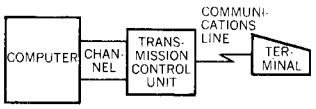
Evolution of the Customer Information Control System/Virtual Storage (CICS/VS) is discussed, along with a description of how CICS/VS manages a pre-SNA teleprocessing network. Following a brief review of SNA (Systems Network Architecture), the role of CICS/VS within an SNA environment is described. The paper concludes by outlining SNA's advantages to CICS/VS.

CICS/VS and its role in Systems Network Architecture

by D. J. Eade, P. Homan, and J. H. Jones

When on-line computer terminals were developed, so that input and output operations could be performed remotely from the computer, data could be entered into the system from the point of origin, and results transmitted directly to the place where they were to be used.[1] This mode of operation employed telecommunications, in which terminals were connected to the computer by a communications line as illustrated in Figure 1.

Figure 1 Remote terminal connection

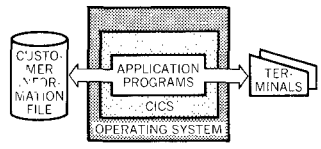


In IBM's System/360, an application program requested input and output telecommunications operations by means of the Basic Telecommunications Access Method (BTAM).[2] Through BTAM's application program interface, the application program was responsible for managing the flow of data on each communications line.

Besides managing the communications lines, an on-line application program had to allocate computer resources dynamically, according to the input, in order to make efficient use of the computer system. This need arose because the program did not control the timing of input operations, which were initiated by terminal operators acting independently. A result of these requirements was that application programs had to be designed to control the on-line system as well as to process data. This overhead inhibited the development of on-line applications for System/360.

IBM's Customer Information Control System (CICS) was developed to provide a simple programming interface for on-line terminal applications and to relieve the application programmer of the complex task of controlling terminals and communications lines. Developed initially for System/360, CICS was modified to take advantage of the virtual storage facilities of System/370, and its name was changed to CICS/VS.[3] Subsequently, CICS/VS was extended to support IBM's Systems Network Architecture (SNA).[4] That evolution is the subject of this paper. Emphasis is on the contrast between CICS support of pre-SNA terminals and the role of CICS/VS in an SNA network. The paper concludes by outlining the advantages of SNA to CICS/VS.

Figure 2 CICS on-line application program environment within the operating system



Structure of CICS

CICS runs under the operating system[5] of System/360 or System/370 as if it were an application program. However, CICS in fact provides an environment for on-line execution of the user's application programs, which actually process the transactions. This concept is illustrated in Figure 2.

system control

The basic components of CICS are designed to accommodate the unpredictable rate of arrival of input messages. Messages may arrive from several terminals in the time it takes to process one transaction, so in order to provide a reasonable response time for each message, the corresponding transactions must be processed concurrently. A separate thread of control is maintained for each transaction, and when one thread is delayed, as in waiting for access to a disk file, another is given control. These threads are called *tasks* and are managed by the CICS *task control program.*

CICS storage is divided into buffers, which are allocated by the *storage control program* to match the dynamically varying load. Since each task exists only for the duration of a transaction, individual buffers soon become available again for other transactions. Application programs are loaded, as required, by the *program control program.*

The CICS *terminal control program* runs as a task, within the on-line environment, just as if it were performing a transaction. However, this program's job is to manage the flow of messages between the other tasks and the remote terminals. Each terminal is solicited for input, and when a message is received, a task is created to process the transaction. The transaction is identified by a code typed by the terminal operator at the beginning of the message. The application program has access to the message in a storage buffer, and may generate a reply in the same buffer,

then issue the simple request WRITE. The terminal control program, having retained an exclusive link between the task and the terminal, delivers the output message.

The application program may terminate at this point, or it may continue a conversation with the terminal by issuing a READ request. In either case the terminal control program solicits the terminal for another input message—to begin another transaction if the application program has terminated, or to satisfy the READ request. This method of operation both provides a simple READ/WRITE interface for the application programs and enables the terminal control program to optimize the flow of messages.

Since CICS manages the multitasking of transactions, it must also manage requests from the tasks for services of the operating system. This requirement prevents a task from making a request and then waiting for the operating system to complete it, thereby delaying all CICS processing. When CICS has made a request on behalf of a task, it gives control to another task to continue processing. Only when requests have been issued on behalf of all tasks does CICS wait for the operating system to complete them. An example is the CICS *file control program*, which manages task requests for access to disk files. This program further coordinates the requests to prevent different tasks from updating the same file record concurrently.

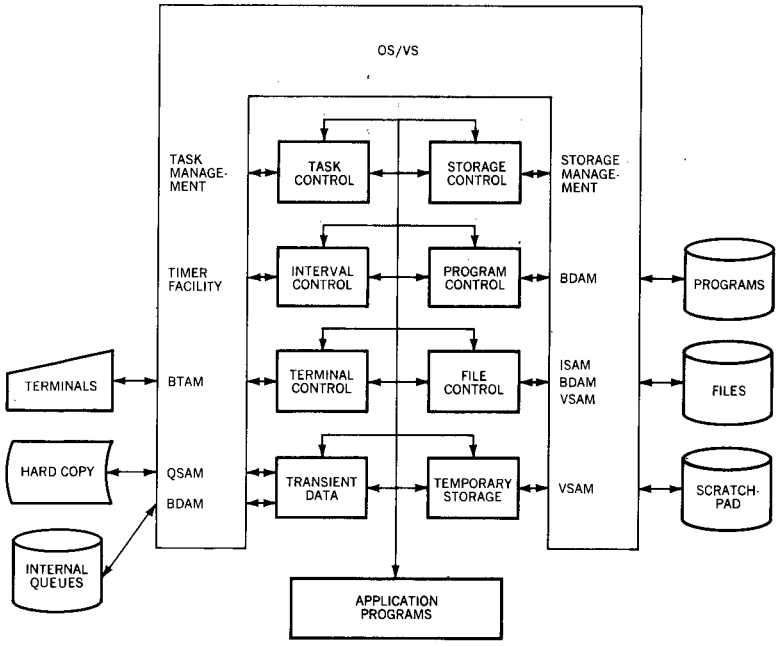
Other components perform ancillary functions that are common requirements for an on-line application program. A queuing facility, *transient data*, enables several transactions to build a sequential set of records for off-line processing or for output to a single terminal. A scratchpad facility called *temporary storage* allows a transaction to build a record containing intermediate results for processing by a later transaction. *Interval control* provides functions based on the system timer, allowing transactions to be initiated at a given time of day. Both the transient data and interval control facilities use *automatic task initiation*, which assigns a task to a terminal without the need for an input message from that terminal. This facility enables transactions to perform message switching.

Some of the components of OS/VS (Operating System/Virtual Storage)[6,7] that are used by CICS are shown in Figure 3. Similar interfaces exist when CICS runs under DOS/VS (Disk Operating System/Virtual Storage).[8]

system definition

The CICS control programs conform to the configuration of the application by interpreting *system definition tables*. These tables describe the hardware configuration, such as terminals and communications lines, and the logical configuration, such as transactions and programs. They also define parameters that

Figure 3 Operating system interfaces



determine the performance of the system, such as the number of tasks that can run concurrently and the number of storage buffers that can be allocated before the system becomes overloaded.

system service

In addition to the fundamental components already described, CICS provides a set of service transactions. These transactions are invoked from terminals in the same way as application transactions, but they operate on the system itself rather than on application files. Most significant is the *master terminal transaction*, which enables a terminal operator to examine or modify control information in the system definition tables. The operator can adapt the execution of the system to different operating conditions. Because it is not desirable to allow all terminal operators to invoke the master terminal transaction, or indeed to invoke certain application transactions, a security code is assigned to each transaction and to each operator. A *sign on transaction* enables an operator, by supplying a password, to invoke these transactions that have security codes compatible with his own.

The functions and structure of CICS are further described in References 3 and 9.

Evolution of CICS

visual displays

CICS has evolved in a way that reflects advances in related areas of computing technology. For example, the development of visual display terminals provided for more extensive communication between the application program and the terminal operator. In the IBM 3270 Information Display System, for example, the display image is structured into fields, some of which contain prompting information to guide the operator, while others are reserved for data. The technique for exploiting this medium required that the application program send a message that defined the structure of the display image to the terminal. This technique posed a new programming problem because of the extra control information required in each message.

basic mapping support

A new CICS component, *basic mapping support* (BMS), was developed to relieve the application program of having to provide display control information. The application programmer defines the desired layout of data on the screen in a description called a *map*. During execution, the application program supplies BMS with the map name and the output data in a format independent fashion. BMS constructs a data stream that contains the output data and fixed data from the map, and it inserts the control information required to produce a display image matching the map specification. The data stream is passed by BMS to the terminal control program using the existing READ/WRITE interface.

To provide printed output, BMS was extended to support typewriter terminals. Thus a program can now run on different types of terminal, leaving BMS to construct the output data stream containing the appropriate control information. This feature allows the program to be independent of output format and data stream. However, it is not strictly correct to say that the program has device independence, since the program may exploit, and thus be dependent on, the operational features of a particular type of terminal.

system improvements

Other changes in technology resulted in further enhancements to CICS. The introduction of virtual storage, for example, reduced the dependency of programs on computer memory size. The CICS storage control program took advantage of virtual storage to provide application programs with a greater ability to react to varying input loads. An interface to IBM's DL/I data base products[10] was added, together with a unified logging, recovery, and restart capability for both the DL/I and CICS files, thus providing a complete data base/data communications (DB/DC) system.

The original CICS application program interface was dependent on the internal structure of CICS. The enhancements and exten-

sions of CICS functions complicated this internal design and thus the application interface. Recently a new *command level programming interface* has been developed which enables application programs written in COBOL or PL/I to invoke CICS functions independently of the internal structure.

Terminal control

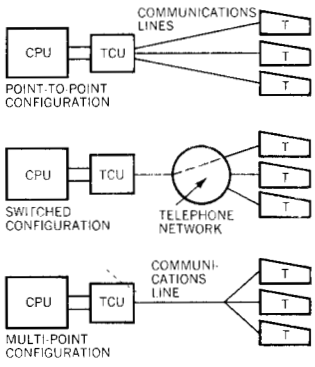
Because the CICS terminal control program addresses many of the network responsibilities of an on-line application, it illustrates many of the problems leading to the concept of Systems Network Architecture (SNA). For this reason, terminal control is discussed in more detail in the following paragraphs. Its logic has two major themes. One is the management of communications lines as a set of input/output devices so as to optimize throughput of the application's messages. The other theme is management of a dialogue between each application program and the operator of a terminal. The main concern in this respect is the convenience of the dialogue to both the application programmer and the terminal operator, without regard for the rest of the application. The reconciliation of these two themes causes some complexity in the implementation of terminal control.

line management

There are three major communications line configurations—point-to-point, switched, and multipoint,—which are illustrated in Figure 4. The multipoint configuration is used most commonly, since it economizes on the line lengths required, but it has the most complex control requirements. For each configuration, a protocol ensures that transmissions sent onto the line are received in their correct context. A line protocol defines certain characters as link control characters, which delimit transmission sequences authorized by the protocol. BTAM provides the terminal control program with an interface to each line, in which each request represents one of the authorized sequences.[11]

A CICS system definition table describes the configuration of the application's communications lines and terminals. This description enables the terminal control program to map the message flow from each line into a separate flow for each terminal, thereby providing the application program with an interface to each terminal rather than to the lines by which the terminals are attached. For each transaction output request, the terminal control program determines the correct line and ensures that a valid link control sequence is used to implement the request.

Figure 4 Communications line configurations



The typewriter terminals initially used for on-line transactions used start/stop, or asynchronous, line protocols, which were distinguished by the separate transmission of each character and a simple set of link control characters. The terminal control pro-

gram was designed to optimize the message traffic produced by interactive use of these terminals. The basic logic of the program is a periodic scan of the communications lines in the configuration. For each line, the scan detects the completion of input/output operations and chooses the next operation from among the requests outstanding for terminals on that line. Precedence is given to output requests, followed by input to satisfy specific transaction requests. Input to initiate new transactions has lowest priority. This scheduling algorithm minimizes the holding of CICS resources by each transaction. In this interactive environment, a different terminal can be serviced during each scan, and a similar response time provided for each terminal.

Subsequently, terminals were developed that used the Binary Synchronous Communications (BSC) line protocol. BSC supported high speed transmission of complete blocks of data and had a wider set of link control characters, providing a greater choice of line scheduling algorithms. Initially BSC terminals were designed for batch functions, in which a large volume of data was transmitted in one direction at a time, rather than for the interactive input and output messages previously associated with CICS transactions. The terminal control program's support for BSC enabled these terminals to make optimal use of the lines when operating in batch mode, but they were less efficient when used in an interactive mode.

Later, the IBM 3270 Information Display System was designed to use BSC lines interactively. The terminal control program was again modified so that the communications line was scheduled in an appropriate manner for this type of operation. However, prior to SNA, the terminal control program could not provide efficient support for a variety of terminal types on the same line.

An on-line system must have logic to recover from data transmission errors on the communications lines. Since the lines are outside the controlled environment of the computer installation, transmission errors can occur frequently but they are often temporary. The system should be able to distinguish between transitory and permanent errors in order to select an appropriate recovery action. When a transitory error occurs, the message must be retransmitted to override the error. When a permanent error occurs, the system should disregard the affected line so that the application, being composed of independent transactions, can continue although in a degraded fashion.

CICS provides for recovery from permanent errors in a manner that again insulates application programs from the physical characteristics of the terminal network. Line input/output operations that are completed in error are processed by a separate routine, the *terminal abnormal condition program* (TACP), which decides

the number of retransmission attempts necessary to distinguish permanent from transitory conditions. Once a communications line is determined to have a permanent error, TACP modifies the system definition tables so that the line is no longer used. TACP also abnormally terminates any tasks that are processing transactions for terminals on that line. An exit from TACP is supplied to a user program that can override TACP actions according to the requirements of a particular installation. Application programs therefore contain no error recovery logic, since they regain control after making an input/output request only if that request is completed satisfactorily.

terminal data flow

The line management logic within the terminal control program provides an interface by which the application can address each terminal independently of its attachment configuration. Supporting this interface is further logic to control the data flow to each terminal to meet the application requirements.

Each WRITE request results in the output of one logically complete message, although it may not be possible to transmit the message as a single block. If the terminal has a buffer, so that it can accept data from the transmission line faster than it can be printed, a large output message is divided into a number of blocks, each of which fits into the terminal buffer. The transmission of each block is delayed until the previous one has been printed. For start/stop terminals, CICS calculates the required delay according to the printing speed of the terminal. For BSC terminals, the link protocol defines sequences used by the terminals to indicate the completion of printing. This arrangement makes efficient use of multipoint lines, since the printing of a message at one terminal can proceed concurrently with transmission to another on the same line, and it illustrates how, prior to SNA, the terminal control program had to provide one function in diverse ways, depending on the terminal type and the line protocol.

Each block of a message is framed by link control characters. In the BSC protocol, different block termination characters are defined to distinguish intermediate blocks from the final one of each message. These characters are equivalent in terms of their link control function. The terminal control program chooses the frame characters on the assumption, stated above, that each WRITE is a complete message. This type of control, which does not regulate the communications line itself, is termed end-to-end control.

Some BSC batch terminals have several output devices, for example a printer and a card punch, that use the same line buffer. The final output device may be selected by the application program in the computer and specified during data transmission.

The method of selection depends on the line configuration by which the terminal is attached. On a multipoint line, the terminal control program selects the output device as part of the process of selecting the terminal itself. On point-to-point and switched lines, this selection information is carried as part of the output message. Thus an end-to-end control function can have different implementations, a situation in which CICS does not insulate the application program from the attachment configuration of its terminal. One of the features of SNA is to separate end-to-end control completely from the link protocol and thereby remove program dependence on the attachment configuration of terminals.

The practice of having several devices share a single line buffer is extended in the 3270 System, in which several display terminals use one control buffer for line input/output and also for operating the end-to-end controls that determine the display image format. Such an arrangement is termed a cluster. The terminal control program uses the link protocol to determine the origin in the cluster of each input message, and to select the correct terminal for each output message.

The automatic task initiation facility may lead to contention when the initiated task writes a message just as the terminal operator tries to invoke a different transaction. The terminal control program attempts to resolve contention by using the link protocol to prevent a terminal from sending input while output is pending from an automatically initiated task. Whenever the link control is insufficient for this aim, any conflicting input, unsolicited by the transaction, is discarded. Again, the terminal control program implements this function in any of several ways, according to the link protocol.

The link control is further used to make the flow of messages between terminal and application program match the program's sequence of READ and WRITE requests. With start/stop terminals, this function relies on the terminals' being used in the interactive fashion for which they were designed. With BSC terminals, which can be used in either batch or interactive mode, the link protocol defines a link control character that allows the direction of transmission to be reversed. In either case, this manipulation of the data flow is said to give the program *direct control* over the terminal. However, since the mechanism used is the link protocol, this level of control may not make the most efficient use of line capacity, particularly in multipoint configurations.

Control of the data flow permits CICS to assign each terminal a service status, such as OUTPUT ONLY or INPUT ONLY, or OUT

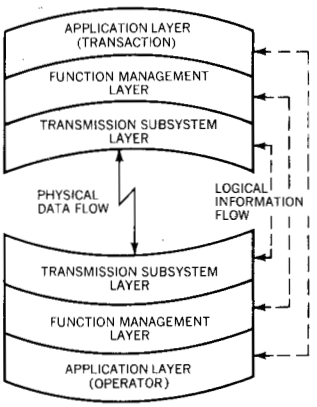
OF SERVICE, in which case no transmission at all is allowed. The service status can be changed by means of the master terminal transaction.

Faults in terminal hardware can be detected through the line protocol in a fashion similar to the detection of transmission errors. The same CICS error handling mechanism is used, since hardware errors are not related to the logic of any particular transaction. The only difference is that TACP restricts the reconfiguration to the removal of the failing terminal, rather than the whole line, from the active configuration.

The advent of SNA

CICS had provided an appropriate system for interactive transaction processing, using typewriter terminals manipulated solely through their start/stop link protocols. This approach had not proved ideal, however, when more sophisticated terminals were developed with more complex protocols, so that device control interfered with optimal line usage. Other on-line systems had encountered similar problems.

Figure 5 SNA functional areas



In planning for terminals and controllers that would take advantage of advances in hardware technology, it became clear that a single, common line control was required throughout the network, along with standardized device controls separate from the line control. The application program must have a view only of the terminal with which it is to operate. Terminal message flow and control must be independent of network management considerations and configuration. The separate functions required to transmit a message from an application program to a terminal must be defined clearly, so that new terminal types could be added to the network without disrupting any functions not directly involved with terminal characteristics.

layered structure

Systems Network Architecture addresses these problems by defining three distinct functional layers: the application layer, function management, and the transmission subsystem.[4] These layers are present in each end point of a communication system, forming a symmetric division of function with clearly defined interfaces, as illustrated in Figure 5. The following discussion is an overview of those parts of SNA that have particular relevance to the CICS/VS environment. A full discussion of SNA is beyond the scope of this paper and is provided elsewhere.[12-15]

The application layer is concerned only with application processing and not with the formats, protocols, or procedures required to route a message through the network. An application program in a computer is an end user of the communication sys-

tem and maintains a dialogue with another end user of the system. The other end user can be either a terminal operator or another application program in a computer or programmable controller.

The function management layer accepts requests from the application layer above it and transforms those requests into a form suitable for processing by the function management layer of the end user to which the request is directed. The function management layer thus is concerned with the presentation of information from one end user to another. The transmission subsystem layer is responsible solely for passing messages from origin to destination, without examining or changing their content. The fundamental difference between SNA and pre-SNA systems is the enforcement in SNA of the division of function between the function management and transmission subsystem layers.

transmission subsystem layer

The transmission subsystem[12] takes full responsibility for transmitting messages through the network, for reconfiguring the network, and for establishing logical connections between the end points of the communication system, which are described in SNA as *network addressable units.*

In the SNA implementation used by CICS/VS, the transmission subsystem is composed of the Virtual Telecommunications Access Method (VTAM)[13] in the host computer and the Network Control Program (NCP)[14] in a communications controller attached to the computer. The network's terminals are attached to the communications controller using an SNA line control discipline called synchronous data link control (SDLC).[15]

The Network Control Program contains two data link control elements, one of which supports communication with the host while the other communicates, using SDLC, with other attached controllers or terminals. An NCP path control element determines to which link (host computer or teleprocessing line) a message must be directed to continue toward its destination. The NCP takes on the line management functions, for which the terminal control program was responsible in CICS/VS, so moving them out of the host altogether.

VTAM provides network control and configuration services, and it contains a data link control element that communicates with communications controllers and device controllers attached locally (that is, directly) to the host computer. Application programs running under VTAM are not affected by changes in the physical network, such as the addition of lines and terminals, because they are provided with a network independent, terminal oriented interface.

VTAM presents its application programs with a logical interface to the transmission subsystem that allows full duplex transmission of data. VTAM also controls the flow of data through the transmission subsystem, ensuring, for example, that a program does not send data faster than the communications controller can dispose of it. This pacing of the data flow is similar to that performed by the CICS terminal control program for pre-SNA buffered terminals. VTAM's application interface can be interruption driven via application exits that gain control when a request has been executed. CICS/VS, as a VTAM application program, benefits from this feature by no longer having to scan control blocks representing network lines to determine what requests have been executed.

application layer

The application layer consists of the communication system's end users—the application programs and terminal operators that wish to exchange information. CICS/VS transactions, for example, are such end users. When interfacing with the transmission subsystem through VTAM, CICS/VS contains both the application layer and the function management layer.

function management layer

The function management layer is the interface by which application program requests are sent through the transmission subsystem to the specified end user. The end user replies, through a complementary function management layer, to the originating application program. Each of these function management elements is represented to the transmission subsystem as a network addressable unit.

SNA defines three types of network addressable unit: a physical unit, a system services control point, and a logical unit. Each node in the network contains a physical unit responsible for the management of its resources, while the system services control point is responsible for overall management of the SNA network. CICS/VS is represented in the network as a logical unit through which other logical units in the network have access to all the CICS/VS application programs. An application program in the IBM 3600 Finance Communication System, for example, is also represented in the network as a single logical unit, although such a program may control several terminals attached to the 3600 System. In contrast, the IBM 3767 Communication Terminal is a simple keyboard printer represented by a logical unit. CICS/VS is concerned only with communication between it and other logical units, rather than with physical units.[12]

sessions

Before a dialogue can commence between two logical units, a session, or logical connection, must be established between them. Establishment of a session includes selecting a physical path over which all messages will flow between the logical units.

When a session is created, parameters are exchanged between the two logical units describing the SNA protocols to be used during the session. The parameters are passed by means of the SNA BIND command. If BIND is accepted, both logical units follow the indicated protocols, which govern such procedures as the way messages are sent, error recovery, and data formatting considerations. These protocols apply to the logical flow of messages between the two function management layers and are independent of the physical characteristics of the links over which the messages flow. The BIND sender is referred to as the primary logical unit, and the BIND receiver is called the second logical unit.

The intelligence of a terminal's function management layer dictates the level of function exhibited by the logical unit representing that terminal. Some function management layers have little variability, while others can tailor themselves to explicit requirements of the subsystem with which they are to communicate. For example, the IBM 3770 Data Communication System can tailor its operation to the differing requirements of both Remote Job Entry and CICS/VS. The session parameters on the BIND command dictate the way in which a logical unit is to behave during the session.

CICS/VS requires that all logical units with which it is to communicate be identified to it at CICS/VS system definition time. It also requires a description of specific features of those logical units and a definition of the CICS/VS security and priority levels to be used by transactions communicating with them. CICS/VS uses this information to chose BIND parameters that suit its transaction processing environment. It is the function management layer that is responsible for enforcing and supporting the rules of its particular logical unit.

The following function management responsibilities are essential to communication between two logical units:

Data flow control

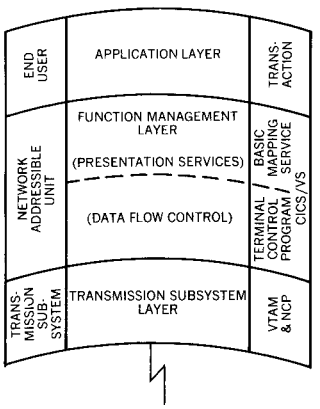
- Control of half-duplex and full-duplex message transmission, message chaining, and responses.
- Resolving contention when both logical units attempt to transmit messages simultaneously.
- Orderly shutdown of a session or temporary suspension of message traffic.
- Recovery of messages rejected because of data errors.

Presentation services

- Transformation of application requests into data formats for specific logical units.
- Control sequences, such as output component selection, between function management layers.

Communication between function management layers in SNA is by means of request units, which may contain either data or SNA commands. Request units are qualified by indicators in a request header associated with each unit. The maximum length of a request unit within a particular session is dictated by storage and buffer constraints in the sending and receiving logical units and is specified in the BIND parameters.

Figure 6 CICS/VS components within the SNA layer structure



CICS/VS as a function management layer

Figure 6 illustrates the three SNA layers and the CICS/VS components responsible for their functions. Part of the data flow control within the CICS/VS terminal control program is in support of messages being sent by a CICS/VS transaction. Other data flow control and session control functions in CICS/VS provide system services on behalf of the CICS/VS transactions.

data flow

CICS/VS retains its simple interfaces for basic mapping support and terminal control when communicating with SNA terminals. CICS/VS transmits a message from the transaction as a chain of one or more request units, the number depending on the length of the message and the maximum request unit length defined in the session's BIND parameters. This chain is the unit of error recovery in the event that an error is detected in any of the request units containing the message. Each request unit is passed to VTAM for transmission, together with parameters for the associated request header.

Messages to be transmitted to CICS/VS also can exceed the buffer capabilities of the sending logical unit, and thus have to be broken into multiple request-unit chains. CICS/VS issues receive requests to VTAM to obtain each request unit, and only when the whole chain has been assembled will it be passed to the CICS/VS application program to satisfy an outstanding READ request.

SNA chains carry with them a response indicator. Definite-response chains require the receiver to inform the sender either that the chain was processed successfully or that an error occurred. Exception-response chains do not require a response unless the receiver detected an error. CICS/VS handles these responses on behalf of its transactions. If a definite-response chain is received to satisfy an application READ request, then CICS/VS

delays responding until the transaction issues its next READ or WRITE request, thereby indicating that the request has been received successfully by the CICS/VS transaction program. Chains sent by CICS/VS normally request exception responses only, unless overridden by the transaction.

bracketing

When two application programs are in direct communication, as when a CICS/VS transaction is communicating with an application program in a device controller, they must maintain synchronization to ensure that messages are sent in the order expected by the receiver and that the sender does not transmit more messages than the receiver expects. In particular, it is essential to know that the complementary application program has not terminated prematurely.

CICS/VS addresses this problem in SNA communication by delimiting the bounds of each transaction using the SNA concept of bracketing, in which the first and last messages of each transaction are uniquely identified. The first request unit sent by either logical unit must have a *begin bracket* indicator set in its request header so that the receiving logical unit can recognize it explicitly as the first message of a bracket. Similarly, the request header of the last message contains an *end bracket* indicator.

Both the CICS/VS and the terminal logical units initially are in contention once a session has been established and messages are allowed to flow. This contention state, termed *between brackets*, allows both logical units to send requests. Through bracketing, CICS/VS can recognize contention situations and resolve them by enqueuing one transaction until transmission of the contending bracket is completed. An additional benefit of bracketing transactions is that spurious messages, such as those resulting from premature termination of a transaction and those received between transactions, can be recognized and purged until a message is received that contains the begin bracket indicator.

CICS/VS employs the BID data flow control command to request permission from the secondary logical unit to commence a bracket for a CICS/VS automatically-initiated transaction. If the secondary cannot accept a CICS/VS-initiated bracket, it can respond negatively to the BID command and in turn send the first message of a bracket that it wishes to initiate. If the secondary logical unit is a program in a device controller, it may choose not to accept a bracket because it is engaged on some local function, causing CICS/VS to queue the transaction for later initiation. When the local function is complete, the program sends the SNA READY TO RECEIVE data flow control command to signify that CICS/VS can now attempt to initiate the queued transaction. This definition of BID and READY TO RECEIVE in SNA recognizes the

general requirement for contention resolution in communications systems and provides a consistent approach across all logical units in which contention may arise.

SNA defines bracket rules to cover all the situations that can arise when both logical units are permitted to start and end brackets. CICS/VS has simplified the use of brackets by stipulating that only CICS/VS can terminate brackets, whereas a bracket can be started by either CICS/VS or any of its secondary logical units. In this way, the management of bracket protocols by CICS/VS is transparent to its transactions, and bracket control is simplified for programs in device controllers that communicate with CICS/VS transactions.

data flow protocols

The transmission subsystem interface is defined for full duplex flow of data, although the program logic for most sessions between logical units calls for a more restricted data flow than full duplex. Any of three data flow protocols can be specified in the BIND parameters for a session: half duplex flip-flop, half duplex contention, and full duplex.

Half duplex flip-flop transmission when in a bracket is ideally suited to the existing CICS/VS application program interface. It can be used for interactive transactions, which are the most common in the CICS/VS environment, as well as batch. With half duplex flip-flop operation, the flow of data is constrained so that only one logical unit can send at a time, and the receiving logical unit must continue to receive until granted permission to send by the current sender. Permission to send is indicated by a *change direction* indicator in the request header. Once a logical unit has sent the change direction indicator, it must be prepared to read until it receives a chain marked either *change direction* or *end bracket*.

In half duplex flip-flop mode, a secondary logical unit designed for interactive applications sends each chain with the change direction indicator set on, thus passing control of data flow to CICS/VS. The transaction may write many messages to the logical unit, and only when the transaction issues a read request will CICS/VS return control to the secondary logical unit. Thus CICS/VS does not release control of the data flow until the application program dictates it.

Even a simple SNA terminal, such as the IBM 3767 Communication Terminal, supports both the end bracket and change direction indicators. The keyboard unlocks to allow operator input only when a CICS/VS transaction issues a read request (change direction) or when a transaction terminates (end bracket). It locks whenever it sends a message to CICS/VS containing the change direction indicator. This common support of the SNA

indicators allows CICS/VS to control terminal operation on behalf of its transactions in a terminal independent manner.

Batch logical units can transmit large amounts of data in a single direction without interruption, as for bulk printer output and diskette input. However, SNA permits the receiving logical unit to interrupt the sending logical unit in order to transmit an urgent message. If the application program attempts to write data before a batch logical unit has sent the change direction indicator, CICS/VS sends the SNA SIGNAL command to the logical unit to request change of direction. Only after the change direction indicator has been sent will CICS/VS honor the WRITE request. This protocol is analogous to the support provided by CICS/VS for binary synchronous batch terminals through BTAM. However, use of the SIGNAL command does not halt all transmission on a multipoint line until the urgent message is sent, as did the equivalent pre-SNA *reverse interrupt* function.

In half duplex contention operation within a bracket, either logical unit can send data when the other is not sending. If the other logical unit also is trying to send a message, one of them (the primary logical unit, under CICS/VS) is required to receive the other's data before attempting to send. This protocol permits an application to send or receive at will, but recognizes that the secondary logical unit has limited buffer resources and cannot actually support a full duplex mode of operation. It is best suited to systems that can accept input messages and stack them for later processing, such as TSO, since with terminals like the IBM 3767 Communications Terminal, the keyboard is unlocked after every transmission when operating in this mode.

The full duplex protocol allows either logical unit to send at will. It depends on the function management layer or the application layer of each logical unit to control the relation, if any, between consecutive chains flowing in either direction. This transmission protocol provides scope for more sophisticated logical unit support, as exemplified by the CICS/VS pipeline facility discussed later.

CICS/VS system services

With BTAM-attached terminals, CICS/VS was responsible for network configuration, which it accomplished by means of the master terminal operator services. It was necessary, for example, to be able to take a terminal out of service for maintenance without also terminating CICS/VS. Also required was a facility for shutting down the whole network in an orderly manner at the end of a day. With the development of SNA, the responsibility for physical network configuration passed to the VTAM network operator. However, it is still necessary to control the logical data flow within CICS, so the master terminal functions use the appropriate SNA data flow control command.

The CICS/VS facility for taking a terminal out of service uses SNA commands to stop the data flow. A programmed logical unit can use the same commands to suspend the flow of data to it from CICS/VS, as when application data cannot be handled immediately because of operator intervention.

The CICS SHUTDOWN facility uses the SNA SHUTDOWN commands followed by a request to VTAM for session termination. The programmed logical unit can also request orderly shutdown of a session by sending the SNA REQUEST SHUTDOWN command. By recognizing these functions as basic to the operation of a teleprocessing network, and formalizing them as SNA commands, it becomes a simple matter to support them consistently for all logical units regardless of the characteristics of the application layer within it.

SNA errors

The definition of explicit functional layers within the teleprocessing network leads naturally to a formal definition of the errors that can occur in each layer, a consistent method of error reporting and correction, and a definition of error responsibility for each layer. The errors are divided into five categories: path errors, request header errors, state errors, function interpreter errors, and request rejection errors.

Path errors occur below the function management layer and essentially are not recoverable, from the point of view of the logical unit, since all possible recovery actions will have been attempted prior to reporting the errors. Request header and state errors generally result from a lack of enforcement of the rules agreed upon by the communicating function management layers in establishing a session and arise from incorrect specifications at system generation time. State errors denote that a message has been lost or become garbled during transmission, or that one of the function management layers has not accurately kept track of data flow within the session. The errors so far described can occur with any logical unit regardless of the application, and so can be handled independently of both the data content of the failing chain and the logical unit in question.

Function interpreter errors result from a function management layer's sending a request that the receiving function management layer cannot recognize. Usually such an error denotes a mismatch in system specification, as when the features of a particular logical unit are described incorrectly at system generation. Recovery depends on the data content of the chain in question. The final category of errors, request rejection, occurs when the function management layer recognizes a request but cannot carry it out. A common example is rejection of a print request because the printer is out of paper.

The reporting of request rejection errors depends on the capabilities of the function management layer associated with the logical unit. For example, logical units supporting visual display terminals will report different error conditions than will those supporting keyboard printers. Yet both kinds of devices may also share error conditions for which the recovery action is common.

CICS/VS error recovery

Every error condition that has a particular error recovery procedure is identified by a unique error code regardless of the logical unit generating the error. Thus, the error recovery procedures are totally independent of the logical unit. This independence allows CICS/VS to recommend default recovery actions for each error, on behalf of the transaction, in such a way that most users will not wish to modify this action via the CICS/VS error program exit. With BTAM support, in contrast, CICS could handle only certain line errors; most other errors had to be handled by the user's error program exit because recovery action depended on data content, device type, and line connection. Rationalization of errors within SNA has allowed CICS/VS to take much of the error recovery burden away from the user's error exit.

Within this error recovery framework, SNA also defines rules for reporting errors and methods of terminating multiple request-unit messages that are rejected by the receiving logical unit. This common requirement led to definition of the SNA CANCEL command to terminate a chain, and the LUSTATUS command to report asynchronous error conditions, so providing a consistent error recovery mechanism that is independent of the logical unit type.

CICS/VS uses an SNA error recovery protocol that ensures that it gains control of the data flow regardless of whether CICS/VS or the other logical unit reports the error in question. In most instances, the end user communicating with a CICS/VS application is a human operator, so when CICS/VS reports an error to the logical unit, it is in fact the operator who must take recovery action. The decision on what action to take depends on the CICS/VS application program, the nature of the data base being manipulated by that program, and the resources being utilised within CICS/VS itself.

Thus the best way to give the operator sufficient information is by means of an error message sent by CICS/VS. For example, an operator's input may be rejected either because CICS/VS temporarily lacks the resources to attach the requested transaction, or because the transaction has terminated abnormally. In the first case it is a simple decision to retry to the message, but in the second case, CICS/VS will have backed out any partial data base

changes so that the operator can continue submitting transactions. Such decisions are beyond the capability of a simple terminal designed to operate with many different subsystems.

When CICS/VS communicates with a programmed logical unit, it is the complementary application program in the communications controller that is interfacing with the ultimate device operator. Textual error messages are not well suited to program analysis, so in this case CICS/VS uses the LUSTATUS command to convey an error code to the logical unit to indicate the nature of the problem. Again, the SNA error recovery protocol used by CICS/VS permits the CICS/VS application program to take recovery action, or CICS/VS itself to send the end bracket indicator signifying transaction termination.

presentation services

Presentation services are those functions that support the specific requirements of end-user to end-user communication. Part of the function management layer, they are responsible for transforming application layer requests into the appropriate format for the designated end user.

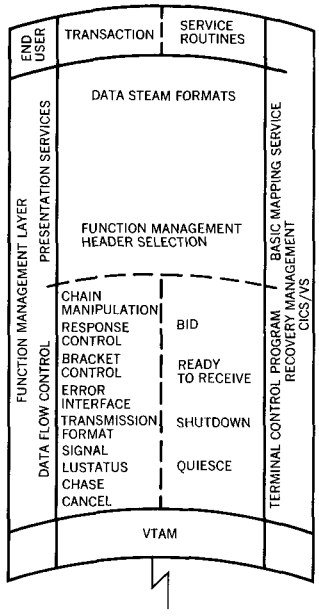
Within CICS/VS, a transaction can either provide its own presentation services or use those provided by CICS/VS. If the transaction provides its own services, it is written to use the CICS/VS terminal control READ and WRITE interface, and the contents of its message buffer must be formatted as required by the logical unit with which it is communicating. Using this level of interface, the application program is said to be device dependent and must be aware of the type of data stream required by the logical unit, whether an IBM 3270 Information Display System or a 3767 Communication Terminal, for example.

CICS/VS presentation services are provided by the basic mapping support (BMS) component, which in turn is built on the terminal control READ and WRITE interface. A CICS/VS transaction using BMS can be written independently of the device and data format, relying on BMS to construct the correct device dependent data stream.

The BMS interface also allows CICS/VS to select the appropriate output device. Thus a transaction can be written for output either to a card punch or to a printer, both represented by the same logical unit, and selection of the specific device will be transparent to the application. In SNA, the output device is selected by means of a function management header that preceeds the data and contains an identifier for the device required.

This selection mechanism is common across all logical units that support multiple output media. BMS formats the application request for the appropriate data stream, then calls on the CICS/VS

Figure 7 Responsibilities of the CICS/VS function management layer



data interchange program to append the appropriate function management header. The data interchange program also can be used directly by a transaction to format output for itself, as when transferring records to a data set in a programmable terminal controller. The data interchange program is responsible for maintaining the correct function management header sequences and for cancelling the device selection in case of abnormal termination of a transaction.

The SNA method of device selection is superior to pre-SNA methods because it does not depend on the line attachment technique (point-to-point or multipoint). The CICS/VS transaction was responsible for determining the appropriate technique for pre-SNA devices, which sometimes required selection characters in the data, and sometimes did not.

Figure 7 illustrates the CICS/VS components responsible for the SNA function management activities discussed above.

CICS/VS session control services

SNA session control functions are contained within the transmission subsystem layer, and they communicate with session control services above the transmission subsystem interface. In addition to its function management layer responsibilities, CICS/VS contains service routines for session initiation, termination, and recovery in the event of abnormal termination. These services communicate with the session control functions to supply the BIND parameters for the session, as well as the message sequence numbers and control commands for reestablishing the synchronization of sequence numbers between CICS/VS and a logical unit, called *message resynchronization.*

session recovery

During a session, outbound messages sent by CICS/VS and inbound messages sent by the logical unit are numbered sequentially. They provide:

- a means by which the receiver of a message can ensure that messages are not lost during transmission.
- a means by which the sender can correlate a response with the original message.
- a means of identifying the last recoverable message, for restarting when a message is lost.

When a CICS/VS transaction makes changes to a protected resource, such as a data base, CICS/VS makes the changes conditional upon the transaction's having been executed successfully to a synchronization point—that is, to a point at which a related sequence of events is complete. Such a sequence, called an

atomic unit of work, might be a data base read-and-update sequence initiated by a request for updating and terminated by a reply. Once a synchronization point is reached, CICS/VS commits the changes, thus making them permanent. If the transaction fails before reaching a synchronization point, CICS/VS will undo (or back out) all changes made to protected resources during the failed atomic unit of work.

A terminal message issued by a transaction during an atomic unit of work is deferred by CICS/VS until a synchronization point is reached, at which time the message is said to be committed and is transmitted to the terminal. If a session terminates abnormally, CICS/VS employs message resynchronization to ensure delivery of the committed message resulting from the latest complete atomic unit of work. Atomic units of work in progress at the time of session failure are defined as incomplete and are backed out.

CICS/VS performs resynchronization after an emergency restart, when a previous session has terminated abnormally, or when the logical unit requests it by means of the SNA REQUEST RECOVERY command. In any of these cases, CICS/VS sends the SNA set-and-test-sequence-numbers (STSN) session control command while the session's normal message flow is supended. The STSN command carries the sequence numbers of the last inbound message and the committed outbound message of the completed atomic unit of work. The logical unit's response to the STSN command indicates whether the committed message has been received. If not, CICS/VS sends the message again. The logical unit also can determine from the sequence numbers in the command that an active atomic unit of work has been backed out, and it can resume processing by sending the transaction-initiating message again.

Advantages of SNA to CICS/VS

In analyzing the advantages of SNA to CICS/VS, there are three criteria to be considered:

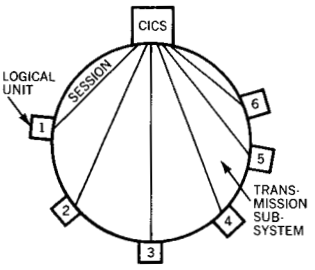
- existing functions that can be provided more effectively by SNA
- new functions that are available to, and being used by, CICS/VS users as a result of SNA
- ways in which SNA provides a base for the future evolution of CICS/VS.

The features of SNA discussed in the following paragraphs are examined with reference to those criteria.

transmission subsystem

A fundamental difference between SNA and pre-SNA communications systems is the clear division of responsibility in SNA between the function management layer and the transmission subsystem. Line management functions such as control of polling and recovery from line errors, which were performed for pre-SNA networks by terminal control, are handled within SNA by the transmission subsystem provided by VTAM and the Network Control Program. In addition, the network control functions performed by CICS/VS, such as modifying the status of a line or control unit by the master terminal function, are consolidated in SNA's system services control point.

Figure 8 SNA network: logical appearance to CICS/VS



CICS is unaware of the physical structure of the communications network. The advantages of this separation of function can be seen by comparing the amount of network definition information required by CICS/VS for SNA and pre-SNA networks. The terminal control table for an SNA network contains a list of logical unit definitions, each of which describes the characteristics of a logical destination within the communications system. The definition is unaffected by the way in which the logical unit is physically connected to the network. Figures 8 and 9 show how logical and physical SNA networks differ in their appearance to CICS/VS. In contrast, a pre-SNA terminal control table contains a complete definition of the communications network. Each terminal is grouped with its associated line or pool of lines, and line management information is associated with each line.

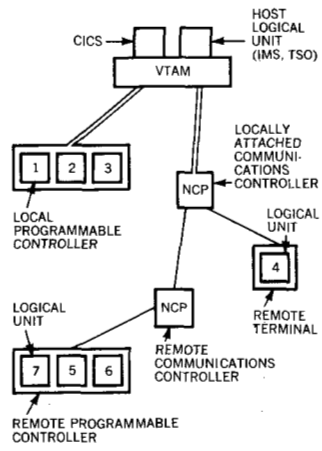
The interface defined by SNA, and provided by the transmission subsystem, for communication between two logical units, is independent of the characteristics of the network. The transmission subsystem is free to implement, in any way, the functions for which it is responsible, as long as that interface is maintained. In the transmission subsystem provided by VTAM and the Network Control Program, the functions of controlling, scheduling, and polling communications lines are implemented not in the host computer but in a special purpose communications controller. NCP/VS uses synchronous data link control, a line protocol more suitable to the requirements of SNA, to communicate with remote nodes in the network. As a user of the logical data paths provided by the transmission subsystem, CICS/VS benefits from the improvements in system performance brought about by the use of these network management functions, without any extra programming investment. In pre-SNA systems, on the other hand, CICS/VS support for each type of terminal depended on the type of connection, and implementation of a new line discipline required a separate programming effort for each type of connection to be supported.

function management

In pre-SNA networks, terminal control used communications line protocols to provide end-to-end control of data flow between a

CICS/VS application and a terminal. For example, contention for control of a half-duplex communications line was used to resolve the logical contention addressed in SNA by the bracketing protocol. This technique of mapping a logical function onto the characteristics of a physical link led to multiple, connection dependent implementations.

Figure 9 SNA network: physical appearance



SNA clearly distinguishes the data flows within a session between two logical units from the physical data paths between two network nodes in the transmission subsystem layer. The function management protocols defined by SNA provide a general set of end-to-end controls for these logical data flows. The use of these protocols to support the CICS/VS application program interface was discussed in the section on CICS/VS as a function management layer. It is worth noting that a number of protocols defined within function management by SNA, such as bracketing, closely parallel particular functions in CICS/VS. They represent a generalization of particular functions provided by CICS/VS for pre-SNA networks.

The advantage to CICS/VS of using these common function management protocols is that CICS need implement them only once. A given protocol is the same, in terms of the request header indicators and the data flow control and session control commands used, for sessions between CICS/VS and many different types of logical unit. When a session is initiated, CICS/VS chooses the appropriate subset of function management protocols and communicates them to the other logical unit. This facility gives CICS/VS a powerful way of dynamically tailoring the characteristics of a logical unit to the needs of a session with CICS/VS.

distributed processing

SNA provides CICS/VS with a framework for supporting distributed processing systems like the IBM 3600 Finance Communication System,[16] 3650 Retail Store System,[17] and 3790 Communication System.[18] Such systems allow the user to execute parts of his application, and store some of the application data, in the programmable controller at a location remote from a central computer. The ability to operate independently of a host computer, at least partially, allows the user extra flexibility in designing his application. Terminals attached to a programmable controller are under direct control of the application programs being executed in the controller, giving the user a high degree of local control over the interface between his application and the terminal.

CICS views a programmable controller as a number of separate logical units, each requiring a specific type of support. The functions required for each logical unit are indicated to CICS/VS by the terminal control table. CICS/VS support for each logical unit is independent of the method by which the controller node is

connected to the network—in fact CICS/VS is not even aware of which logical units are in the same controller node. Logical units in which the application layer is a user-written application program are referred to as programmed logical units. CICS/VS also supports fixed function logical units, such as the 3270 data stream compatibility logical unit in the 3790 System. The facilities provided by CICS/VS for fixed function logical units are less general, since the extent of the function in the application layer of the logical unit is limited.

programmed logical units

Typically, an application in a programmable controller interacts with a terminal operator and attempts to use local resources to process operator-requests. When a particular request must be processed by the host computer, the controller application invokes a CICS/VS transaction, and they then exchange data until the operator's request can be satisfied. At that time the result of the processing is passed to the terminal operator. The operator is not directly involved in the programmed logical unit's interaction with CICS/VS. In some cases the operator may not even be aware that a host interaction has occurred.

CICS/VS uses the same function management protocols to control sessions both with programmed logical units and with other, terminal oriented logical units. In a programmed logical unit, the application program can use the data flow control protocols of the function management layer to provide application function. For example, CICS/VS uses the BID command to request permission to automatically initiate a transaction. The decision to allow the CICS/VS-initiated transaction may be made by the programmed logical unit on the basis of the state of the application's processing or the availability of resources at the programmed logical unit, or the decision may be made by the terminal operator.

The presentation services processing provided by basic mapping support (BMS) for pre-SNA terminals is concerned with the creation of device dependent data streams that are suitable for delivery to a physical terminal. In a session with a CICS/VS application in which the end user is an application program in a programmed logical unit, the required presentation services may be independent of device considerations. The functions provided by the CICS/VS terminal control interface allow applications to exchange data without any restrictions on format. The application layer in the programmed logical unit assumes responsibility for any data stream formatting that is required for delivery of the output to its final destination. This destination may be a terminal operator, but it could be a data set or a line printer, depending on the nature of the application.

When communicating with a programmed logical unit, a CICS/VS application may choose to have device dependent processing done in CICS/VS to allow the application to operate compatibly with either a programmed logical unit or a simpler, terminal oriented logical unit. For a 3790 programmed logical unit, BMS builds SNA character string data streams that are suitable for presentation to devices controlled by the 3790 application. When communicating with a programmed logical unit in the 3600, BMS allows the CICS/VS application to associate a logical device with each output request. The logical device for the application data is identified by a function management header, thus allowing the CICS/VS application to distinguish among many logical destinations within a single programmed logical unit.

Associated with each logical device may be various properties such as paging status and page size, and, optionally, a 3600 device type such as the 3618 Administrative Line Printer or the 3604 Keyboard Display. The logical device provides the communicating applications with a high degree of flexibility since a logical device need not be associated with any physical destination within a logical unit, but could be used to identify particular types of processing to be performed on the data.

fixed function logical units

The 3270 data stream compatibility logical unit of the 3790 Communication System is an example of a fixed function logical unit within a programmable controller. Its device oriented level of function allows existing CICS/VS 3270 applications to communicate with 3270 terminals attached to the 3791 controller. Since it uses the same function management protocols (such as chaining and bracketing) as other terminal oriented logical units, like the IBM 3767, there is no need for duplicate device dependent logic within CICS/VS, as there was in pre-SNA terminal control for two different types of terminal.

However, the application layer in the logical unit can use the facilities of the function management layer to provide improved or new application function, in a manner similar to that of a user programmed logical unit. The compatibility logical unit uses bracketing protocol to control the sharing of a single 3270 printer among several logical units. The printer is used mainly by CICS transactions as a bulk printing device. Within a bracket, which corresponds to a single CICS/VS transaction, a printer is devoted to one logical unit. Between brackets, the bulk printing logical unit releases control of the printer, which may then be used by other compatibility mode logical units that control displays. These logical units use the printer to produce hard copy of a display image. The printer is re-acquired by the bulk printing logical unit when it receives a BID command, which informs the logical unit that CICS/VS wishes to initiate a new bulk printing transaction.

The presence of an intelligent presentation services layer within a programmable controller means that some functions, previously resident in BMS, can be moved into the controller. In the CICS/VS support for 3275 host conversational logical units attached to the 3650 Retail Store System, BMS builds 3270 data streams based on the map definition of the screen image, as it does for other 3270 terminals. The fixed portion of the screen image is not built by BMS. Instead, the map name, supplied to BMS by the application program, is appended to the output data sent to the 3650, where the fixed portion of the final screen image is supplied by a presentation services layer within the 3650. This distribution of function to the controller improves the use of the communications network by avoiding the repeated transmission of constant information.

batch processing logical units

Essential to the support provided by CICS/VS for distributed processing systems is the ability of CICS/VS applications to communicate with logical units that operate in batch mode. The programmable controller can be used for store-and-forward applications, allowing the user to exploit the ability of a distributed system to function independently of the host computer. In this case, the application in the controller consists of transactions that store intermediate results, typically on disk. Subsequently the results are transmitted to CICS/VS in a batch while local transactions are being executed. In the 3790 Communication System, the data are transmitted from the transaction data set by the batch logical unit. Data sent to the 3790 from CICS/VS can include updates to the application's files in the controller, as well as data for the print data sets maintained by the 3790.

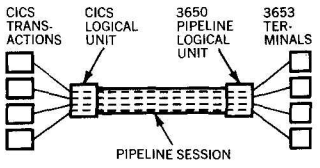
Support for the batch logical unit like that available in the 3790 is achieved by extending the logical-device concept so that a data set name can be associated with a logical destination. The selection of destinations, or components, is managed by the CICS/VS data interchange program using common function management headers, which provide for communication with batch terminals of limited function, such as the 3771 Communication Terminal, as well as with more sophisticated logical units like the batch logical unit in the 3790 Communication System. With batch logical units that have the appropriate level of function management capability, CICS/VS applications can use the message protection and resynchronization facilities in CICS/VS to prevent the repeated transmission of complete data streams after a session failure.

full duplex logical units

The flexibility of SNA's function management protocols allows CICS/VS and a program in a programmable controller to use a single session to service many similar requests from a large pool of identical terminals, thereby minimizing the use of the communications system's resources. The secondary logical unit trans-

mits requests and replies from a large number of terminals through the single full-duplex session, referred to as the CICS/VS *pipeline session.* CICS/VS takes each inbound message and attaches a task to it, as illustrated in Figure 10. Each task can issue one WRITE request, which CICS/VS sends immediately. The transaction can send only one reply before terminating. These restrictions allow a high volume of simple one-in and one-out transactions, all from a single logical unit, to be processed in parallel. Typical applications for this type of logical unit include credit validation transactions using point-of-sale terminals. The CICS/VS pipeline facility is available to any logical unit that obeys the correct function management protocols and is independent of any particular controller.

Figure 10 CICS/VS pipeline session



Summary

The explicit definition of the function management layer by SNA allows CICS/VS to use a uniform set of protocols for controlling communications between CICS/VS applications and destinations in a telecommunications network. The definition of the communications system in terms of discrete functional layers in every node means that the same protocols apply both to sessions with programmed logical units in distributed processing systems and to sessions with simpler, terminal oriented logical units. The clear division of function among the application, function management, and transmission subsystem layers, together with the concept of independent parallel conversations between paired layers, allows the capabilities of each layer to increase without affecting the capabilities of the other layers. These aspects of SNA have permitted CICS/VS to implement its data communications functions in a manner that can be logically adapted and extended to meet the future needs of users' application.

ACKNOWLEDGMENTS

The combined efforts of many people have contributed to the development of CICS/VS. The authors wish to acknowledge the principle contribution of the IBM Programming Center in Palo Alto, California, to the initial SNA implementation of CICS/VS, the assistance provided by members of the IBM Program Product Center in Sindelfingen, West Germany, and the work of those at the IBM Programming Centre in Hursley, England, who extended the initial design and implementation. In addition, the authors wish to thank their IBM colleagues who assisted in critical reviews of this paper.

CITED REFERENCES

1. James Martin, *Design of Real-Time Computer Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1967).
2. W. A. Clark, "The functional structure of OS/360—Part III: Data management," *IBM Systems Journal* **5**, No. 1, 30–51 (1966).

3. *Customer Information Control System/Virtual Storage (CICS/VS) General Information*, order number GC33-0066, IBM United Kingdom Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire SO21 2JN, England.
4. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15,** No. 1, 4–23 (1976).
5. Herbert Hellerman and Thomas F. Conroy, *Computer System Performance*, McGraw-Hill Book Company, New York (1975), Chapter 8: "The IBM OS/360 Operating System."
6. M. A. Auslander and J. F. Jaffe, "Functional structure of IBM virtual storage operating systems—Part I: Influences of dynamic address translation on operating system technology," *IBM Systems Journal* **12,** No. 4, 368–381 (1973).
7. D. G. Keehn and J. O. Lacy, "VSAM data set design parameters," *IBM Systems Journal* **13,** No. 3, 186–212 (1974).
8. J. P. Birch, "Functional structure of IBM virtual storage operating systems—Part III: Architecture and design of DOS/VS," *IBM Systems Journal* **12,** No. 4, 401–411 (1973).
9. *Customer Information Control System/Virtual Storage (CICS/VS) Application Programmer's Reference Manual*, order number SC33-0077, IBM United Kingdom Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire SO21 2JN, England.
10. W. C. McGee, "The information management system IMS/VS," *IBM Systems Journal* **16,** No. 2, 84–168 (1977).
11. For a description of the interface provided by BTAM, and the corresponding link control sequences, see *OS/VS BTAM*, order number GC27-6980, IBM Corporation, Programming Publications, Department 63T, Neighborhood Road, Kingston, N.Y. 12401.
12. P. G. Cullum, "The transmission subsystem in Systems Network Architecture," *IBM Systems Journal* **15,** No. 1, 24–38 (1976).
13. H. R. Albrecht and K. D. Ryder, "The Virtual Telecommunications Access Method: A Systems Network Architecture perspective," *IBM Systems Journal* **15,** No. 1, 53–80 (1976).
14. W. S. Hobgood, "The role of the Network Control Program in Systems Network Architecture," *IBM Systems Journal* **15,** No. 1, 39–52 (1976).
15. R. A. Donnan and J. R. Kersey, "Synchronous data link control: A perspective," *IBM Systems Journal* **13,** No. 2, 140–162 (1974).
16. *Customer Information Control System/Virtual Storage (CICS/VS) IBM 3600 Guide*, order number SC33-0072, IBM United Kingdom Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire SO21 2JN, England.
17. *Customer Information Control System/Virtual Storage (CICS/VS) IBM 3650 Guide*, order number SC33-0073, IBM United Kingdom Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire SO 21 2JN, England.
18. *Customer Information Control System/Virtual Storage (CICS/VS) IBM 3790 Guide*, order number SC33-0075, IBM United Kingdom Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire SO21 2JN, England.

Reprint Order No. G321-5054.