduced. Command level, sometimes called the High-Level Programming Interface (HLPI), provided a number of enhancements over and above the previous CICS macro-level API. No longer would programmers need to be concerned with internal CICS control blocks or interfaces. The programmer could code in a more English-like syntax and vocabulary and perform application debugging and development, using as an aid the Execution Diagnostic Facility (EDF). Command level provides precompile diagnostics. It is also easier to learn and maintain. Most significant is the fact that the command-level interface provides a much better architectural inter-

face than does the macro-level interface. Since 1977, command level has been indicated as the preferred API for CICS applications.

In 1978, CICS delivered Version 1 Release 4, and a major enhancement to that release was the introduction of Intersystem Communication (ISC), illustrated in Figure 4. ISC enables applications to issue requests for data resources such as VSAM, ISAM, or BDAM file records, DL/I data base records, and transient data or temporary storage queue records, without regard to the physical location of such resources. CICS ISC provides function shipping of individual application requests to read or write data resources in such a way that they are transparent to the requesting application. CICS routes the request to the resource-owning system, executes the request, and returns either the data or a return code to the requesting application.

True distributed systems had become a reality.

In 1979, it was announced that CICS would include a new facility that was similar to ISC but that would allow multiple CICS systems in the same machine to be connected. With this new Multiple Region Operation (MRO) support, the user could configure multiple CICS regions or partitions, with terminal, application, and data resources distributed as the user required. MRO enabled the user to enter a system through terminals connected to one system and have transactional input routed to other connected systems for execution. With function shipping, the data resources used by an application could be connected to the same or different CICS in the same or a different machine. Thus, true distributed systems had become a reality.

New operating systems and hardware systems came into being in the early 1980s. By this time, many users depended on the ability of CICS to keep pace with other enhancements in related products. More was expected of CICS to support extensions to SNA, such as Logical Unit 6.2 support and the new office systems such as DISOSS, Scanmaster, and Displaywriter.

IBM introduced a new relational data base capability with SQL/DS and DB2. Coincident with those announcements came the expected CICS attachment support. With CICS/OS/VS Version 1 Release 6.1, CICS began supporting advanced data base facilities such as Data Base Recovery Control (DBRC) and IMS Data Sharing.

In the mid-1980s, many users are migrating to the expanded environments of MVS/XA. CICS users are moving their workload data and application storage requirements that are greater than the so-called 16-megabyte (MB) line to gain the benefit of the extension of virtual storage to 31-bit addressing. The initial CICS support of MVS/XA relocates a number of storage allocations above the line in a transparent fashion, requiring no change to user application programs. As new compilers become available that are capable of generating 31-bit addresses, users are able to move application programs above that line also. With programs, data buffers, temporary storage, and other functions in expanded storage, considerable relief is expected to be provided to those systems that had been storage-constrained because they were operating below the 16-megabyte line.

These examples of CICS enhancements since 1977 demonstrate the product's importance to its users because it has been supportive of user requirements and expectations in a timely manner. In prior years, the user may not have placed high priority on anything other than the function needed to meet a particular application or system need. With the advent of virtual storage systems, increasing hardware capacity, and lower-cost performance—due primarily to advances in hardware technology—the user was more inclined to expend some of that increased capability on items other than API function. Whereas the CICS macro-level interface had served users functionally from 1968 to 1977, the time had come for advancements in application development. Hence, the positive acceptance of the new command-level interface made possible easier programming and more reliable code.

In the late 1970s, more than at any previous time, users were seeing a need to distribute their workloads. Somewhat contrary to the interest that seemed prevalent during the early 1970s to centralize much

of the data processing function, users began to think in terms of putting a data processing function where the work and the data resided, rather than bringing them to some central location. With this capability came user interest in distributed systems. In many cases, the user looked for symmetry and transparency. That is, existing applications should not have to change if resources were reconfigured. Also, functions that could be performed in one environment should be available in other environments. To that extent, CICS MRO and ISC have addressed the requirements for symmetry and transparency. Transaction routing, function shipping, and distributed transaction processing are available to the application and system designer, with minimal or no effect on the participating application(s).

LU 6.2 may very well be the key to future networking of systems.

The CICS support of SNA and in particular the support of LU 6.2 has been significant. LU 6.2 is the basis for Advanced Program-to-Program Communication (APPC) and is widely used to enable potentially dissimilar end points to communicate with each other. APPC can interface between two or more CICS systems. However, it can also be used to allow a System/38, System/36, or other LU 6.2-capable system to communicate with CICS. The underlying concept of LU 6 formats and protocols is that the end points have agreed to exchange architected message models. This basic agreement of support enables potentially differing systems to exchange meaningful data. Although this may seem simple by today's standards, it has taken some years to achieve. LU 6.2 may very well be the key to future networking of systems.

The initial support of DB2 by CICS is significant in more than merely its access to new function. The DB2 attachment to CICS uses a new interface called Task Related User Exits, or sometimes the Resource Manager Interface (RMI). This new CICS interface has been put in place to accommodate non-CICS software that is intended for use by CICS application programs. The significance of this new interface is that CICS need not be further involved with any non-CICS code. No other special interfaces are required to permit the use of non-CICS code with CICS applications. Through the use of RMI, CICS and its related products are now able to achieve release independence.

RMI permits the developer of a resource manager (such as DB2) to activate code asynchronously with respect to CICS (before or after CICS initialization). RMI also provides a means by which a CICS application program can issue function requests to the resource manager and receive the architected response. Also very important is the provision of a commit/abort architecture. That is, when a CICS application issues a syncpoint request or when it abnormally terminates, such events are communicated to any connected resource manager so that it can take similar action on those resources for which it is responsible.

These and many other examples could be cited to demonstrate advances in hardware and software technology that have created new environments and new opportunities for new and existing CICS users. The expectation has been that products such as CICS provide timely support of new facilities and remove any major inhibitors to the user's system growth. The process just discussed has led to CICS/VS today, which is shown schematically in Figure 5.

CICS in the future

Consider Figure 6 as we look forward into the remainder of this decade and further into the 1990s. Here are some thoughts those of us who function as system developers should give to requirements of future systems. Here, the systems engineer of today contemplates the direction for systems of the future. Up to now, the developers of CICS have attempted to provide balanced releases. They have tried to give these releases something for all users by providing advances in numerous areas. One might attempt to categorize these areas in the following way.

Processors and operating systems. If CICS is to continue as a strategic base product, providing environmental management for terminal-oriented transaction systems, it seems natural to assume that it should function in large- as well as small-systems environments. At the so-called high end, today's capacity of 100 transactions per second might well be extended to perhaps 500 transactions per second in the next decade. To some users, a 500-transaction-

Figure 5 CICS/VS today

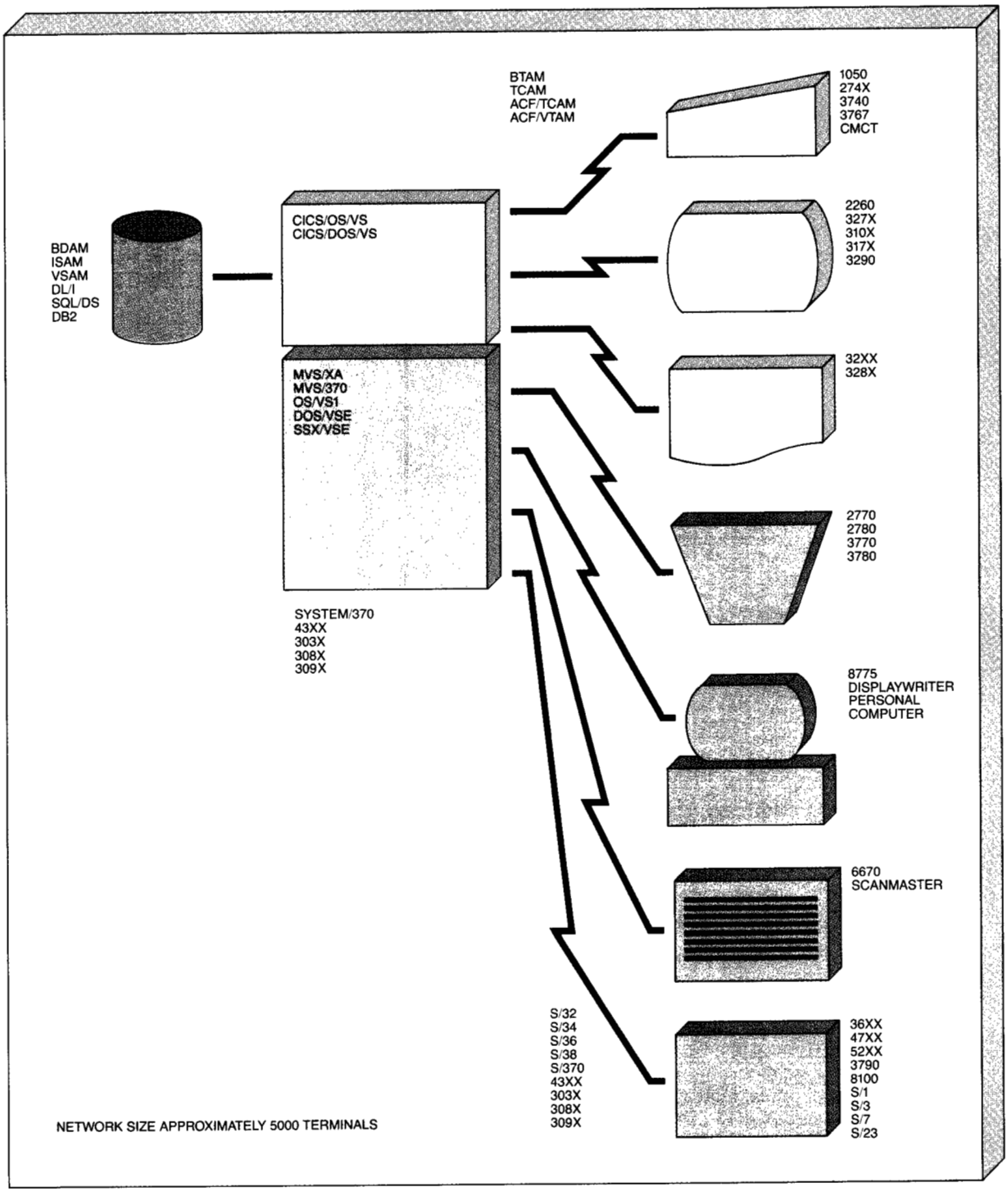
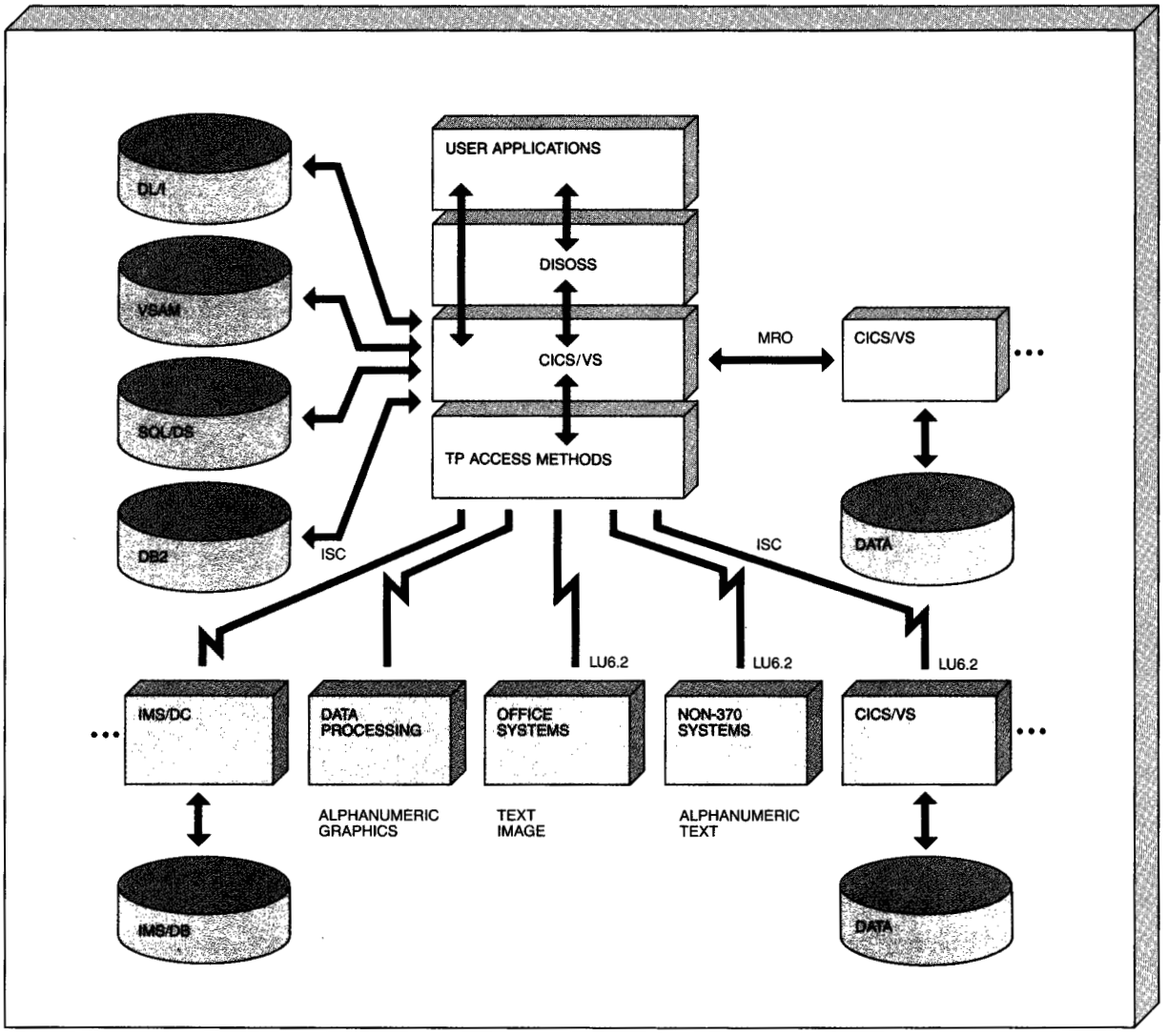


Figure 6 The strategic role of CICS/VS



per-second system may sound unrealistic, but with the advent of low-cost terminal devices, consumer or point-of-sale terminals, intelligent work stations, personal computers, etc., the size of networks is expected to increase dramatically. With that growth will come a corresponding increase in message or transaction rate. At the low end, CICS functions today on an IBM 4300-class machine. With intelligent work stations (IWS)—such as the IBM Personal Computer, PC/XT, or PC/AT—having System/370 instruction capabilities and sizable storage capacities, there is a natural expectation that system or application code that was once considered appropriate only for host machines might now be considered for placement and use on an IWS.

Thus we should strive to understand and document the requirements for high-transaction-rate systems. Such systems have the ability to distribute their workload, both data and applications, across multiple processors in a tightly coupled system or across IWS within a network. In order to achieve the throughput implied by such systems, host nodes and IWS must work in concert.

Networks and terminals. Probably the greatest growth challenge for a CICS-based system to deal with will be the expected large numbers of IWS that will be used in the coming decade, together with greater connectivity to other CICS- or LU 6.2-capable nodes using Intersystem Communication (ISC). This implies increased use of transaction routing, function shipping, and distributed transaction processing. The user will expect to have data processing capabilities local to where the need exists and relevant data and applications accessible by other nodes within the network. Equally, an IWS user will expect to have access to applications and data, with security understood, anywhere in the network.

Large networks will pose new requirements for the definition of such networks.

Large networks will pose new requirements for both the definition and the management of such networks. Compared to early methods of resource definition, today's requirement indicates the need for more dynamic definition and recognition of resources to be used with on-line systems such as CICS. This applies to all types of resources, not just terminals. Resource definitions will be shared among all using subsystems and will be more global in nature.

CICS and business data. Probably no one data organization will be adopted by all users in the near term. This comes in part from the observation that the usability of CICS in the past has been attributed to its support of both standard data management and data base management systems. VSAM should continue as the strategic, standard file access method, perhaps to evolve to a similar but even more comprehensive data management facility in the 1990s.

During the remainder of the 1980s, CICS users are expected to increase their use of relational data base management systems such as DB2 and SLS/DS. Hierarchical data base systems are also expected to grow in both number and size, the expansion being aided by facilities such as IMS Data Sharing. This makes resource sharing possible among batch and on-line systems, on the same or adjacent machines, and with complete integrity.

Applications. COBOL has been the predominant choice of programming language among CICS users since 1974, and it is expected to remain a popular choice for some time to come. Application generators or aids have been produced to assist the user with the definition and implementation of new or changed applications. The need for more and greater application development assistance is expected to increase. Most users are expected to employ multiple programming languages, tools, and aids to meet their requirements for application development and maintenance.

The IBM Cross Systems Product (CSP) is perhaps a representative application development aid; it assists the user with both the creation and the execution of his application. The IBM Screen Definition Facility (SDF) is another current product that is indicative of the type of aid which can assist the application developer.

VM/SP has made its Conversational Monitor System (CMS) facilities as well as compilers usable on the IBM PC/XT and PC/AT. For those CICS users who already do application development under VM or who are considering the use of VM for application development, this also may be indicative not only of an offload of host work to IWS but also of enhancing the use of IWS as an application development machine.

CICS itself will probably continue to focus on providing environmental facilities. That is, the CICS user should expect to be able to run in current operating systems, use current access methods, implement applications using current programming languages, and utilize available application development aids. CICS has repeatedly indicated the strategic importance of the command-level interface. Thus one would expect users to code to that interface or, where application generators are used, to produce executable code that also uses the command-level interface.

Packaging and installation. In recent years and in recent CICS releases, there has been a movement toward more pregenerated code or systems. In prior decades, many users were often resource-constrained, so that tailoring or customizing of systems

was extremely important. With increasing capacities at lower cost, such concern has diminished. Attention is now being focused on other than the physical installation of software products. The user today seems prepared to install and use pregenerated systems, which suggests object code installation and service. The need for code modification and/or unique system generation seems to have declined. The trend of need for user exits or extensions of base code probably will continue.

In Version 1 Release 2.0 in 1976, CICS introduced the concept and support for both source- and object-level compatibility. This has been extremely important to users with large inventories of applications and allows the customer to continue using existing programs that do not require recompilation due to the installation of a new release of CICS.

Other areas of interest. As we look forward, CICS developers must also consider aspects of systems, applications, and software products that perhaps did not receive sufficient attention in the past. Now, however, not only are they affordable but they are also increasing in importance and value to today's user of data processing. The following are a few of these areas of interest.

High availability. Many users have wanted high availability, but previously they may not have been able to justify the system investment needed to increase or ensure higher availability. Today, CICS can contribute to higher availability and faster restart, through the use of such facilities as MRO. CICS developers have indicated their intention to support the MVS Extended Recovery Facility (XRF).

Continuous operation. For many users, their data processing has shifted from batch systems to the long-running, on-line systems of the present. The application investment in on-line systems has created the expectation of almost unlimited accessibility of those systems. This implies that the designer and implementer of on-line systems now must provide for maintenance and change of the system with minimum disruption to the end user. Maintenance facilities provided should permit transition to new or redefined systems without affecting the end user.

New applications and environments. Transaction processing systems, such as CICS, have done well to date with applications dealing mostly with character-type output devices such as display terminals. With the considerable power of today's systems and the expectation of even greater processing power in the future, systems should be able to grow into areas not heretofore considered viable or feasible. Some of the systems growth of the near future will come from such technologies as on-line business graphics. Perhaps image or facsimile processing can also be considered as application enhancements.

The challenge for CICS developers is this: As terminal networks grow and invite more use of existing applications, and as new devices and new processes are implemented, an expectation will exist for the underlying components to support such systems.

Concluding remarks

This paper has briefly described the past, during which CICS came into being to address major user requirements for transaction processing. We have touched on some of the changes that have occurred in both hardware and software which not only affected CICS and motivated a change or enhancement, but also enhanced the ability of CICS to serve its users. We have looked at the present with a view toward the foreseeable future. CICS users and IBM employees have contributed to the art of systems engineering in the growth and history of CICS. We look forward to continuing this joint interest in systems in the future.

General references

IBM Customer Information Control System/Virtual Storage, Direction and Strategy, G320-5890-1, IBM Corporation; available through IBM branch offices.

Customer Information Control System, Program Product Version 1.6 and 1.7, Program Numbers 5740-XX1 (CICS/VS) and 5746-XX3 (CICS/VS), General Information, GC33-0155-2, IBM Corporation; available through IBM branch offices.

B. M. Yelavich *IBM National Accounts Division, 5205 N. O'Connor Road, P.O. Box 160969, Irving, Texas 75016.* Mr. Yelavich joined IBM in 1957. He is currently working in the IBM NAD Dallas Systems Center–Large. Mr. Yelavich has been involved with CICS product support since 1968; he earned an IBM Outstanding Contribution Award in 1970 for the development of CICS education materials. In 1983 he received an IBM Exceptional Achievement Award from the System Products Division for his contributions regarding CICS. As of September 1985, he advanced to Senior Technical Staff Member; he is the first marketing division employee to hold that position.

Reprint Order No. G321-5253.