APPC/MVS distributed application support

by F. W. Voss

Advanced Program-to-Program Communication for Multiple Virtual Storage (APPC/MVS) is a major evolutionary change to MVS for applications that need to connect and communicate across the enterprise. APPC/MVS is an implementation of Systems Network Architecture (SNA) LU 6.2 session-defined protocol. This new MVS environment includes services to enhance the creation, execution, and management of MVS peer-to-peer and client/server applications. In addition to providing connectivity and communications services, APPC/MVS also has scheduling facilities for managing concurrent work originating from other systems in the enterprise network. The objective of this paper is to survey these facilities by examples of usage and by relationships to existing MVS services. Topics include various types of distributed models and approaches, design considerations, and characteristics that represent candidates for an APPC/MVS implementation. Relationships of the APPC/MVS environment with the Customer Information Control System/Enterprise Systems Architecture (CICS/ESA™), Information Management System/Enterprise Systems Architecture Transaction Manager (IMS/ESA® TM), Time Sharing Option Extensions (TSO/E), and batch environments are included.

Events in the world of computing and communications technology over the last few years can only be described in adjectives that denote rapid acceleration and extraordinary change. Evolutionary progress in chip power and miniaturization continues to improve information processing tools and possibilities for businesses to grow competitively in today's global markets. Advances in desktop and laptop capabilities, for example, continue to increase the capabilities and power of the end user. This user power is further enhanced through workstation connectivity to local area networks and wide area networks that allow work groups and departments to focus on new business needs through collaboration and information interchange.

A key distinction from the world of yesterday's terminals is that workstations break the historical attachment limits of a terminal to a single host. The advances in program-to-program communications technology support this break as multiple concurrent connections can be made between application parts located in any number of systems. Applications can communicate with each other on demand, which means the information resources of an organization can immediately flow to where they are needed. Information technology exists to deliver information to the end user, including chief executives, business and technical product makers, and the people who use applications that drive today's businesses. As remote information sources become more accessible, communication becomes a powerful competitive business tool. The right information, on time, to the right desktop, can help determine the future prospects of a competitive corporation.

Workstation and mainframe differences. The workstation maximizes the productivity and performance of a single user by providing direct, exclusive, nonshared access to "local" resources.

When the workstation can "plug into" the resources of an enterprise, we encounter new issues and concerns. These arise from the clash of opposing philosophies: that of a workstation end user's view of complete individual freedom versus the concerns of a business. Corporations with existing Multiple Virtual Storage (MVS) applications and data investments have a continued need to assure that key business assets are managed with a high degree of privacy, security, and integrity control as they adopt distributed approaches. Further, the work demands that originate from remote workstation users cannot be predicted with any accuracy, and these requests need a reasonable degree of resource management usage in order to accommodate distributed work on existing production systems. These concerns have been addressed in IBM's Advanced Program-to-Program Communication for Multiple Virtual Storage (APPC/MVS) solution. The challenge is to take advantage of the attributes of both the workstation and mainframe by synergistically combining them into a more competitive business advantage with appropriately designed business applications.

While the workstation evolution is on a path to provide low-cost instruction execution for a single user, the architectural evolution of the mainframe will continue in a direction that supports multiprocessing for large numbers of users who must share common data. Mainframes differ from workstations in their storage subsystem architecture, which is focused on concurrent data transfer. A mainframe IBM System/390* is comprised of multiple execution processors and scores of channels and input/output (I/O) devices. This large collection of microprocessors is designed to execute many pieces of work in parallel and supports high-bandwidth concurrent data transfers between multiple I/O processors (channels) and main storage. At any instant, Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA*) may be executing multiple applications and moving data at multimegabyte-per-second rates across many channels and devices. With channels potentially in the hundreds and external storage devices in the thousands, the capacities of a System/390 are readily distinguished from that of a programmable workstation. Many existing systems today support 5000 or more concurrent users sharing and updating very large databases in excess of 300 gigabytes (one billion or 10^9 bytes). Hence, the mainframe with attendant software retains a historical advantage of managing and integrating large databases, and with connectivity allows businesses to both aggregate and disseminate key information across the enterprise. An important direction for MVS/ESA will be to continue to enhance connectivity and communication services for applications.

The MVS/ESA operating system (hereafter referred to as MVS) is also not a single entity but rather a collection of complementary subsystems that include: Customer Information Control System/Enterprise Systems Architecture (CICS/ESA*), Information Management System/Enterprise Systems Architecture Transaction Manager (IMS/ESA* TM), Time Sharing Option Extensions (TSO/E), and DATABASE 2* (DB2*). (For simplicity, CICS will be used to mean CICS/ESA, and IMS will be used for the IMS/ESA transaction manager.) Each is designed to take advantage of multiprocessing attributes of a System/390 and multiprogramming capability of MVS/ESA. MVS and its companion subsystems have a successful history in addressing growing business needs for data sharing with high availability, integrity, security, and storage management across vast networks of end users. While access to host data is provided by transaction products, APPC/MVS also provides for writing a similar style of application.

Connection on demand with any remote target application is a feature of the APPC architecture.

Early transaction systems supporting display terminals used dedicated and continuously maintained connections. Workstations, however, do not necessarily need continuous connections. Connection on demand with any remote target application is a feature of the APPC architecture and available with APPC support in CICS, IMS, and APPC/MVS, as well as a myriad of other products supporting APPC.

Transmission speeds for data communications have been increasing by 50 percent per year since 1980.[1] Furthermore, we can expect to see two orders of magnitude improvement in bandwidth before the end of the century. This increased ability to pipe data will open significant opportunities for a broader scope of applications to benefit both the individual and the corporation. Applications such as PRODIGY** and emerging multimedia applications are but glimpses into the future of what is possible through the synergy of workstation, networking, and host relationships. A key role of the mainframe will be to integrate and manage very large data reservoirs and to provide access to users wherever they are located.

Future application possibilities. Easy connectivity and the unfolding availability of very high bandwidth fiber optic cable opens new application possibilities. While compact disk read-only memory (CD-ROM) can deliver large amounts of information locally, users must anticipate information needs in advance and have the required CD on hand.

Recent gains in digital image compression and full-motion digital video technology have been dramatic. Standards are being defined and implemented in specialized compression and decompression chips to offer image compression ratios exceeding 100 to 1, allowing for economical storage and faster transmission with no visible loss in quality.[2] According to Negroponte, these techniques will permit the shipment of one hour of video over a fiber running at a gigabit per second, in only five seconds.[3] Image transfer across high-speed networks along with these performance improvements in computing systems will allow industry and academia to make headway on grand challenge applications.[4] These include, for example: climate modeling, pollution dispersion, human genome, ocean circulation, semiconductor modeling, and superconductor modeling. In addition, new capabilities will be available to the home shopper, the home mechanic, schools, and laboratories.

Dynamic, as-needed reference will be able to offer new home service information utility opportunities. The future home owner or car owner who needs to do an immediate plumbing or automobile repair job will benefit from on-line connectivity. With connectivity to an advanced host multimedia application, a "mini how-to video" from a video library could be imported to local intelligent workstation storage for immediate use. Future purchase decisions will be possible through home on-line catalog access. Prospective buyers will be able to enter decision parameters such as price ranges, colors, and other specifications for product selection. Choices will be demonstrated on line, along with consumer satisfaction ratings.

The possibilities for education are endless. Future physicians and surgeons can have access to successful operating procedures and the latest techniques in their field of study on line. Through the use of host artificial intelligence (AI) services, engineering designs can be stress-tested and analyzed, geologists can evaluate seismic data or satellite images for natural resource management decisions, and doctors will be able to gain computer-assisted analysis of medical scans or DNA sample evaluations for genetic disorders. Collaborative real-time video conferences can be held on the "workstation of tomorrow" by professionals who need to solve and deal with immediate business, scientific, engineering, or medical problems. The potential for using information technology to benefit humanity and our planet has never been greater.

Future emerging multimedia applications will demand concurrent access to very large specialized libraries that will be globally distributed. These libraries of tomorrow will contain images, video, voice, text, and data to be made available to future workstation users at the click of a mouse.

This paper discusses the features of APPC/MVS and their application to both end-user needs and corporate needs in the distributed world. We begin with an explanation of some key concepts behind APPC technology. Particular features of APPC/MVS are described, along with scenarios of usage. The aim is to show how APPC/MVS sets the stage for new distributed application possibilities. Relationships between APPC/MVS and existing subsystems are described and application attributes that can take advantage of APPC/MVS features are defined.

Concepts and terminology

SNA and LU 6.2. A Systems Network Architecture (SNA) network is a collection of geographi-

cally dispersed systems that support a common architecture for connectivity and communications. Each system provides one or more ports or logical units (LUs), and each LU has a name. The LU name can be thought of as the place where applications, each with its own unique name, reside. For each LU, SNA products provide a connectivity and data transport interface to the network and a programming interface for use by application programs. This arrangement separates and "hides" physical networking details to make the job of writing applications easier. Early SNA support was geared to address the needs of devices, such as the IBM 3270 information display systems that depended on continuous connectivity to a host. To support distributed processing and dynamic connectivity on a peer-to-peer basis, a new approach to communications was needed. This led to the development of the APPC architecture. LU 6.2 session-defined protocol and APPC are one in the same because LU 6.2 is the formal definition for a set of program-to-program communication services. APPC consists of two distinct parts: a set of communication services with their application programming interfaces (APIs), and the formats, content, and rules for data flow (the protocol). The APIs define the semantics of the interface to these communication services and not the underlying protocols. This abstract API is often referred to as the APPC verbs or requests, the APPC interface, or just APPC. In this paper the usage of APPC and the term LU 6.2 refers to the protocol portion of the LU 6.2 architecture. The *protocol* defines the allowed sequence of verbs to allow initial connection, data exchanges, and disconnection between the partner programs.

APPC consists of a set of communication services with APIs, and formats, content, and rules for data flow.

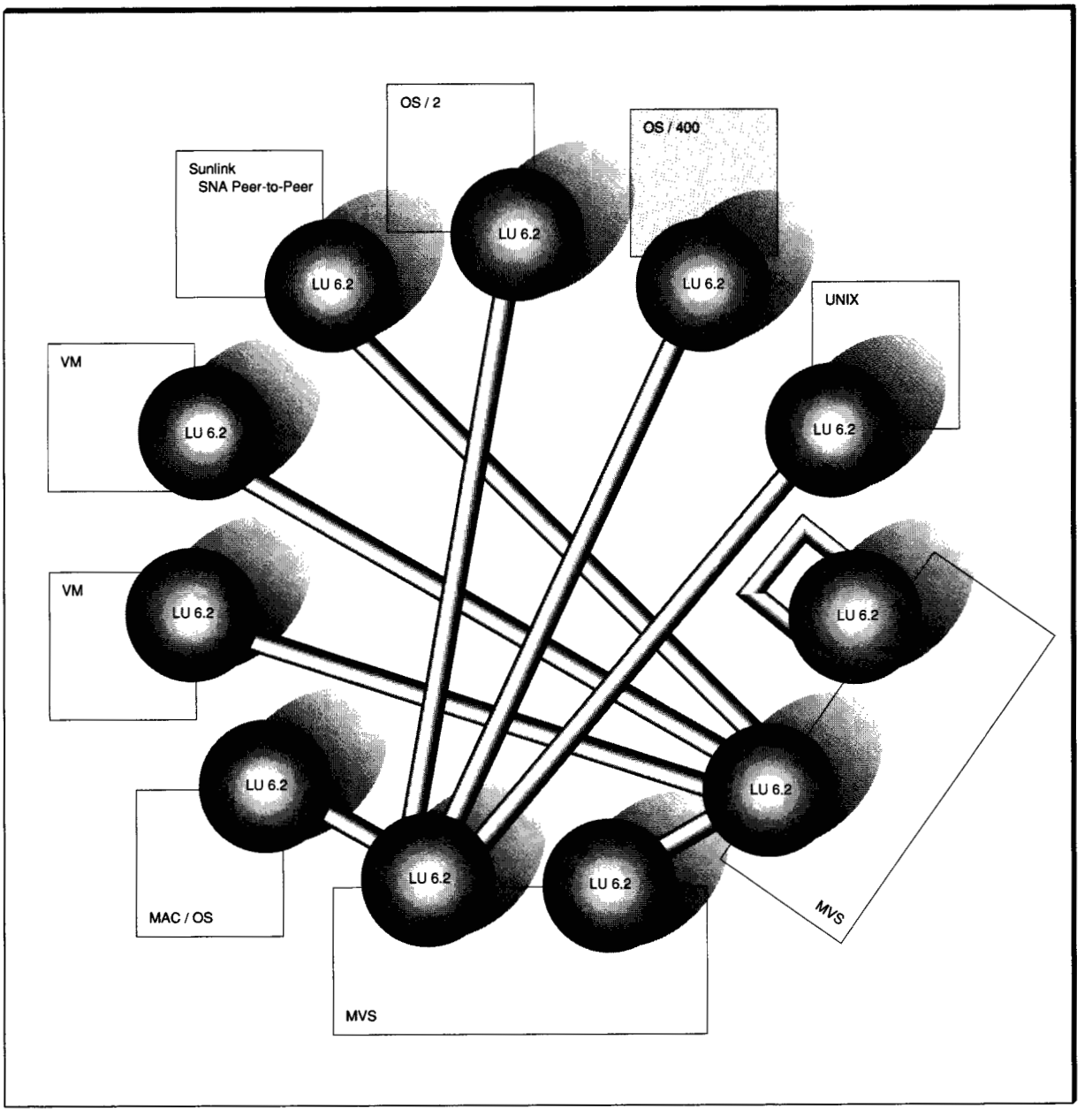
APPC also supports the dynamic activation of partner programs. Partner references are usually coded by name and communications products supporting SNA connections between the two locations. In Figure 1, we see an SNA network of connections between like and unlike systems. APPC is a well documented and widely accepted industry standard and many non-IBM systems also provide APPC support. In the figure, each of the connecting lines shows a possible LU 6.2 conversation between two applications.

Conversational model. APPC supports a *conversational model* for information exchange. This model allows for a variety of exchanges between two or more programs. Included in the protocol are application services to request a conversation, send and receive data, control and synchronize exchanges, and end conversations. Types of exchanges supported by the model include, for example, a single exchange, a continuous interactive exchange, or a continuous flow of data from one partner to the other. The model is analogous to the telephone system where the duration of a conversation can vary depending on the amount of information exchange needed.

From a workstation end-user perspective the activity between partner programs happens perhaps in response to a screen selection.

The process starts when an application executes an APPC ALLOCATE_CONVERSATION call or connection request. (See Figure 2.) Once this request is issued, the operating systems do the rest of the work without application involvement. Communication components respond by establishing both a logical and a physical network link or connection between the LUs, called a *session*. A session is needed before SEND and RECEIVE requests can exchange data between the partners. Session startup is a relatively costly process in terms of network linkage setup and handshaking exchanges between the LUs. Data exchanges flow over a *conversation* within the existing session and, relatively speaking, with good performance once the session has been established. Since an MVS application may use several conversations at once, each is logically represented by an internal name on SEND and RECEIVE requests. The process ends when either one of the partners terminates the conversation. This model then is geared for dynamic exchange when required, and can be utilized by both workstation and host systems. It is significantly different from the world of 3270-

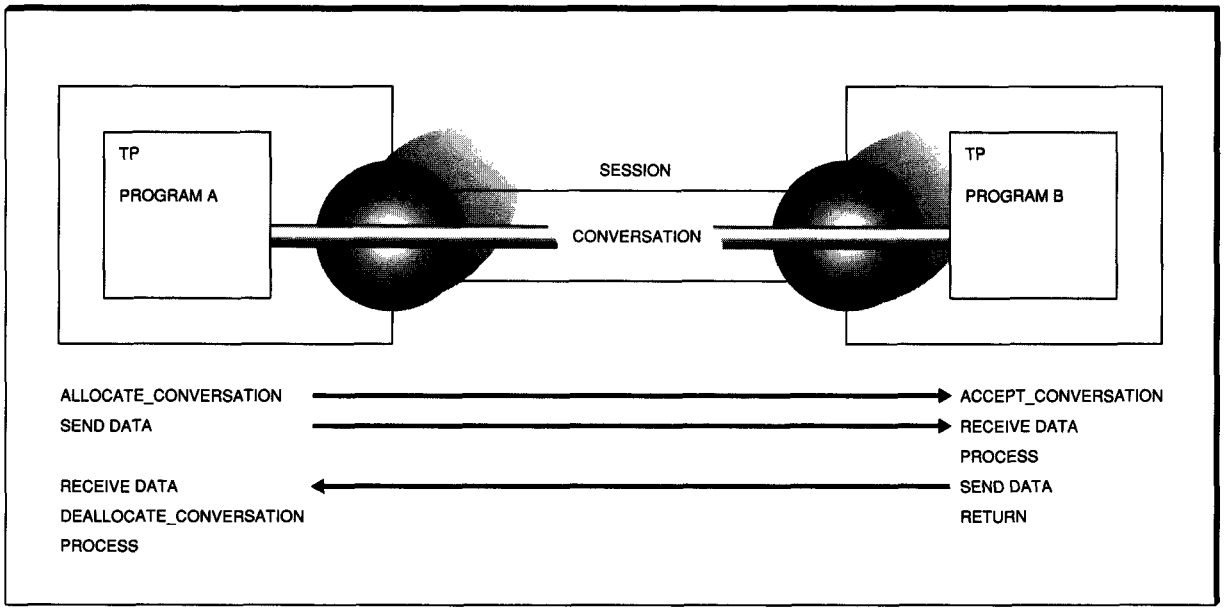
Figure 1 An SNA network



type terminals that are dependent on dedicated sessions to a host.

Initial session establishment includes an exchange of LU capabilities so that each is aware of its partner's level of support. During this exchange for APPC/MVS, an authentication process also occurs. An exchange of security information using encryption algorithms is performed to assure that an originating LU is authorized to establish sessions. This authentication process is defined and controlled by a systems administrator, in this case IBM's Virtual Telecommunications Access Method (VTAM*), and a security

Figure 2 A conversation between two applications



product such as IBM's Resource Access Control Facility (RACF).

The conversational model then, supports a series of synchronous exchanges between partners. These concepts are illustrated in Figure 2. The partners require programming design in tandem and this makes the approach more difficult to design, develop, and test. A SEND by one partner must be matched in logic by a RECEIVE in the other. Better tools are needed to assist this design and development process. The model is best employed by applications that can take advantage of this powerful dialog facility. Despite the current difficulties in programming an APPC application, the paradigm is able to deliver considerable productivity to a very wide variety of end users once the application is completed.

Furthermore, networks are not always reliable and use of APPC can assure that critical business applications, e.g., fund transfers, are implemented with a complete handling of all possible error conditions. Confirmation, synchronization control, and error notification allow very robust applications to be built.

Other models. There are two other current programming communications paradigms, namely Remote Procedure Call (RPC) from the world of UNIX** and "message passing" or Message and Queuing (M/Q).

Briefly, RPC supports the automatic connection to a remote service when an application executes a call for that named service. Events usually occur synchronously, similar to that of a main line program invoking a subroutine, only in this case the subroutine resides on another system. Automatic network connections are made by the operating systems on behalf of the requestor, and parameters are passed for processing to the named server. Results from completed server execution are then returned to the requesting application for continued processing. One essential difference between APPC and RPC is that RPC is an easier-to-use model when single information exchanges can meet application needs.

Announcements have been made regarding IBM directions to support the Open Software Foundation's Distributed Computing Environment, as well as to provide Portable Operating System Interface for Computer Environments (POSIX**) services support on MVS. Distributed Computing Environment includes RPC, and POSIX is a set of standard application and system services from the world of UNIX. POSIX is the work of a com-

mittee, and it represents a set of open standards for an operating environment.

In the Message and Queuing (M/Q) paradigm, a client application sends messages to a remote service for processing. The message usually contains the target service name along with information to be used by the server. Messages are handled asynchronously by operating system message queuing services at both origin and destination. System support routes the message to the server, which in turn issues a RECEIVE to acquire the message. From a programming perspective, M/Q is an easy-to-use model where the interchange of relatively short messages meets application needs. Application logic in each partner must be sensitive to message format for successful communication.

Each approach has its place in the information exchange realm, just as humans use a variety of communication mechanisms such as the telephone, electronic mail, and postal services. Further, while SNA is the networking avenue used by APPC, networking protocols such as Transmission Control Protocol/Internet Protocol (TCP/IP) and Open Systems Interconnection (OSI) are other key transport facilities in use today. RPC is supported by TCP/IP, and the M/Q paradigm has been implemented on many products using TCP/IP, SNA, OSI, and other network transport facilities.

Because APPC is supported across all MVS environments, e.g., batch, it is possible to write an application that could connect to and interoperate with an OSI network, a TCP/IP network, and an SNA network. Such programs could serve as a router, gateway or translation application for communication between different partners.

APPC/MVS programming support. One level of programming support in APPC/MVS is a set of callable LU 6.2 verbs.[5] This level is appropriate for systems programmers or product builders as it provides for explicit programming control of communications options. For example, APPC/MVS has extensions to handle asynchronous LU 6.2 requests and locate mode. *Locate mode* permits the use of buffers in data spaces for send or receive requests. A second level of programming support is IBM's Common Programming Interface for Communications.[6,7] This is a standard subset of LU 6.2 services for application builders that is also available across Operating System/2*, Disk Operating System (DOS) and Windows** workstations via Networking Services/DOS, Application System/400*, and Virtual Machine/Enterprise Systems Architecture* (VM/ESA*) operating systems as well as many non-IBM systems. In general, since APPC is hardware-independent, APPC applications may be written without any concern about the system in which their partner is executing.

In addition to providing SEND and RECEIVE data commands, the APIs also provide control services such as request for confirmation, confirmation of success, reporting of errors, request and granting of permission to send, starting a new transaction, and ending a transaction. Figure 2 illustrates the use of some of the more commonly used APPC verbs between two partner programs. The sequence and flow of exchange has a strong similarity to the use of a telephone between two people.

Exchange between programs written with the Common Programming Interface for Communications (CPI-CI) or the LU 6.2 programming interface is supported. An APPC/MVS application may freely intermix LU 6.2 and CPI-CI verbs as needed; however, if there is need to port the application to a different system, the use of CPI-CI is recommended.

Distributed versus cooperative. *Distributed processing* is where parts of an application are distributed between two or more systems. For example, a distributed application would be useful for concurrent data exchange between multiple systems. Either partner may be designed to initiate the APPC connection request. A distributed application may be more complex than illustrated with Figure 2; that is, it may concurrently communicate with multiple partners. An example is an MVS batch application that updates a set of remote libraries with common information. In this case, a single client application on MVS may request connections to server partners on workstations or local area network servers to perform parallel updates. In this paper, *cooperative processing* is the combining of the strengths of the workstation with the strengths of the MVS host by having parts of the application run in both places. In cooperative design, the workstation usually initiates the connection request. Cooperative processing is a subset of distributed processing.

IBM's Virtual Telecommunications Access Method (VTAM) defines programs that use APPC calls as *transaction programs* or TPs. In this paper, a TP and APPC application are the same. While APPC/MVS schedules an application based upon an external APPC stimulus, this does not imply that all applications using APPC are transactions as defined by the transaction processing metrics associated with CICS and IMS, e.g., response time or numbers of transactions per second.[8]

Clients and servers. In this paper, a *client application* initiates an APPC connection request and a *server application* receives the request. While many of the needs for remote processing are currently addressed by CICS and IMS transaction products, APPC/MVS addresses a class of applications that are beyond the scope normally considered for a CICS or IMS transaction. While APPC/MVS supports any class of application, from simple to complex, its key role is to support applications that can take advantage of host resources, for example, application servers that need considerable processing logic (e.g., millions of instructions), scores of data references or large numbers of connections, or high volumes of continuous communications traffic. Additionally, APPC/MVS applications can use all of the existing native MVS services to supply more complex client or server functions.

APPC/MVS goals

With APPC/MVS a host can meet the needs of directly connected workstations, connected local area network servers, and peer-to-peer connectivity to other host systems. It is to this end that APPC/MVS was developed, that is:

- To provide services for a broad suite of new application servers that can take advantage of access to host data as well as MVS[9,10] attributes of system integrity, resource access security, multiprocessing, storage management and capacity
- To aid the development of MVS "client" applications for the management, security, and control of information assets of the enterprise
- To support the enterprise need for handling increased demands for information storage and the place where administrative focus can be applied to the distributed enterprise
- To supply an environment for development of specialized processing and compute servers (e.g., knowledge base, array processing) that can be combined with desktop processing to deliver increased productivity for end users

This paper explores how APPC/MVS, introduced in Multiple Virtual Storage/System Product Version 4.2.0, more readily "opens" accessibility to MVS applications and data resources through support of connectivity and industry standard APPC application programming interfaces (APIs) and protocols.[11–15] APPC protocols offer advanced features such as security, confirmation flows, and error handling. The same is true for connected workstations, local area network and wide area network environments, and between MVS applications running in the same or different MVS systems.

Support for APPC has been available in MVS with Advanced Program-to-Program Communication/Virtual Telecommunications Access Method, and APPC/MVS builds on this support to provide additional capability. Key distinctions between this earlier support and APPC/MVS are:

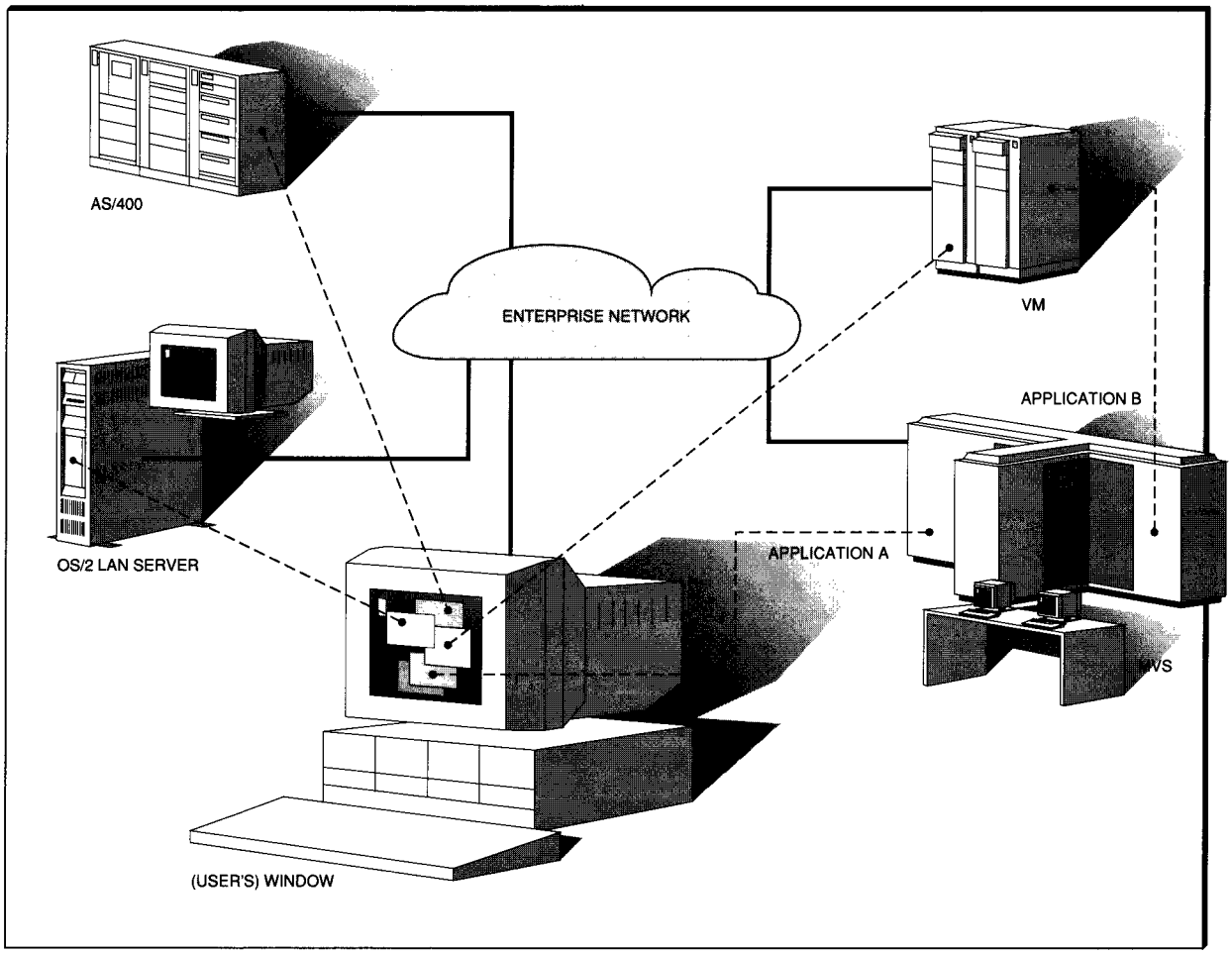
- APPC/MVS manages scheduling of concurrent requests with workload balancing.
- Application coding is easier through the CPI-CI APIs.
- High-level language calls are available for APPC services.
- APPC services are available to multiple MVS environments, e.g., batch, IMS, and TSO/E.
- Facilities are included for traditional MVS features such as accounting, resource allocation, and security control.

Using APPC/MVS

Window to the enterprise. A workstation with multitasking allows the user to concurrently work with icon selection methods and use multiple windows. Behind each window is an application that displays the data needed for a selection or work-related results. Cooperative applications allow windows to show data and compute results of information anywhere in the enterprise.[16,17]

Local workstation applications using APPC can transparently connect to other named applications in the enterprise. Figure 3 illustrates this principle by showing a variety of connections. From the point of view of the end user, connec-

Figure 3 A window to the enterprise

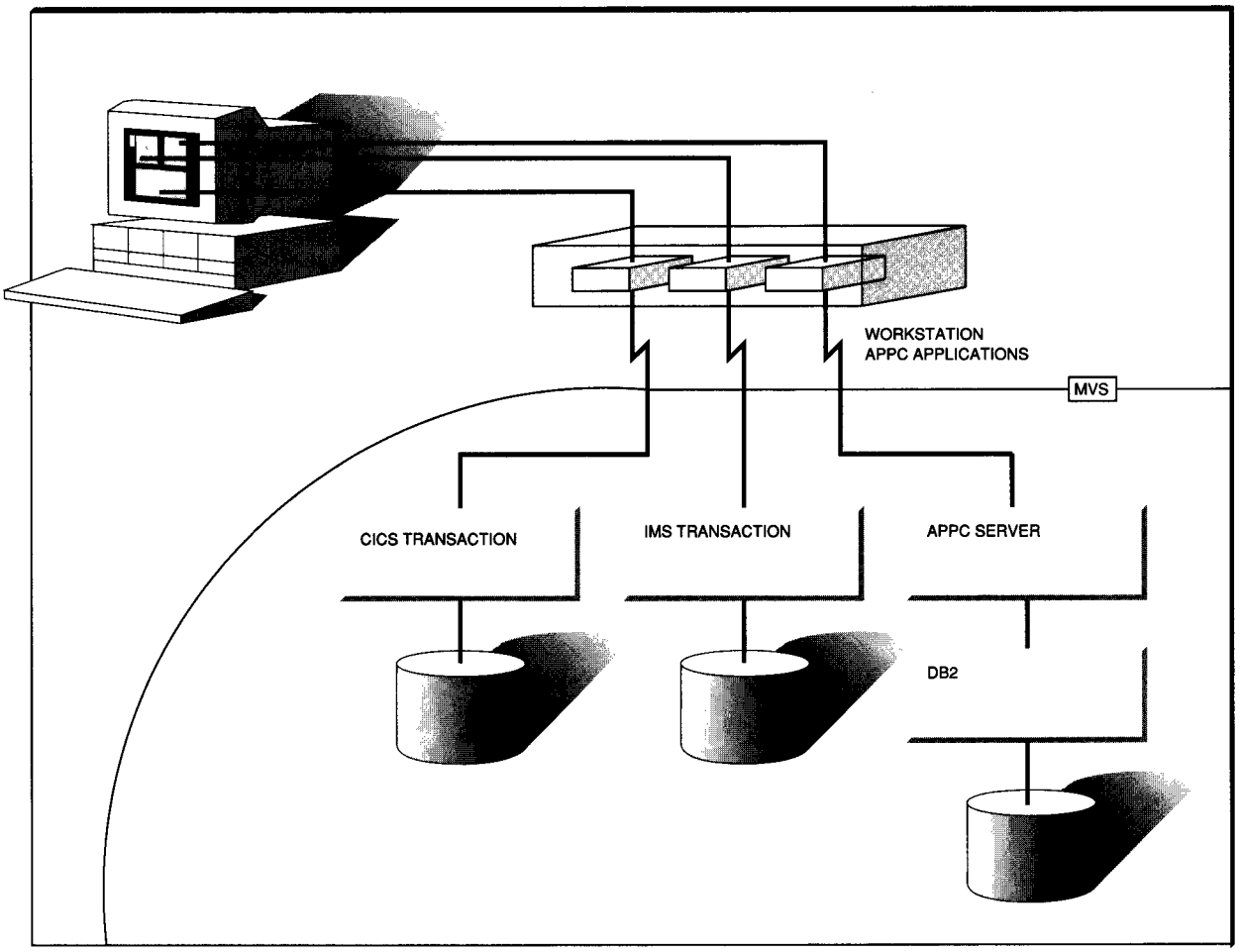


tions to remote services occur transparently. Transparent invocation gives the user a "single-system image" of the network resources.

The developer of a cooperative application needs to design the client part to display information in a window, along with the logic to handle user interactions. Based on user interactions with windows, application logic would invoke a named host application as appropriate.

A key workstation application is *distributed file management* (DFM).[18] With availability of MVS DFM, local workstation applications will be able to transparently reference remote MVS files. In the case of Operating System/2, the access method will dynamically invoke communications management for connecting to the appropriate MVS access method partner. The MVS DFM *agent application* will allocate the needed MVS files and perform data record SEND and RECEIVE to the workstation application as needed. The capability exists for concurrent access to multiple databases by one or more workstation applications. Several workstation client applications can work concurrently with different partners executing in the MVS system, e.g., one could execute under CICS, another under IMS, and a third under APPC/MVS. This data integration case is illustrated in Figure 4. Data from each subsystem can be related, compared, and displayed by workstation applications.

Figure 4 APPC applications



Design considerations. When a distributed or cooperative application is needed, the chore for an application designer is to decide which functions are to run on each platform. In the general cooperative case, application processing and window operations would be relegated to the workstation, while large amounts of data access and handling (e.g., sorting) would be performed by the APPC/MVS server. Workstation configurations on which the application is to run may also dictate how much logic is to be performed at the workstation versus the host.

Data location. Scherr states that "as a general rule, users and the applications and the data they use should, whenever possible, be placed in the same data processing node." [19] This guideline however, is subject to several factors that include: the application's requirements for data that for business reasons must remain remote, and the resource constraints (i.e., the minimum storage configuration) of the standard workstation in the enterprise. Rapid improvements in bandwidth and connectivity in the next few years (e.g., fiber technology and gigabit networking) will foster designs that access remote data. The ease at which data will be able to flow where needed further reduces the requirement for vast amounts of local workstation storage. Hence, one approach will be to pipe needed data from a host for local independent processing. Others that require interchange will be described later.

Portability. Application portability enables growth to larger capacity systems while protecting user programming investments. When an application is written using high-level languages and CPI-CI, a high degree of application portability between systems is possible. As an example, an Operating System/2 or Application System/400 server could be recompiled to execute on a larger system, while leaving the client parts of the application untouched. Of course, system directories would have to be updated to reflect the new target site of the server, and some additional changes may be needed to accommodate file access differences. Portability allows users to more readily change their enterprise network configuration by upgrades or additions of new systems to match business needs.

Security. MVS data assets are protected from various forms of intentional or unintentional threats from unauthorized access, or modification. Specific features are available in MVS to assure that only properly authorized users can read, write, create, or delete information. The Virtual Telecommunications Access Method session bind process prohibits unauthorized systems from connecting to an MVS system. When an application is established in APPC/MVS, the execution environment inherits security information defined by the installation for the individual using the client application. Unauthorized users are prevented from using one or more application servers. This also helps insure that application execution will not exceed the authority granted to the user for sensitive resource access.

Applied conversations. The following are examples of using the conversation model in MVS.

On-demand connect, simple. This model is used for most client/server applications. "Simple" in this case implies a single information exchange where results can be returned fairly quickly. Parameter-driven service requests such as a table lookup or complex query from MVS files fit this model.

On-demand connect, complex. This model would apply when a dialog of multiple send and receive exchanges is needed. For example, a request is made to a library search service to acquire all abstracts of documents matching a set of keywords. A database is searched and the set of abstracts is delivered for workstation display. The end user browses the abstracts and identifies the documents desired either to be printed or delivered for use at the workstation. Similar processing examples exist in design and simulation processing, where intermediate results may be delivered to a workstation application for user refinement or selection of options, and returned to MVS for final processing.

Call back. The key difference in this approach from the two prior cases, is that the session and conversation are not retained once initial exchanges are accomplished. This approach is useful when the service requested at the host may take some time, e.g., several minutes. In this case, the application would be designed to extract specific origin information (using APPC/MVS services) in order to "call back" another participating program at the user's location when results are available.

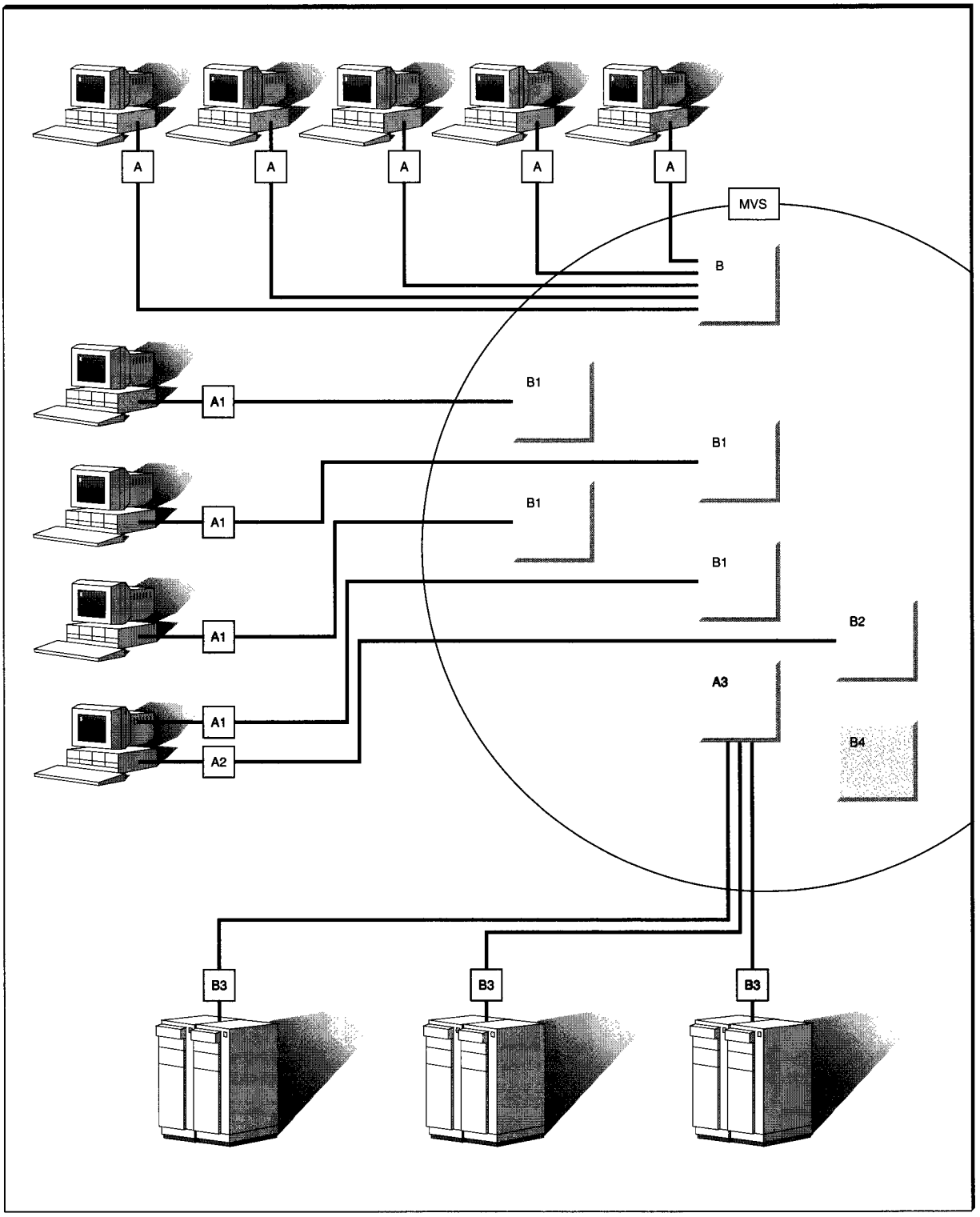
APPC/MVS supports the invocation of a sequence of batch job steps. The first job step could provide needed data exchange for setup. Subsequently, processing would continue where final results would be returned by a later job step in the process. Examples include: a complex query with report generation, sorting a large set of records, collecting information through connections and interchange with other systems, elaborate knowledge-based analysis, and performing numerically-intensive analysis of seismic or medical scan data. Saving results in a user data file can be planned to handle situations where reconnection to the requestor cannot be immediately made.

Call back can also be an effective approach for extending and complementing a CICS or IMS transaction environment. Long-running or compute-intensive work could be requested to run in APPC/MVS from CICS or IMS transactions with results directed to one or more locations.

Continuous data exchange. This approach is used where two or more locations need to retain a connection for the continuous exchange of data. Examples of this peer-to-peer usage across multiple locations is discussed in the section on MVS-to-MVS connectivity.

Scheduling alternatives. The scheduler supplied in APPC/MVS has also been designed to service many concurrent requests for applications. APPC/MVS handles requests for establishing conversations with named programs in a variety of ways. Nor-

Figure 5 APPC MVS scheduling



mally, most applications will be scheduled for execution in their own APPC/MVS address spaces. An address space can be thought of as a private execution environment for an MVS application.

Figure 5 illustrates the different scheduling methods available using the APPC/MVS and an approach where user installations can code their own schedulers if needed. While the illustration portrays workstations, the approaches depicted apply to any client system.

The scenarios depicted in Figure 5 are supported by APPC/MVS and parallel the uses described previously.

Multiple instances. As illustrated, the APPC/MVS scheduler may schedule multiple requests for the same named server on behalf of either the same or different workstations' paired *A1–B1* cases. These requests are satisfied by scheduling copies for execution in independent MVS address spaces. Each will be a fresh copy of the application, hence the programmer need only code the application to handle a single conversation. Another design approach allows applications to stay resident. This provides some gain in performance; however, the programmer must code to reset internal controls to handle subsequent invocations.

For installations using Resource Access Control Facility, each execution address space is personalized or conditioned with inherited authority established for that user. That is, access to programs, files, and other resources will only occur within authorized bounds. Security interfaces are externalized for other comparable security products. By scheduling each request to its own address space, isolation and insulation between copies of the requested server programs is guaranteed. For example, in the event of an application logic store or move data error, the scope of damage will be confined to the address space of execution allowing other applications to continue with normal execution.

Multiple applications. A workstation running Operating System/2 with multitasking capabilities can establish multiple concurrent conversations with different programs controlled by APPC/MVS. This capability is illustrated by the workstation application pairs: *A1–B1* and *A2–B2*.

MVS client. In this case, an MVS batch program *A3* acts as a client program making requests of remote systems *B3*. This is useful for distribution or collection applications, where the host either has a master copy of information to be broadcast (e.g., a new price or inventory status file) or is responsible for collecting remote activity such as the day's activity from remote locations. Further, an MVS batch program could be used to assist in remote workstation updates for new client applications.

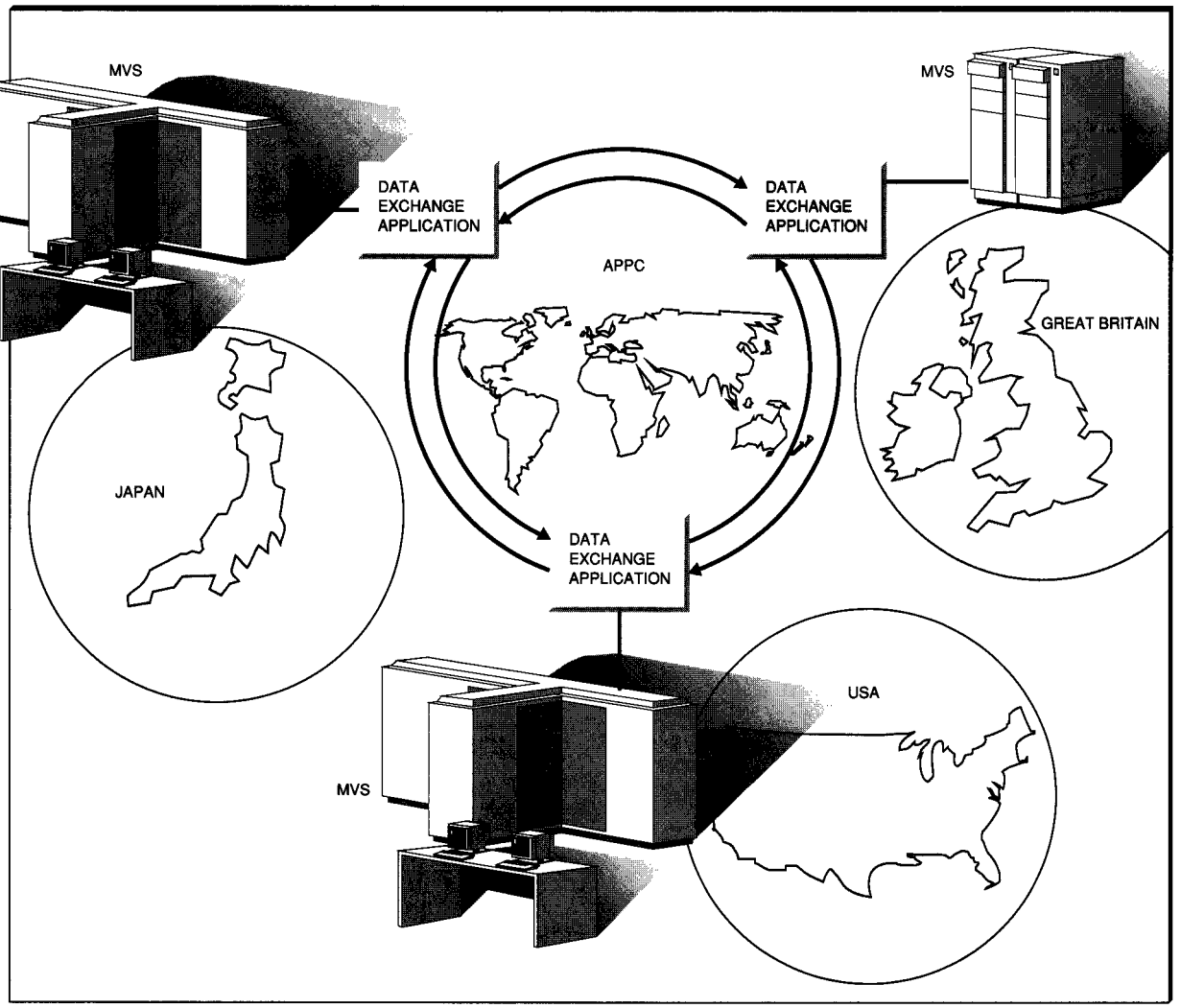
Call back. This is illustrated by the instance of *B4* that has deallocated a conversation, released the session, and is currently executing with the intention to return processing results when completed to a remote client application.

Multiclient or single server. The upper section of Figure 5 shows one server application *B* performing the scheduling and execution of incoming connections from five client workstations *A*. This type of server application manages multiple inbound conversations by either serializing the requests in an internal queue, scheduling the conversations on a task basis, or scheduling them to subordinate address spaces. In this case, the product builder codes a scheduler, using the defined scheduler services of APPC/MVS. Applications running under the control of such a scheduler may use any of the LU 6.2 or CPI-CI communication services.

MVS-to-MVS connectivity. Figure 6 illustrates an application designed to address a business need for immediate or continued communication between geographically dispersed processing centers. While the picture shows three MVS systems, the concept also applies to situations where an APPC/MVS system could interoperate with others supporting APPC such as Application System/400s and System/390s running with IBM's Customer Information Control System/Virtual System Extended (CICS/VSE*) or the virtual machine operating system.

The connection duration in each case is subject to application requirements. A single connection or conversation may be started and the partners perform SEND and RECEIVE data exchanges throughout the day. Thus for example, customer accounting, production control, inventory, or financial applications information processing at a "home" location can be brought to current status from summary data exchanged with other geographic areas. In other cases, the connection could be established over periodic time intervals, where,

Figure 6 MVS-to-MVS connectivity example



for example, a connection is made every hour and applications communicate for several minutes.

Some application scenarios using this approach are:

- Each of the three centers could support an identical set of application servers that handle a particular geographic area. At the same time, each system may need to perform database updates for an application specialty such as order processing or customer accounts. Daily activity occurring at each of the locations needs to be transmitted for each partner to do its specialized processing and database update, e.g., collect district orders for composite manufacturing control.
- There may be a legal or business need to periodically synchronize activity and status. For example, banks are required to meet minimum liquidity requirements in ratio to loans outstanding. Periodic synchronization can enable closer management of any business resource or change in business operating conditions.
- One location may serve as a backup disaster recovery location for another. In this case, log journals of on-line activity could be continuously transmitted to provide a backup remote

copy. It should be noted that there are several IBM products that perform this function. The purpose here is to point out that APPC can be used in cases where additional application material is needed to complement existing approaches.

- A wide variety of options are possible for performing work balancing. Data files or portions of data files can be transmitted to locations needing direct access to the data. Instead of moving data the application can be moved for execution elsewhere. An APPC application client can send the application and job control to another site for submission and processing. Job results may then be packaged and returned.
- Sets of remote unattended processors might be managed from a central site, thus reducing the need for local personnel. In this case, status information would be continuously monitored for handling exceptions. This APPC/MVS application would be a surrogate automated remote operator.
- Corporate applications can be developed and tested in one location. When ready for production, an "install application" can update remote satellite location application libraries for the new version to be used by the next production cycle.
- Corporate reports may be needed on a periodic basis; e.g., information can be gathered for processing at the headquarters from current status at dispersed locations.
- Corporate on-line instruction manuals for business processing procedures can be produced by a variety of departments and installed at remote locations for immediate reference and use.

Several recent products enhance System/390* connectivity configuration possibilities through the use of fiber optic serial channels and switches. Also, APPC Virtual Telecommunications Access Method channel-to-channel connections can be made for either a campus situation or cross country using interconnect controller products. These offer very high-speed links for moving data between systems.

Structure overview. APPC/MVS consists of two key components: APPC services and a scheduler. An APPC request is first received by the Virtual Telecommunications Access Method (VTAM) and passed to the application that manages conversations for the LU name. This is APPC communication services or APPC services. APPC services validates incoming requests and directs the request to the appropriate scheduler for the LU name. It is possible but not necessary to have multiple schedulers present. The scheduler manages the queuing and dispatching of arriving work.

Name scopes. Through interactive dialog facilities, customer administrative personnel define the initial configuration of the APPC/MVS environment. Subsequent changes to the environment as needed are also performed using dialog facilities without the need for shutdown. An APPC/MVS environment definition includes a name scope for defining a set of applications. This name scope is the LU and is defined by an LU name. Some key elements of an LU name are:

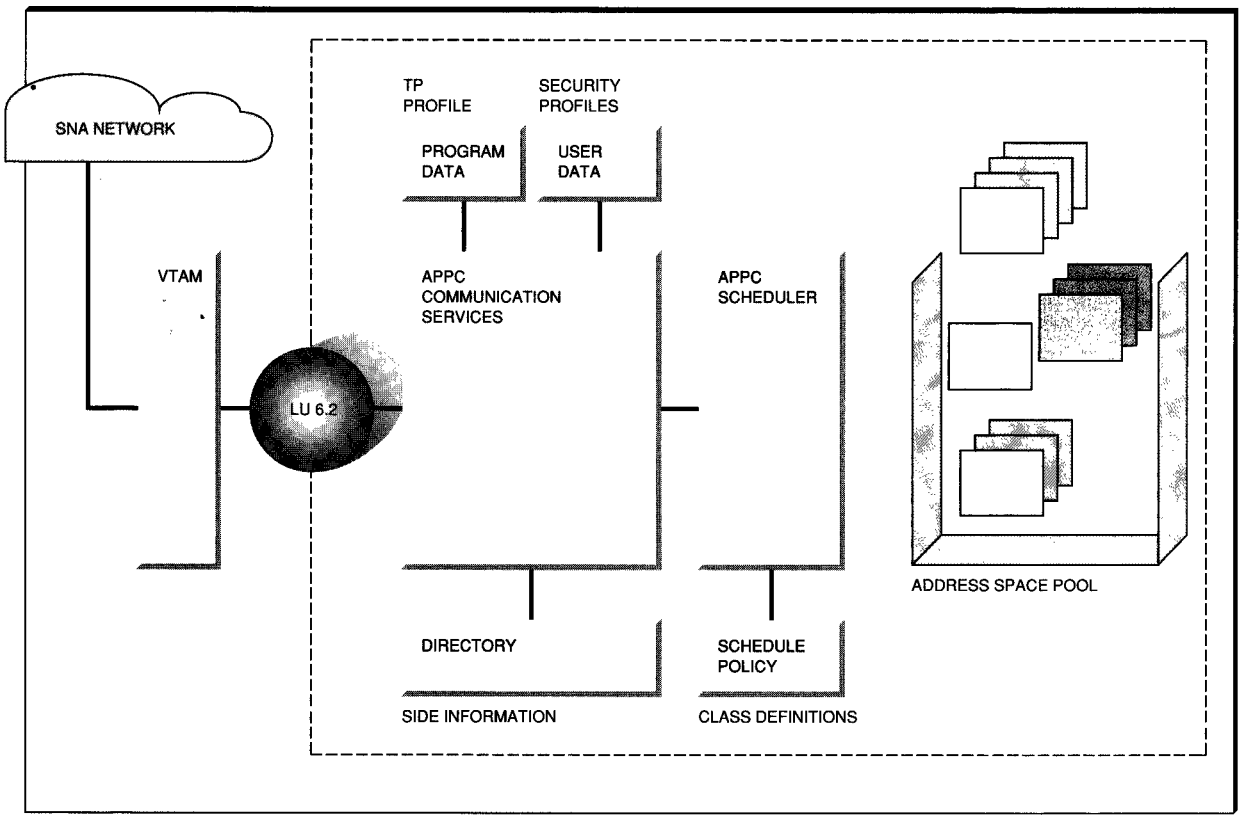
- The applications belonging to the LU
- The resources needed by each application, e.g., file names
- Scheduling rules for each application in the LU
- A default scheduler or specific scheduler name
- A set of work classes used to control scheduling policies
- The directory for supporting outbound communications

The LU is defined by an *add* dialog process that specifies the LU name and names other resources to use within its domain or subenvironment. Multiple LU names or subenvironments are useful to separate work belonging to different projects, or for separating a production and test environment. Facilities are provided to define the overall system resources to be allocated to each name scope; thus, the relative importance of each environment can be specified. Within each subenvironment, the importance of particular applications with respect to prioritization and responsiveness can also be controlled.

Initialization. During APPC/MVS initialization, APPC services dynamically creates structures needed by VTAM. Work classes are also established as well as a pool of address spaces reserved for incoming APPC request scheduling. Work class definitions spell out resources to be applied to arriving work assigned to the named class. This initial environment is depicted in Figure 7. When initialization is complete, VTAM will flow incoming APPC work to APPC services.

The default scheduler delivered with APPC/MVS heuristically manages arriving work based upon

Figure 7 Pool of available address spaces

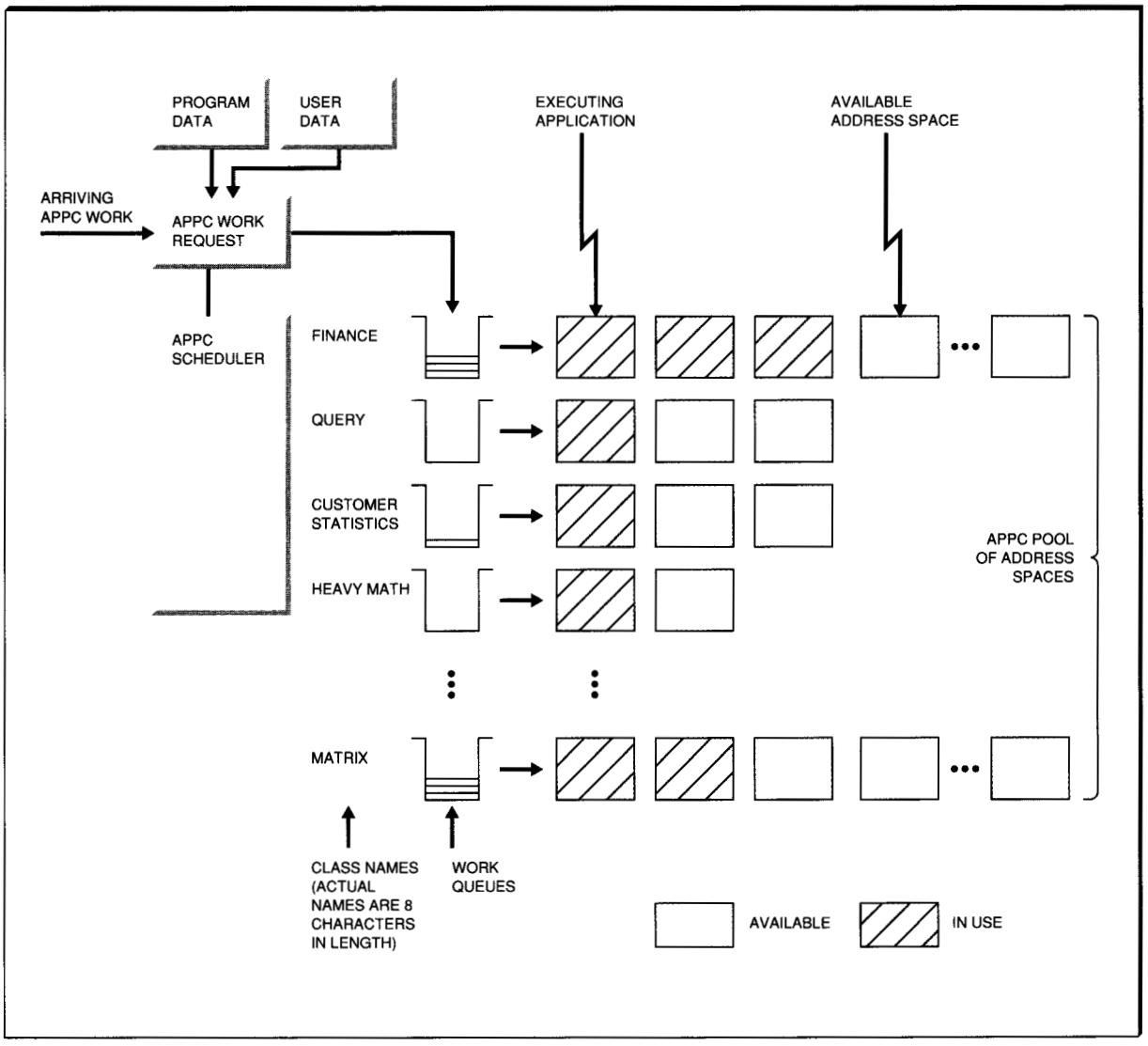


customer-defined policies. APPC applications are represented by a transaction program (TP) profile that defines the resources needed and maps the application to a work class. Each class is defined with controls for a maximum and minimum number of address spaces that are allocated from an address space pool. The numbers of address spaces expand and contract within each class dynamically based upon workload. Applications invoked by the APPC/MVS scheduler are defined in MVS as a new work class for accounting and control purposes. This class is in addition to the traditional types of work currently supported by MVS, namely: transaction, interactive, and batch work classes.

Facilities are available to retain one or more copies of an application in its address space so that initial setup time need not be repeated on subsequent invocations. This performance optimization is of value for servers that will be frequently used or where setup is fairly elaborate, e.g., the allocation and opening of a large number of files.

Inbound processing. Inbound and outbound are used to describe APPC requests from the perspective of the MVS application. An *inbound*, or arriving, APPC request contains the LU name, the application name, and the user identification of the person at the workstation, or user of a peer mainframe application. A security check is performed that verifies that the user may invoke the requested APPC/MVS application. A work request containing user-related, project-related, and application-related information (e.g., which file names to use) is assembled from a predefined TP profile. See Figure 8. The work request is queued to an assigned work class determined from the profile for the application. Available address spaces or initiators dequeue work and start the application. Work request content is also used to set up and initialize the application address space.

Figure 8 Scheduling work



For example, security controls and accounting controls for the application are part of the work request. This material originates from predefined security authority given to user and accounting information in the application profile. Systems resources used by the work are accounted for using standard MVS facilities. Once the application starts it may need to issue other APPC requests to other partners. These are called outbound requests.

Outbound processing. When a host application initiates the APPC ALLOCATE_CONVERSATION request it is called an *outbound* request. These can be made by existing MVS units of work such as TSO, started tasks, batch jobs, IMS transactions, and APPC/MVS controlled programs that were previously scheduled by inbound requests. For example, a batch job would use output requests to distribute or collect information to and from remote sites, such as inventory, orders, shipments

Table 1 APPC/MVS internal connections

Origin	Receiver of Initial Connection				
	APPC	TSO/E	Batch	CICS	IMS
APPC	Yes	Yes(1)	No	Yes(2)	Yes
TSO/E	Yes	Yes(1)	No	Yes(2)	Yes
Batch	Yes	Yes(1)	No	Yes(2)	Yes
CICS	Yes(2)	Yes(1)	No	Yes	Yes(2)
IMS	Yes	Yes(1)	No	Yes(2)	Yes

Note: (1) APPC-provided TSO Test Service. (2) Conversation must flow through VTAM.

en route, or price status. A system directory or *side information file* (see Figure 7) is used to translate symbolic destination names used in programs to information needed by VTAM to connect to remote partners.

When a request for an outbound conversation is received by APPC/MVS, it is processed locally if the requested partner is located on the same machine and the LU is managed by APPC/MVS. Thus it is possible for APPC to be used as a limited internal interprocess communication mechanism.

Internal cross-talk. When the target application is to be scheduled for execution in the same MVS system by APPC/MVS, optimized cross-memory techniques are used to transparently flow LU 6.2 formats and protocols between the executing partners. APPC/MVS supports internal communication between TSO/E units of work (such as programs and commands), batch and started tasks, IMS transactions, and APPC/MVS scheduled applications executing in the same MVS system.

Table 1 portrays the capability of MVS applications that can issue an ALLOCATE_CONVERSATION request to a partner residing in the same machine. Once connection is established, two-way communication between the partners can flow.

TSO/E. TSO/E users can invoke commands, programs, and procedures that utilize the communications programming verbs supplied by APPC/MVS and establish conversations with other programs. Note that access to the TSO/E environment is still controlled by Systems Network Architecture LU 2 protocols that assume user access from the IBM 3270 family of terminals or 3270 data stream protocols; once logged on to TSO/E the user can invoke applications that then utilize APPC/MVS communication services.

This capability enables TSO/E applications access to CICS- and IMS-managed data via internal conversations to transactions that can extract and send the data to the TSO address space for processing. Thus customers can acquire value from APPC/MVS using existing 3270-type terminals.

Batch. Batch programs can also have conversations with target programs in the network or in the same MVS image, for example CICS or IMS transactions, or programs scheduled by APPC/MVS.

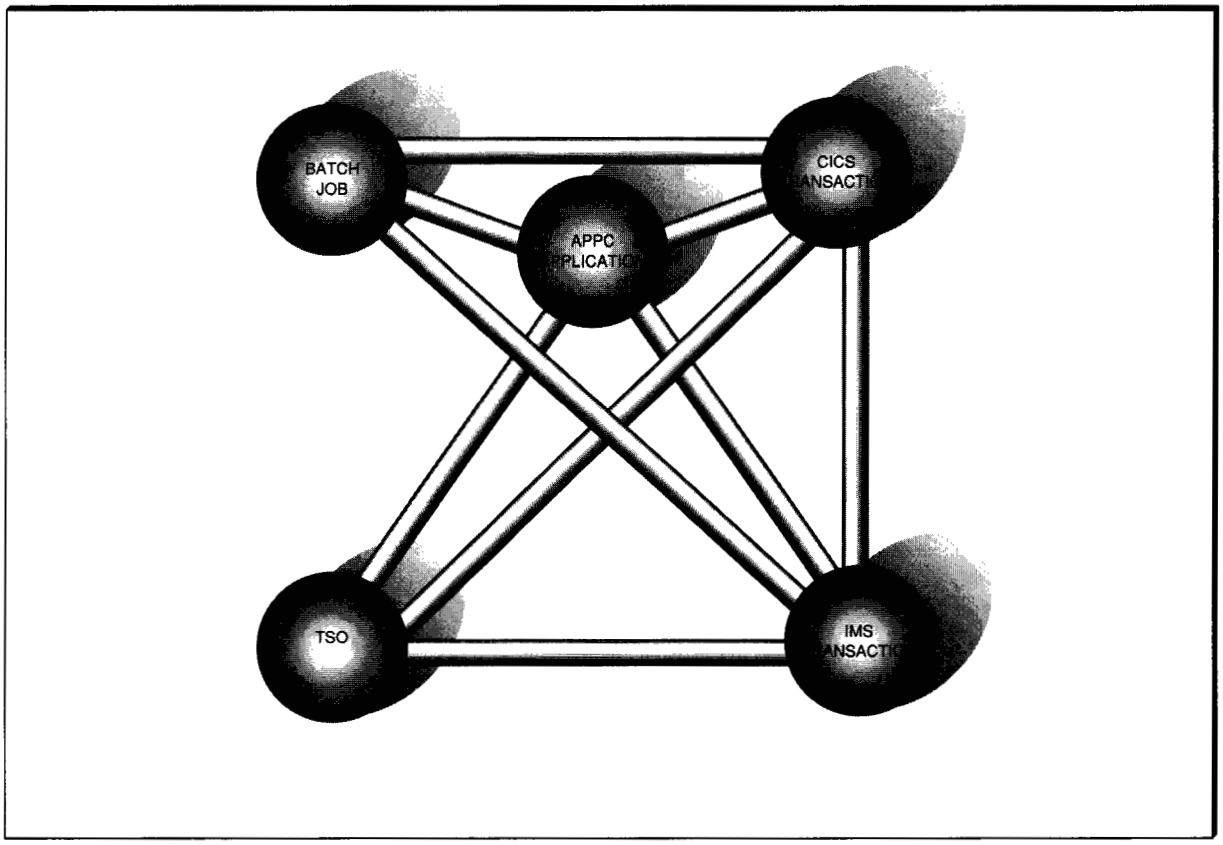
IMS. Early in the design of APPC/MVS it was decided to split out the scheduler and establish a generalized interface for use by products that could benefit from APPC communications support. These services are restricted to authorized applications subject to installation controls. This *transaction scheduler interface* can be used by an installation wishing to develop scheduling functions other than the ones provided in IBM products, or by application builders willing to provide APPC scheduling functions in their own products. In addition, this interface is utilized by IMS/ESA Version 3.2 to allow IMS transaction programs (TPs) to establish conversations with other APPC-capable environments. From the IMS application point of view, these communication capabilities offer several advantages:

- Existing IMS TPs can be invoked from APPC partner programs. This implicit APPC support can be used by applications that have to communicate with APPC- and non-APPC-capable systems, or by applications that do not want to be sensitive to the APPC protocol.

  In this environment, the IMS TP is completely isolated from the conversational model of communication defined in the LU 6.2 architecture. The IMS control region interacts with the APPC partner program to send and receive data over the conversation managed by APPC/MVS. Using the IMS message queue, the control region buffers the data exchanged over the APPC conversation. That data can be retrieved or inserted by the IMS transaction program, using the current IMS API. Guaranteed message delivery is provided for applications using this mode of operation.

- New IMS applications can be coded using the CPI-CI interface. This is a totally different style of programming from the current IMS queued

Figure 9 Internal APPC connections



API. The program accepts the conversation and communicates directly with its partner for data interchange, without the IMS message queue manager being involved in the process. IMS provides synchronization facilities for local resources used by these applications by handling Resource Recovery Interface calls defined in IBM's Common Programming Interface and uses them to drive the IMS sync point manager.

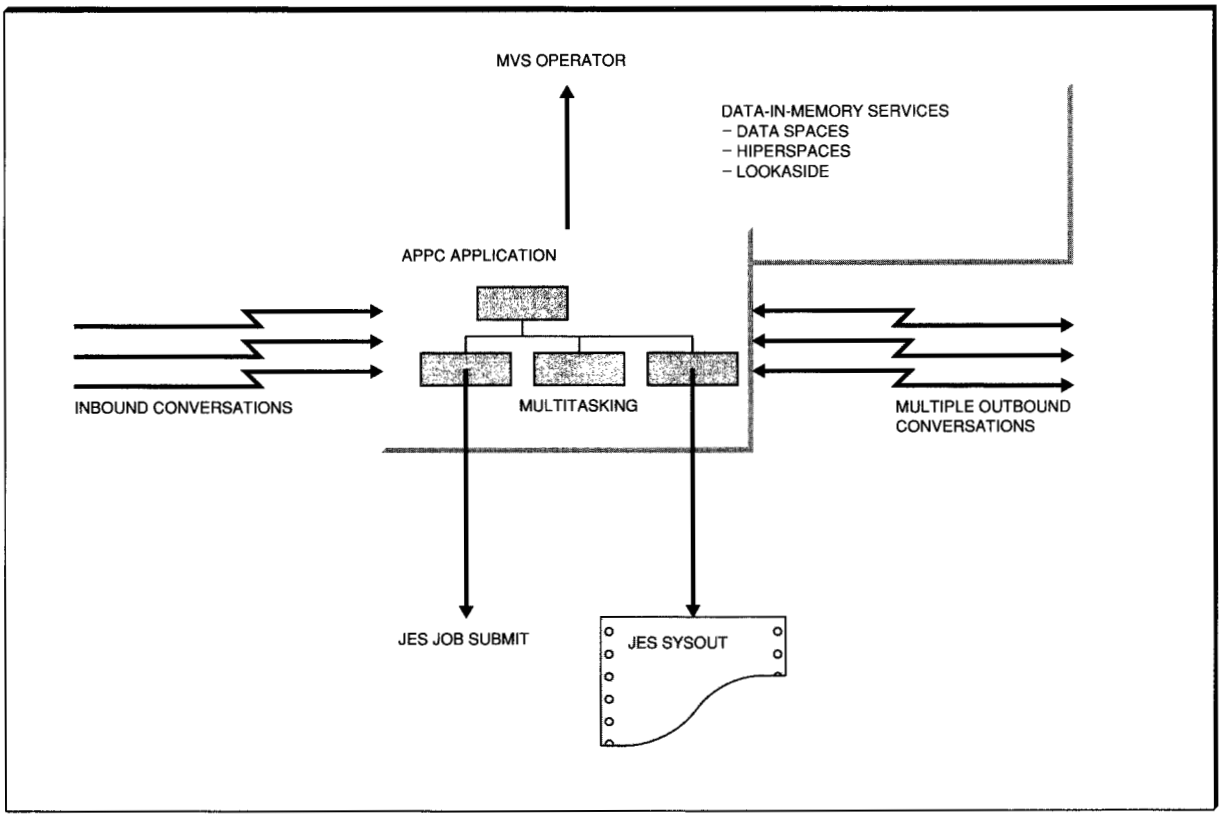
CICS. CICS provides a functionally rich programming interface that is implemented across its product family. Native CICS services include APPC communication capabilities that allow CICS transactions to exchange data between CICS or non-CICS partners. In the MVS environment, CICS is defined as a separate LU and uses performance-oriented VTAM interfaces. Conversations between the APPC/MVS-supported environments and CICS transaction programs using CICS APPC services must flow through VTAM.

Within the restrictions in Table 1, APPC can be used as a generalized method for communication between two MVS applications running in different address spaces. In this case, the partner must be started by APPC/MVS. A needed facility to overcome this restriction is to allow any MVS application to register with APPC/MVS services as a server for APPC inbound traffic.

Figure 9 summarizes pictorially the internal connections possible.

Through re-engineering of existing applications and the addition of APPC capability, new combinations of application integration across diverse business environments are possible. Installations can benefit from improved program-to-program communication capabilities, integrate current LU 2 and LU 6.2 environments, and help provide a single system view for their end users. For example, a decision support application executing in the

Figure 10 MVS system services



TSO/E environment can establish communication paths with CICS, IMS, and APPC/MVS partner programs for data retrieval. The installation now has expanded application options to choose from, depending on the application complexity and frequency of invocation. Analogous scenarios also apply to traditional CICS and IMS 3270-type transactions that can trigger the execution of long-running work in the APPC/MVS environment.

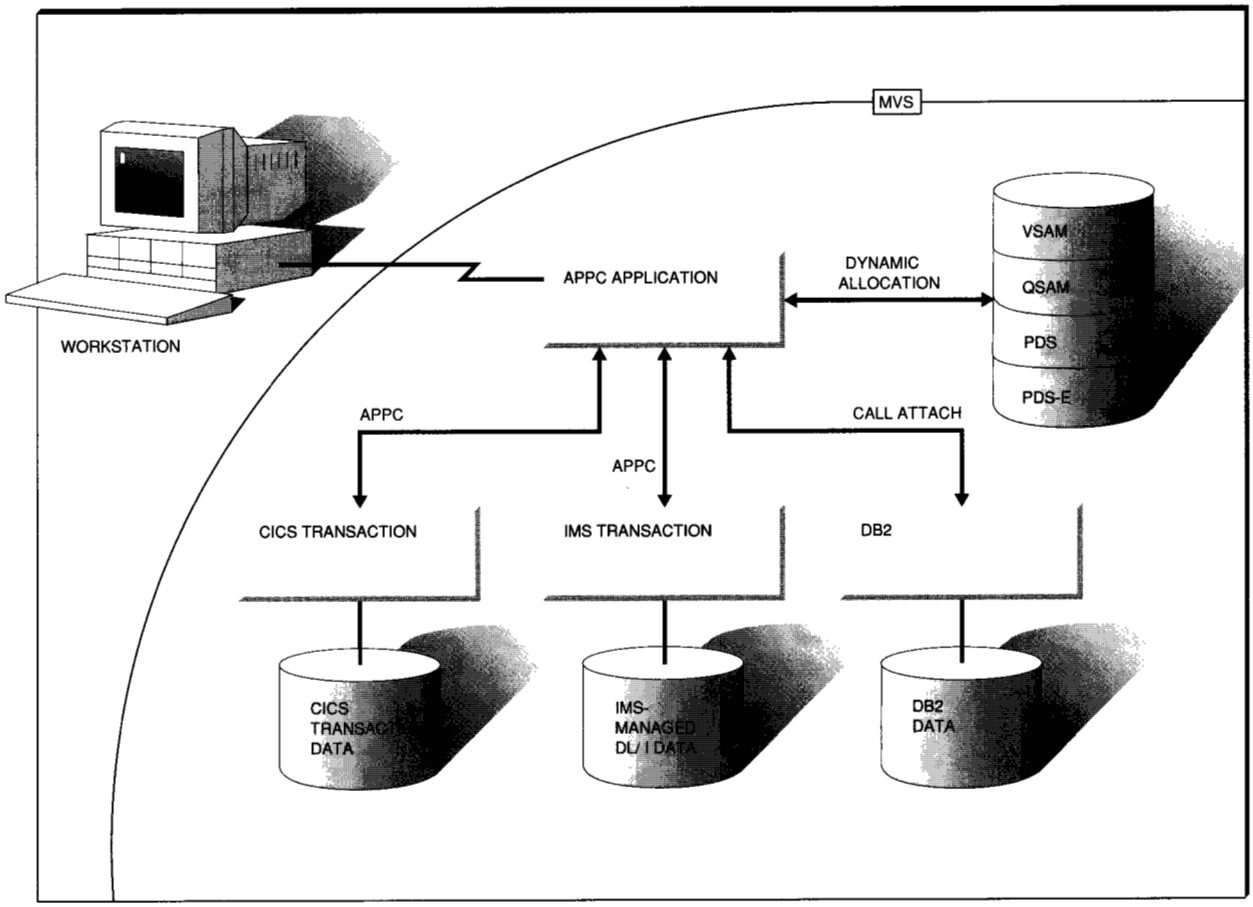
System services. As illustrated in Figure 10, during execution an APPC/MVS program executes in a full-function MVS address space and has access to an extensive and powerful set of MVS services.

An APPC/MVS program can communicate with one or more partner programs through APPC conversations. An APPC/MVS program can use a selected set of TSO/E services, called the TSO/E *environment services*, that provide many services available on the interactive environment of MVS, for example, string and command parsing.

APPC/MVS programs may allocate data buffers for communication in data spaces.[20] MVS LU 6.2 programming services provide options for locate mode on send and receive calls to operate on data that reside in data spaces. An asynchronous form of LU 6.2 calls is also available to build multiclient applications that need to handle multiple conversations concurrently. MVS programs can create and process data in look-aside spaces mapped into real or expanded storage. For certain applications, data and tables in shared memory offer large performance savings from I/O avoidance.

Other MVS service examples include operator communication, program management, security, accounting, and performance services.

Figure 11 APPC data access



Finally, an APPC/MVS program can use any Job Entry Subsystem (JES) service during its execution. Program-generated output (*SYSOUT*) can be scheduled for local or remote printing. JES job submission services can also be used. Submitted jobs can, in turn, use all APPC/MVS facilities for communication.

APPC/MVS programs—data access. Figure 11 presents some of the ways APPC/MVS applications can access existing or new customer data investments. Direct access to MVS-controlled data is available and the corresponding data sets can be allocated through MVS methods, via allocation performed before program execution or dynamic allocation performed during program execution. An APPC/MVS application may create and access private VSAM, Queued Sequential Access Method, Partitioned Data Set, or Partitioned Data Set-Extended data sets using MVS access methods. If the application needs to share and update MVS data sets across concurrent copies of the application, it may be necessary to implement private protocols to synchronize updates. An approach is to assure that only one instance of the application can own and update the data. It is possible to limit such a data-owning application to a single execution instance with APPC/MVS class controls. Related applications needing to perform updates could make update requests that would be serialized. Such designs are possible using MVS cross-memory facilities or internal APPC conversations.

Direct access to data managed by DB2 can be achieved through the DB2 call attach facility and the DB2 TSO attachment facility. With DB2 Version 2 Release 3, APPC/MVS applications using PL/I,

FORTRAN, COBOL, and C can use the new high-level language services for the call attachment facility; for releases prior to DB2 Version 2 Release 3, an Assembler H program is required between the call attach facility and the APPC/MVS application. This access enables MVS client and server applications to be coded with embedded Structured Query Language statements, where DB2 as a resource owner will manage the sharing and locking of DB2 data.

IMS controlled data can be accessed in several ways: the APPC/MVS program can schedule a conversation with an IMS transaction, which may then perform the needed data extraction or update to IMS managed data; direct access capabilities can be made by APPC/MVS programs that have been coded to the rules for an IMS batch message processing program. In this case, an APPC request may start the batch message processing application, which may access IMS full function databases and fast path databases. This is possible because the tasking structure of an APPC/MVS application parallels the tasking structure for an MVS batch job step.

An APPC/MVS program could be designed to work with a CICS customer-written transaction for data access. This approach provides for CICS data sharing, locking, and update with integrity under CICS management. While APPC/MVS shares similar communications and scheduling capabilities with CICS and IMS, the CICS and IMS subsystems retain their strong history of database management and protection. The recovery and commit facilities available to the APPC/MVS program are provided by each of the subsystems or resource managers that control the data that the program references. Currently, it is the responsibility of the APPC/MVS application to assure that related updates across multiple resource managers are coordinated.

Opportunities may exist to take advantage of customer data investments in new ways:

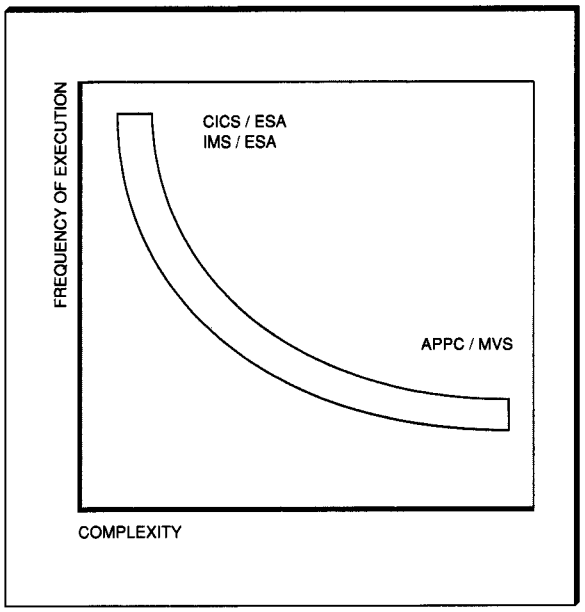
- Applications that need to integrate information from multiple sources, e.g., DB2, IMS Data Language/1, or CICS File Control, or need to establish conversations with other systems to access and collect data are possible. See Figure 4.
- If large amounts of data need to be frequently referenced by cooperative applications, these data can be loaded into a data space and shared across a set of applications.
- APPC applications can monitor the on-line database activity for *data triggers* or business exception conditions. Upon the occurrence of an exception, an alert can be sent to a workstation application on the appropriate executive's desk for immediate and automatic display.

APPC/MVS and subsystem relationships

Evolution of processing styles. The history of MVS is continual evolution and change to meet new customer needs. MVS data processing evolved from the batch-oriented model introduced in the 1960s to two distinct models of computing that became prevalent in the 1970s, namely: on-line transaction processing and interactive processing. These styles became effective as technology enhanced hardware capabilities in several areas such as display terminals, networking products, better direct access storage device performance and density, and increased channel and compute power from high-end systems. At the same time, the use of improved technology lowered costs enabling more on-line applications to be justified. Technology will continue to influence the processing models or paradigms supported by computers. For MVS they have always been evolutionary and additive. Such is the case with the addition of the conversational model and distributed style of processing to the already existing styles of transaction, interactive and batch. For additional discussion of computing paradigms since the 1960s, see Reference 21.

TSO/E relationships. TSO or Time Sharing Option was originally designed to give typewriter-type terminal users access to host services. Each user session was associated with an MVS address space where services of TSO and MVS could be requested. This interactive environment is suitable for simulation and modeling, forecasting, numerically intensive computing, large graphics, decision support systems, and program development tasks. The services are typically unstructured and nonrepetitive, create spontaneous workloads, and are characterized by requiring the manipulation of large amounts of data for extended periods of time, during which a large number of user interactions are allowed for data editing, analysis, and manipulation. Response time service levels are not normally as critical as in transaction processing systems. In MVS this environment has been addressed by TSO/E and an SNA LU 2 con-

Figure 12 Product positioning



nection, which uses a dedicated session and exchange of data using a 3270 data stream. Workstations are capable of emulating 3270 screens and can be used concurrently for TSO work as well as APPC work.

As described earlier, many TSO/E services are available to an APPC/MVS application. A key exception are the TGET/TPUT functions and other services that rely on 3270 data stream support. Additionally, new services have been provided with APPC/MVS to enable interactive testing and debugging of an APPC/MVS application in a TSO/E address space.

For developing multiple parts of a distributed application, a TSO registration service is provided. This service allows a TSO/E user to have an inbound APPC request routed to an application entry point in the user's TSO/E address space. After registration, APPC traffic for the named application and for the registered userid will be routed to the TSO/E space for debugging purposes.

A TSO/E application now has added data access capabilities with APPC/MVS connectivity. An application may need to extract data from either the CICS or IMS environments for analysis and presentation using a 3270 data stream. In this case, an APPC conversation can communicate with an appropriate transaction written to support the application.

Further, a client or server application that uses the Restructured Extended Executor language (REXX) may be developed using TSO/E debugging facilities. Once successfully tested, it may be installed as an application in APPC/MVS.

APPC/MVS, then, provides added capabilities while coexisting and cooperating with the TSO/E interactive environment.

CICS and IMS cooperative processing. The on-line transaction processing environment addresses the automation and improvement of efficiency of many day-to-day business activities. Most are business operations that are generally predictable and repetitive in nature, such as the handling of automated teller machines, order entry, status inquiry, accounts receivable and payable, and customer record management, to name a few. This model allows for services with well-defined logic that is characterized by handling high-speed flows of finite amounts of data from the terminal to the MVS processor and its controlled data. The users share the environment of programs and data and repetitively process similar transactions, expecting fast response times from those services. To accommodate these requirements, on-line transaction processing services require small amounts of processor resources (e.g., storage, cycles) during a small period of time for each transaction invoked.

In MVS this environment is supplied by the CICS and IMS products. These products have been optimized to handle the transaction style requirements with great efficiency. Their key focus is on delivering high-volume transaction rates, with critical response times and low cost per transaction, while also providing key roles for data access and sharing through commit and backout facilities. Over the years, they have been enhanced to support very productive programming services, and a large skilled base of programmers familiar with these programming interfaces provided by these subsystems exists in the application programmer community.

From the traditional MVS on-line transaction processing environment, CICS has extended its of-

fering to the workstation controlled by Operating System/2, allowing cooperative applications to be developed using the CICS specific services and interfaces and the defined communication interface CPI-CI. A CICS partner running in Operating System/2 may now establish an APPC conversation with either a CICS or APPC/MVS partner.

The transaction style of application processing will continue to be a key essential business solution for applications needing the performance attributes delivered by CICS and IMS. APPC/MVS coexists and complements the transaction products as those transactions may invoke and communicate with more resource intensive service applications.

The addition of APPC/MVS to the MVS operating system gives MVS a more complete set of support facilities to address distributed business problems. Two key aspects need to be considered when selecting an environment for a new distributed or cooperative application. What is the expected frequency of use and how complex is the application? For those that do not result in a clear-cut choice, selection of an environment can be based upon available programming skill or the need for environment-unique functions or services.

The amount and type of processing needed by an application can vary across a wide spectrum, from transactions that are handled with subsecond response time to batch processing, which can take several hours to perform a job. Figure 12 shows the relationship of transaction-style processing characteristics with cooperative characteristics, where the amount of work required by an application (*complexity*) directly affects the expected frequency or numbers of requests per second, and application responsiveness (*frequency of execution*). Clearly, as more resources are needed for a task to be accomplished, responsiveness must also diminish. To illustrate these relationships, consider the following increasingly complex applications.

To start, a simple application accomplishes a single function such as a customer name and address lookup, handling a debit or credit from an automated teller machine, or doing a credit card authorization. Both CICS and IMS are capable of handling thousands of these types of transactions per second. To continue, other applications may require a high number of accesses to multiple database records. Such could be the case in determining the status of previously ordered equipment with multiple components. Larger and more complex applications are those which have increased I/O demands, perhaps hundreds of database accesses in addition to needing other system resources, e.g., computation and additional communications. As applications become more complex, the expected volumes or frequency will diminish, assuming computing resources are held constant.

•

The overlap between the transaction and cooperative facilities is real. For some applications a case can be made for either environment. It is perfectly valid for a customer to select a transaction style approach for a new application if there is considerable need for on-line database update or the need for other transaction environment features not available in APPC/MVS, or to take advantage of available programmer skills. While customers may address more complex applications with a special-purpose application that runs in its own application-owning region in CICS or an IMS transaction that runs with a lower priority, APPC/MVS offers a viable alternative. The relative importance of transaction work versus distributed work can be governed by customer controls when multiple environments share the same MVS system. Figure 12 suggests when each of the CICS, IMS, or APPC/MVS products should be used.

Application examples. Some of the application characteristics that are suited for APPC/MVS are similar to those of the interactive computing model. In contrast with the well-defined logic typical of on-line transaction processing environments, these applications are more ad hoc in nature. This is typical of simulation, modeling, forecasting, and decision support systems, and knowledge-based, planning, and control applications.

These applications require large amounts of resources during their execution. Resources refer to either data or processing requirements. Typical examples of good APPC/MVS applications are numerically intensive computing, statistical analysis, linear programming, simulations, very large queries, sorts, applications that need continuous data exchange between locations or need multiple concurrent connections for broadcast of changes or collection of activity, and, in general, any type

of batch activity that the installation wishes to initiate from the workstation.

Other characteristics relate more to the way APPC/MVS implements the environment for application execution. APPC/MVS applications run in full function address spaces and are allowed to use the full set of MVS services. Application requirements that can be addressed in APPC/MVS are:

- The ability to create multiple tasks during application execution. Examples are collector and distributor applications that require multiple concurrent conversations to request or distribute data from or to the network. Data collection could be done to exploit the MVS capabilities for workstation data backup and archive; data distribution could be used for workstation data and program maintenance.
- Handling large amounts of data and loading it in real or expanded storage for I/O avoidance. Many library, archive, and business directory applications can fully take advantage of these MVS services for data space and hiperspace[20] usage. Hiperspace commonly means high-performance storage and can be used as a very efficient cache for highly referenced data.
- Large processor or data requirements are better handled in a personalized environment implemented in an MVS address space. The inherent isolation and reliable characteristics of this construct could be used to guarantee no interference with other environments.

The following sample list of applications, some repeated from Reference 22, is intended only to give the reader a sample of concrete examples in which the above-mentioned characteristics apply.

Systems management. Companies have a need to service local area network servers with software updates and new application packages. Installing distributed applications also calls for updates to other controls such as system directories that tie physical locations with names used within programs. These servers can be driven from a central site to update remote libraries in a concurrent fashion. Further, performance monitoring of an enterprise network is possible through alert management. APPC/MVS applications can be written to detect, analyze, correct, and track incidents across all nodes of an enterprise network.

Communication gateways. Application servers connected to different networks (e.g., SNA and OSI) can be written to translate one protocol to another, providing a data interchange service between the two networks.

Loan processing. A loan officer gathers information about the required loan amount and customer profile data and sends it to MVS. The MVS application sends the data to a mortgage insurer system, while at the same time knowledge-based processing is applied to evaluate the risks of granting the requested loan. Final results are then relayed back to the loan officer. In between the submission of the request, the connection need not be continuously maintained. The host application can be designed to "call back" the client.

Complex circuit design and test. Initial chip or component design is performed at an engineering workstation Once preliminary design testing is accomplished locally, the design is submitted to an MVS system for a more complete system design stress test and analysis. Results are returned for design refinements as needed.

Aircraft gate scheduling. Initial scheduling data could be downloaded to a local-area-based server. Users from the workstations could use IBM's Presentation Manager* interface to present a graphical view of the tarmac with the current aircraft-to-gate mapping. The mouse could be used to get additional flight details when pointing at a certain occupied gate. The whole picture could be updated on a real-time basis as new flights arrive or depart. Gate allocation for flights coming early or late could be handled through knowledge-based processing on the host, and the data downloaded through APPC facilities to the local area network server for real-time display update.

Decision support system. Today, customer purchases and product prices are gathered at point-of-sale systems installed in supermarkets. These valuable data can be batched periodically for APPC transmission to the producer. Decision support systems could then extract the relevant data for each product, and deliver an accurate current picture of sales status by geographic regions. This allows decisions to be made as to where sales promotions are needed, how well and why competing products are selling, and other important decision data. The competitive advantage of such data could even be exploited to feed information back to the supermarkets and influence their deci-

sions on shelf allocation space, based on the trend for each product's sales.

Configuration systems. A manufacturer of complex electronic equipment could use the cooperative model for processing orders for its products. The orders could be accepted at the workstation, and a first screening of the basic characteristics and prices of the requested product done at this level. Data could then be shipped to MVS for full configuration and pricing according to the requested parameters. This step could require large amounts of data, depending on the complexity of the product. A sample list of possible options could then be sent to the workstation for final customer decision.

Vendor applications. Several software vendors have stated their intent[23] to provide applications that use APPC/MVS as their base. The solutions provided by those applications focus on business areas such as financial applications, cooperative mathematical analysis, cooperative three-dimensional graphics on the IBM RISC System/6000*, DOS support for APPC, cooperative insurance applications, cross-system monitoring, accounting applications, software distribution, file transfer, cooperative application generators, version and configuration management, human resource management, and user-definable computer-aided software engineering (CASE) applications.

MVS and its companion subsystems have evolved through the years to address new needs. This evolution continues with the availability of APPC/MVS. APPC/MVS is a good beginning; however, there are a broad set of challenges that remain to address distributed needs. The following list is not intended to be exhaustive, but to recognize the need for further development.

- Tools for development—While considerable progress is underway with new CASE tools, there is a need for developing all parts of a distributed application concurrently, and having those parts system-verified before installation.
- Systems management—Tools that facilitate and carry out the installation and management of client and server partners, along with appropriate directory updates as a single operation remain to be developed.
- Accounting—Tools that enable the collecting and assembly of activity across a network are needed. To assist this effort, there is the need to synchronize time across global networks as well as to uniquely identify and associate work elements.
- Full duplex—This is a performance optimization for using APPC protocols that remains as a future goal.
- Interoperability[24]—APPC/MVS has begun to open up access to MVS resources. Further requirements have been collected to more completely enable interoperability with other networking protocols such as TCP/IP and OSI, and to more closely work with Advanced Interactive Executive* (AIX*) for data sharing, user interfaces, and other communications paradigms such as remote procedure calls and messaging and queuing approaches.

  The challenge is to obtain agreement on an approach and standard that enables all applications to cross-talk across all networks.

Conclusion

We've explored the facilities and structures of APPC/MVS and how the these facilities can be applied to new application approaches in a variety of ways. We've further seen how new partnerships are made possible, both internal to MVS, that is, between customer investments in the transaction subsystem products and the shared database access support they provide, as well as between MVS and workstations.

APPC/MVS is one step in the continuing evolution of MVS to address continuing needs for new capabilities. The integration of APPC/MVS services provides an environment for new host applications that can address many outstanding business needs in the world of distributed enterprises. Along with its subsystems and applications, MVS delivers reliable support for the management, security, and integrity of corporate information assets. Continued enhancements and support in MVS of other communications models such as remote procedure calls, as well as message and queuing support across all types of networks (e.g., TCP/IP, OSI, and SNA), will assure that MVS applications continue to play an important role in future enterprise networks by delivery of information to end users.

Acknowledgments

This paper is dedicated to the builders of APPC/MVS and its extensions. Their hard work has provided the infrastructure and services for MVS to better meet the needs of open and distributed processing. I also wish to recognize and thank the many IBMers who took time to critique early copies of this paper. In particular I'd like to mention: Edward Cobb, Paul Gorgen, Mike Kistler, Arthur Neil, Manuel Patrone, and Robert Pilz, for their assistance and constructive input.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Prodigy Services Company, UNIX Systems Laboratories, Inc., Institute of Electrical and Electronics Engineers, or Microsoft Corporation.

Cited references and notes

1. A. G. Fraser, "Designing a Public Data Network," *IEEE Communications* **30**, No. 10, 31–35 (October 1991).
2. J. Markoff, "A System to Speed Computer Data," *The New York Times* (January 23, 1991). This article describes image compression advances by the Joint Photographic Expert Group, or JPEG, and the Motion Picture Expert Group, or MPEG.
3. N. P. Negroponte, "Products and Services for Computer Networks," *Scientific American* **265**, No. 3, 76–83 (September 1991).
4. "Grand Challenges: High Performance Computing and Communications," The FY 1992 U.S. Research and Development Program. A copy may be obtained by request to: Committee on Physical, Mathematical, and Engineering Sciences, c/o National Science Foundation Computer and Information Science and Engineering, 1800 G Street, N. W., Washington, D.C. 20550.
5. *MVS/ESA Applications Development: Writing Transaction Programs for APPC/MVS*, GC28-1121, IBM Corporation (January 1992); available through IBM branch offices.
6. *SAA Common Communication Support Summary*, GC31-6810, IBM Corporation (December 1991); available through IBM branch offices.
7. *SAA CPI Communication Reference*, SC26-4399, IBM Corporation (October 1991); available through IBM branch offices.
8. *Transaction Processing: Concepts and Products*, GC33-0754, IBM Corporation (December 1990); available through IBM branch offices.
9. The *IBM Systems Journal* **28**, No. 1 (1989), focuses on the role of large systems and the evolution of the architectural enhancements used to meet the current and future demands.
10. B. R. Aken, Jr., "Large Systems and Enterprise Systems Architecture," *IBM Systems Journal* **28**, No. 1, 4–14 (1989).
11. *MVS/ESA General Information Manual for MVS/ESA System Product Version 4*, GC28-1600, IBM Corporation (December 1991); available through IBM branch offices.
12. J. P. Gray, P. J. Hansen, P. Homan, M. A. Lerner, and M. Pozefsky, "Advanced Program-to-Program Communication in SNA," *IBM Systems Journal* **22**, No. 4, 298–318 (1983).
13. *APPC Product Implementations*, GG24-3520, IBM Corporation; available through IBM branch offices.
14. *SNA Transaction Programmer's Reference Manual for LU 6.2*, GC30-3084, IBM Corporation; available through IBM branch offices.
15. *SNA Format and Protocol Reference Manual: Architecture for LU Type 6.2*, SC30-3269, IBM Corporation (December 1985); available through IBM branch offices.
16. *SAA Writing Applications: A Design Guide*, SC26-4362, IBM Corporation (August 1988); available through IBM branch offices.
17. *SAA CUA91 Reference*, SC34-4290, IBM Corporation; available through IBM branch offices.
18. R. A. Demers, "Distributed Files for SAA," *IBM Systems Journal* **27**, No. 3, 348–361 (1988).
19. A. L. Scherr, "SAA Distributed Processing," *IBM Systems Journal* **27**, No. 3, 370–383 (1988).
20. K. G. Rubsam, "MVS Data Services," *IBM Systems Journal* **28**, No. 1, 151–164 (1989).
21. L. G. Tesler, "Networked Computing in the 1990s," *Scientific American* **265**, No. 3, 54–61 (September 1991).
22. *MVS/ESA SP Version 4, APPC/MVS Technical Presentation Guide*, GG24-3596, IBM Corporation; available through IBM branch offices.
23. *MVS/ESA SP Version 4, Software Vendor Solutions Reference Guide*, G326-0039, IBM Corporation (September 1990); available through IBM branch offices.
24. *SAA/AIX Interoperability Enterprise Solutions for the 1990's*, G320-9929, IBM Corporation (February 1990); available through IBM branch offices.

General references

U. D. Black, *OSI: A Model for Communication Standards*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1991).

D. E. Comer, *Internetworking with TCP/IP, Principles, Protocols and Architecture*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1988).

R. J. Cypser, *Communications for Cooperating Systems OSI, SNA, and TCP/IP*, Addison-Wesley Publishing Company, Inc., Reading, MA (1991).

Future Networks, New Developments, New Opportunities, R. Reardon, Editor, Blenheim Online Publications, London (1989).

D. Leebaert, *Technology 2001: The Future of Computing and Communications*, M.I.T. Press, Cambridge, MA (1991).

J. Martin, *Telecommunications and the Computer, 3rd Edition*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1990).

MVS/ESA SP Version 4 APPC/MVS Technical Presentation Guide, GG24-3596, IBM Corporation (December 1991); available through IBM branch offices.

MVS/ESA SP Version 4 Technical Presentation Guide, GG24-3594, IBM Corporation (December 1991); available through IBM branch offices.

K. Ziegler, Jr., *Distributed Computing and the Mainframe: Leveraging Your Investiments*, John Wiley & Sons, Inc., New York, NY (1991).

Accepted for publication December 15, 1991.

Fred W. Voss *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602.* Mr. Voss is a senior programmer at the Mid-Hudson Valley Programming Laboratory developing synergy approaches for the System/390 operating system family. Mr. Voss was a member of the early team that designed APPC/MVS and was instrumental in providing early education to IBM and customer personnel. This activity included consultation with many other laboratories in IBM, with specific focus on CICS, IMS, and DB2 products. He received a Division Award in 1991 for APPC/MVS planning and education leadership. Mr. Voss joined IBM in 1956 as an applications programmer and soon moved to systems programming. He assisted in the design and development of IBM's Advanced Administrative System, used by IBM branch offices to manage daily business activity. He was a manager of a standards organization which had focus on internal programming issues across multiple locations. Mr. Voss transferred to Poughkeepsie in 1972 to work on advanced operating system designs and has since held numerous design and architectural positions. His current interests include application services and connectivity across S/390 systems. He received his B.A. in math in 1956 from New York University, and an M.B.A. in statistics in 1966, also from New York University.

Reprint Order No. G321-5479.