SNA Perspective

Volume 13, Number 3
March, 1992
ISSN 0270-7284

The single source, objective monthly newsletter covering IBM's Systems Network Architecture

IBM's Leading Communication APIs Face Off: CPI-C and APPC

Companies are investing a growing portion of their information technology staff time and budgets in creating and maintaining distributed applications. Because the inherent incompatibility of networked applications and their environments is the main cause of schedule and budget overruns, coherent connections and application portability are very important. IBM and other vendors have for years been developing various application program interfaces (APIs) to enhance application portability.

This article is the first of a two-part exploration of IBM's two principal APIs for interprogram communications in multiplatform networks: Advanced Program-to-Program Communications (APPC) and Common Programming Interface-Communications (CPI-C). In this first part, we will focus on technical considerations. We begin by defining APPC as the original LU 6.2 API and note some of its limitations as it has been implemented, including incompatible subsets, incompatible base sets, nonportability, and architectural dependence. We then introduce CPI-C, which is based to a great extent on APPC, and IBM's Systems Application Architecture (SAA) framework of which CPI-C is a part. We note IBM's intentions for CPI-C which, if sustained in implementation, may deliver a platform-independent, architecture-independent, and relatively simple interprogram interface. In the second part, we will focus on CPI-C. We will explore the market for CPI-C, the competition (e.g., Remote Procedure Call (RPC) from the UNIX-TCP/IP arena and the OSI Transaction Processing (TP) standard), and the evolution we expect in CPI-C.

(continued on page 2)

IBM Makes Partners of SNA and OSI

IBM's SNA represents the largest proprietary network architecture installed with over 50,000 licensed sites around the world. This makes SNA the leading de facto networking standard. For many SNA users, the addition of or migration to de jure standard OSI protocols is a significant element in their strategic networking plans. Although several users have been making their open standards investment in TCP/IP today and delaying their OSI push, most continue to emphasize their strategic commitment to OSI for multiprotocol connectivity.

(continued on page 10)

In This Issue

(continued from page 1)

The benefits of architecture-neutral APIs include the following:

- A significant improvement in programmer productivity
- A reduction to manageable levels of runtime environment incompatibilities
- An increase in the probability of projects being completed on time and within budget
- A significant reduction in retraining skilled personnel on different runtime platforms
- Significant easing of the classical rift between MIS and workstation developers
- A significant reduction in the life cycle of networked application development
- A marked improvement in organizational ability to respond to changing technologies
- A marked resistance to obsolescence of products because of a stable underlying architecture
- Application independence from vendor product life cycle
- Generation of applications that are kept transparent of delivery infrastructures
- The development of a common set of communication syntax and semantics
- The development of "standard" standards to allow colocation of multiple protocol stacks (SNA, OSI, and TCP/IP)
- A sense of predictability and assuredness in the overall network resource

APPC and LU 6.2

IBM introduced logical unit 6.2 (LU 6.2) in 1982 as its basis for platform-independent interprogram connectivity. (LU 6.2 was preceded by LU 6.0 and LU 6.1, which were comparatively quite restricted in scope.)

APPC = LU 6.2

There is some debate about whether there was a distinction between the terms LU 6.2 and APPC in the past (e.g., It has been said LU 6.2 was a protocol based on APPC services). In any case, IBM's official position today is that the two terms mean the same thing and are interchangeable.

LU Software versus API

However, there is a distinction between LU 6.2/ APPC and its API. LU 6.2 can be implemented with an API, but an API is not always required. For example, LU 6.2 implementations in the 6611 router and the 3174 do not have an API.

Further, in addition to the native APPC API, other APIs can be used to interface to LU 6.2. An obvious example, of course, is CPI-C.

The APPC API is an interprogram protocol boundary. In this role, the APPC API exists between a transaction program (TP) and layer six (Presentation Services) of the SNA architecture.

STPs and ATPs

Before proceeding with the discussion, it is important to introduce two more terms that identify the users of LU 6.2 and APPC: application transaction programs (ATPs) and service transaction programs (STPs). STPs are primarily programs that may be considered part of the network and provide a limited, specific set of network services while ATPs support user applications. ATPs are written on behalf of four principal end users: databases, files, programmable workstations (including PCs), and intelligent printers.

The placement of the APPC API is shown in Figure 1 (see page 3). In the figure, two ATPs are logically connected through a conversation.

Conversations are transaction-based relationships established between remote programs to share data. Each of the ATPs is locally associated with a separate LU 6.2 platform. Each LU 6.2 platform is, in turn, locally defined as a resource in an SNA node such as a physical unit 5 (PU 5) host or node type 2.1 (NT2.1) platform. The LUs are connected through a session and their underlying nodes are

shown as connected over a link. The APPC API acts as a structured protocol boundary between each ATP and its LU. The ATP and LU exchange verbs (interprogram calls), parameters, data, and return codes across the APPC API. The resulting conversation can be understood from two perspectives:

- From the ATP perspective, the conversation represents a logical connection through which to exchange data with another transaction program.
- The LU perspective is that the conversation is a temporary resource activated on an already active session with another LU.

Figure 2 provides a closer view of the APPC API from the LU 6.2 perspective. From this perspective, the ATPs are external to the LU, which includes all that lies in the shaded box. The ATP on the left directly sends to and receives from the APPC API. These send/receive sequences can carry three types of verbs: mapped conversation verbs, basic conversa-tion verbs, or type-independent verbs (verbs whose syntax is independent of mapped or basic conversations).

Basic Conversations

The nine base set verbs for basic conversations with the APPC API are:

- ALLOCATE. Requests conversation set-up
- CONFIRM. Requests that the remote transaction program (TP) confirm receipt of data
- CONFIRMED. Responds affirmatively to a CONFIRM request
- DEALLOCATE. Requests conversation conclusion

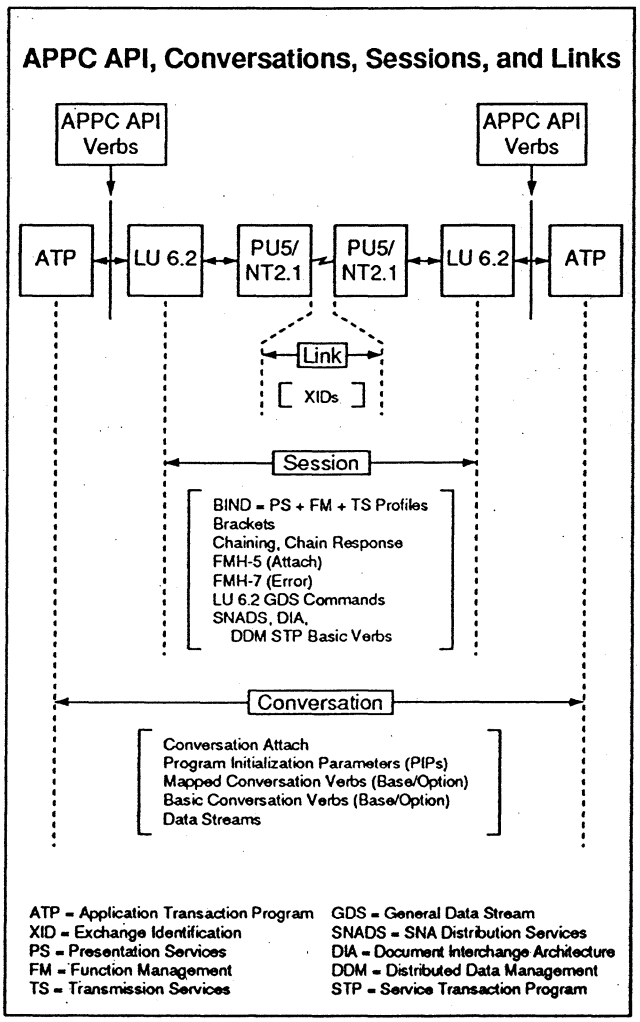


Figure 1

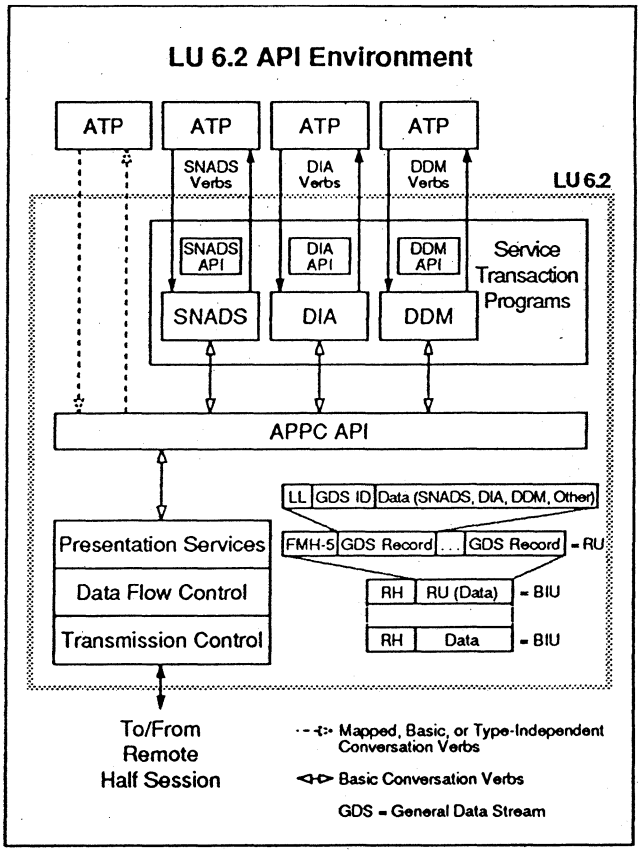


Figure 2

- GET_ATTRIBUTES. Requests conversation and LU characteristics
- RECEIVE_AND_WAIT. Requests the local LU to loop to receive incoming data
- REQUEST_TO_SEND. Requests the remote TP to allow the local TP to send data
- SEND_DATA. Requests that data be sent to the remote TP
- SEND_ERROR. Notifies the remote TP of an error

Basic API

The basic conversation API is generally used only for STPs internal to LU 6.2 to convey specific service. This is because basic conversations require that the transaction program (ATP or STP) using the API include a record length reference for all data sent across the interface. This means that the TP must be involved with partial construction of the resulting variable-length record stream.

Some of the specific services that use the basic conversation API are shown in Figure 2 and include SNA Distribution Services (SNADS), Document Interchange Architecture (DIA), and Distributed Data Management (DDM). SNADS is a general purpose store-and-forward distribution service to deliver files, software, revisable- and final-form text, as well as mixed object data. DIA is an electronic office set of services including document distribution (the basis of SNADS), document library, application processing, file transfer, and session. Whereas SNADS is a batch service, DIA is interactive. DDM enables a local TP to access a remote, record-oriented file or a remote set of relational database records. All three are server-requester environments based on a client-server model.

It would be unusual for a user to write LU 6.2 programs using basic conversations. IBM recommends that user applications all be written using mapped conversations.

Mapped Conversations

ATPs default to mapped conversations for their logical connectivity. Mapped conversations provide data formatting services that programs would otherwise have to perform themselves. The mapped conversation verbs for the APPC API are the same as the basic conversation verbs with the inclusion of the prefix "MC_" (e.g., MC_ALLOCATE).

Mapped APIs

The mapped conversation verb APPC API can be written in macroassembler. However, it is generally written in a third-generation language (3GL) such as COBOL, C, RPGII (System/36), RPGIII (System/38), RPG/400 (AS/400), or PL/I or a fourth-generation language (4GL) such as REXX.

Issues with APPC APIs

Several issues have emerged with the APPC API. Its limitations are:

- Incompatible subsets of option set verbs
- Incompatible base sets
- Nonportability
- Architecture dependence

Incompatible Subsets of Option Set Verbs

The majority of APPC products provide the base set of nine verbs as well as some of the large number of option set verbs provided by the APPC API for both basic and mapped conversations. However, various implementing platforms (MVS CICS, VM TSAF, OS/400, OS/2 EE, DOS) support incompatible subsets of the option set verbs. The result is a wide range of inconsistent functionality among the various implementations.

Incompatible Base Sets

Even more striking, the various implementations of the base set of verbs are inconsistent across platforms. In other words, although the underlying architecture is consistent, the syntax of the native API is implemented differently across different platforms. As an example of two workstation

versions, OS/2 EE and DOS APPC implementations share several verbs and yet are quite at variance in the others they each support. For another example, the host CICS implementation of APPC API verbs and supporting parameters is handled in EXEC CICS statements and What_Received Indications are stored as Executive Information Block (EIB) fields. This approach is quite different from the workstation implementations and exacerbates the classic rift between host developers in MIS and departmental workstation developers.

Nonportability

Significant rewrites of code are still necessary in order to move TPs from one platform to another. In other words, while a range of APPC-supporting platforms may all support ATPs being written in a specific 3GL or 4GL, APPC TP code is still, to a great extent environment-dependent. The direct result is duplication of effort to propagate interprogram functionality across the enterprise.

Architecture Dependence

The APPC API was originally developed to provide for interprogram connections in hierarchical, host-centric SNA networks. While the APPC API and LU 6.2 have evolved to support underlying peer connections, the APPC API nonetheless remains SNA-specific.

However, it must be noted that the APPC API and supported verbs and parameters were proposed to the International Standards Organization (ISO) early on to assist in its development of a standard for transaction processing. These provided a significant input to the development of the OSI TP service and protocol standard.

Need for CPI-C

For these reasons and others, IBM determined that it was necessary to evolve the APPC API to a greater degree of platform and architecture independence. This resulted in the specification of CPI-C, which is indigenous to SAA.

The Context for CPI-C: SAA

Although this article is not a presentation on SAA, a brief review of SAA concepts and terms is important to be able to understand the context for CPI-C.

SAA is a collection of selected software interfaces, conventions, and protocols that is intended to provide the framework for development of consistent applications across the SAA strategic platforms—the S/390/370, AS/400, and PS/2 processor families. SAA provides for cross-system consistency by defining and standardizing on selected IBM-licensed program interfaces, conventions, and protocols. These, in turn, provide a common platform for application development, portability, execution, and communication across the SAA processor architectures.

The SAA architectural components are Common User Access (CUA), Common Communications Support (CCS), and Common Programming Interface (CPI). CUA provides a window-based, object-oriented, and interactive user interface to applications. CCS provides the interprogram connection and networking component of SAA.

Common Programming Interface

CPIs include SAA programming languages and interfaces, one of which is CPI-C. CPIs have four major objectives, all of which seek to support end-user services through CUA by providing application developers with the requisite interfaces to achieve the overall SAA goal of application portability among SAA environments. The four CPI objectives are:

- End-user consistency
- Programmer productivity
- Interprogram transparency
- Enterprise-wide application development

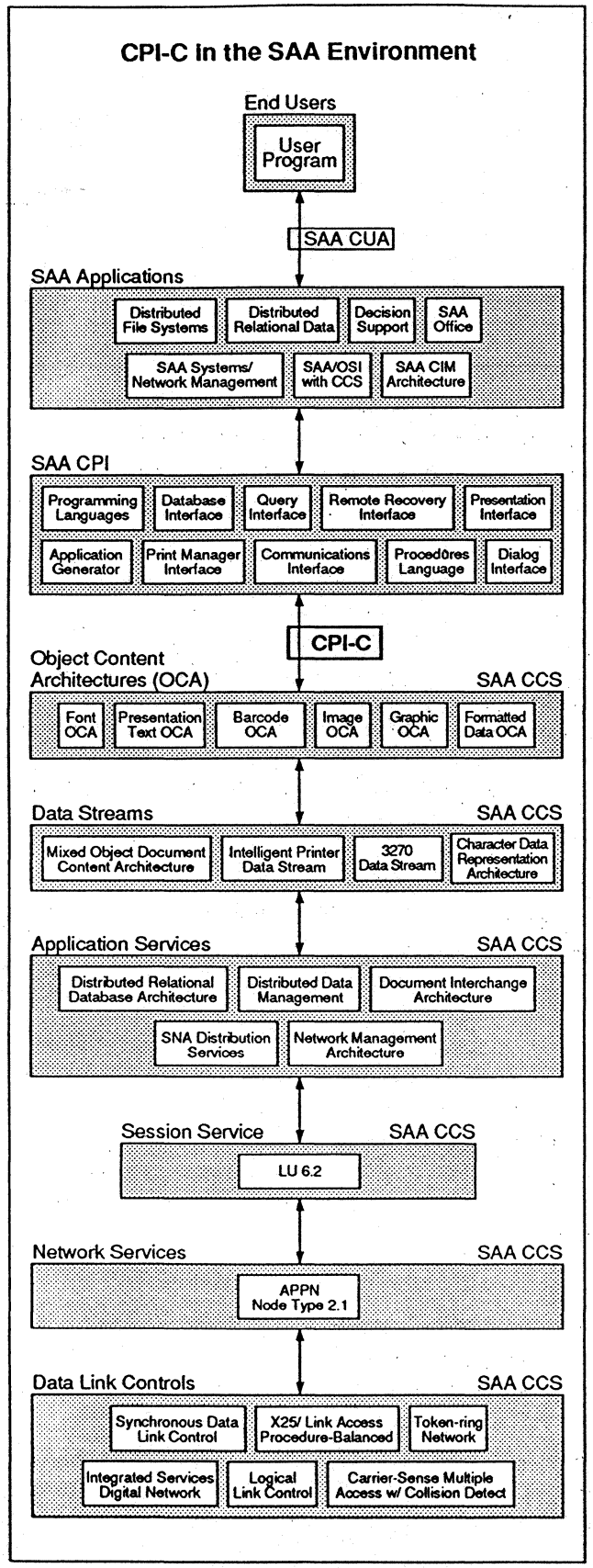


Figure 3

Figure 3 provides an overview of the SAA architectural environment.

Viewed from the top down, the figure shows the flow between the SAA components:

- End users and their applications gain access to local and remote SAA applications through the CUA.
- These applications, in turn, are written in an SAA programming language (in the CPI component of the figure).
- Local SAA applications can request access to remote programs through various services (again in the CPI component) such as the database interface, query interface, repository interface, etc.
- All of these applications interface with underlying CCS (interprogram and networking) services via CPI-C. These CCS service categories include object content architectures, data streams, application services, session services, network services, and data link controls.
- LU 6.2 is defined as the sole CCS session service.
- APPN/NT2.1 is defined as the CCS network services component.
- SAA CCS data link controls support a range of WAN and LAN interfaces.

CPI-C

Since LU 6.2 is the sole SAA session service, CPI-C provides the interprogram services necessary to interconnect SAA applications through conversations running in LU 6.2 sessions. To this extent, as stated earlier, CPI-C is architected on the basis of the APPC API.

Figure 4 (see page 7) provides an overview of the CPI-C relationship to interconnected programs and their local LUs. In this sense, CPI-C is found in the same location as its APPC API predecessor. Workstation program D connects to host program B in a program B-to-program D conversation. This conversation, in turn, runs as an active resource in an

LU 6.2 session established between LU x at the host site and LU z at the workstation location.

Benefit of CPI-C over APPC API: Transparency

One of the most significant accomplishments of CPI-C over APPC API is that a local program requesting access to a remote program is able to maintain a much higher degree of transparency from the underlying connection services. (See the sidebar "CPI-C Simpler Than APPC API.")

The CPI-C approach is far less complex and laden with overhead for the local program than the APPC API procedure. Also the CPI-C program, unlike APPC API, is not required to have extensive knowledge of all of the characteristics of the remote program.

Summary

IBM has implemented CPI-C on all of the SAA platforms. *SNA Perspective* believes that CPI-C will soon also be supported on DOS and on the RS/6000 as the strategic AIX platform. This generalization of CPI-C to both SAA and AIX environments will greatly assist in engendering portability among those application platforms. In this sense, we would consider CPI-C capable of providing the basis for a truly architecture-neutral interprogram service.

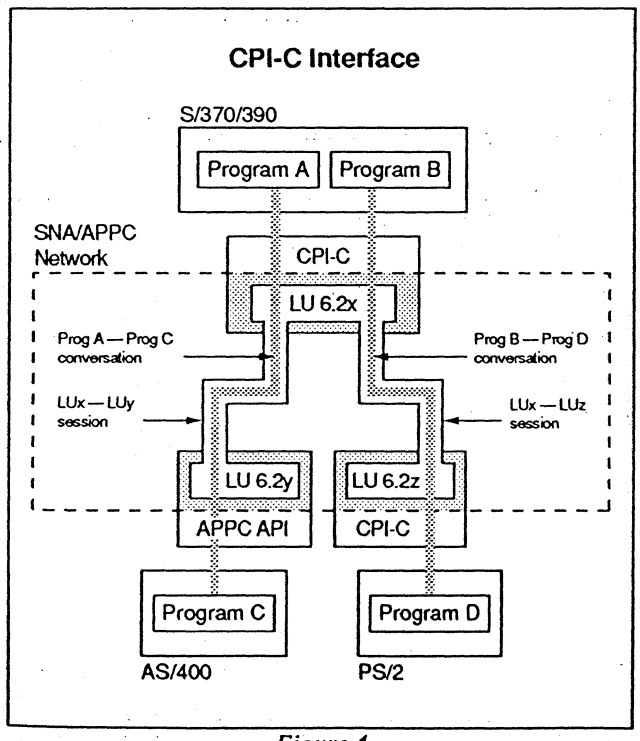


Figure 4

In the second article in this two-part series, we will conclude our exploration of CPI-C by analyzing the following:

- The architecture-neutral imperative
- The importance of interprogram interfaces and CPI-C to users, IBM, and other vendors
- APPC/CPI-C coexistence
- CPI-C platforms
- IBM's experience with CPI-C as a common SNA/OSI transaction processing interface

We will also include:

- A comparison of CPI-C to RPC
- CPI-C in the standards bodies, including its adoption as an X/Open specification
- A discussion of expected CPI-C directions, which include:
  - Additional SNA/OSI services
  - A SNA-TCP/IP common interface
  - A SNA-OSI-TCP/IP common interface
  - CPI-C as the basis for multiple protocol boundaries
  - Improved server support in CPI-C including non-blocking calls and support for multiple incoming conversations
  - CPI-C support for X.500 directory services
  - Automatic data conversion (e.g., EBCDIC to ASCII)

by Thomas J. Routt ■

CPI-C Simpler Than APPC API

APPC

A program that interfaces with APPC API and wishes to begin a conversation with a remote APPC API program must issue several supplied parameters (modifiers) with the ALLOCATE or MC_ALLOCATE verb. For example, a local OS/2 EE TP could potentially have to issue, along with an MC_ALLOCATE verb, parameters in the following format:

```
MC_ALLOCATE with
   tp_IDvalue,
   partner_LU_aliasname,
   mode_namename,
   tp_namename,
   return_control (WHEN_SESSION_ALLOCATED,IMMEDIATE, or WHEN_SESSION_FREE),
   synch_level (NONE or CONFIRM),
   security (NONE, SAME, or PGM-USER_IDname, PASSWORDname),
   pip_data_lengthn,
   pip_dataaddress.
```

- The TP_ID identifies the TP that is issuing the verb.
- Partner_LU_alias is the name of the partner LU (in ASCII) as it is known to the local TP. This name is generated as an eight-character ASCII string.
- Mode_name is the logon mode name (in EBCDIC) that designates allocated session characteristics. This name is generated as an eight-character EBCDIC character string. When an SNA subarea network is involved in the conversation, the mode name determines the Class of Service (COS—from the subarea perspective, the combination of Virtual Route Number and Transmission Priority Number). Mode_name is a Table Assembler (A) name.
- TP_name is an EBCDIC name of the partner TP that is less than 64 characters and is a Table Assembler Extended (AE) name.
- Return_control indicates when the local LU returns control to the local TP following request for a session resource.
- Synch_level states the conversation synchronization level that can be used. Synch_level NONE indicates that the conversation cannot issue any verbs or recognize any returned parameters that are associated with synchronization. Synch_level CONFIRM indicates that the conversation partners can issue verbs and/or recognize returned parameters that relate to confirmation processing.
- Security specifies the information that the partner LU uses to validate access to the partner TP resources.
- PIP_data_length*n* refers to the length of any program initialization parameters (PIPs) for the partner TP. PIP data can range from 0 through 32,767 bytes.
- PIP_data*address* states the address of the PIP data (carried as a single GDS logical record) which the local TP sends to the partner TP. PIP data are a means of passing initial parameters or environment setup information to a target TP or operating system.

CPI-C Simpler Than APPC API (continued)

CPI-C

Figure 5 shows that the CPI-C environment is far simpler than APPC API, at least from the TP perspective. In this figure, a local program is running under the control of the local operating system in the local node.

Because CPI-C is part of SAA, the node is NT2.1. Node services available to the local program include:

- System administrator set/access of side information
- Program start-up processing
- Program normal/abnormal termination

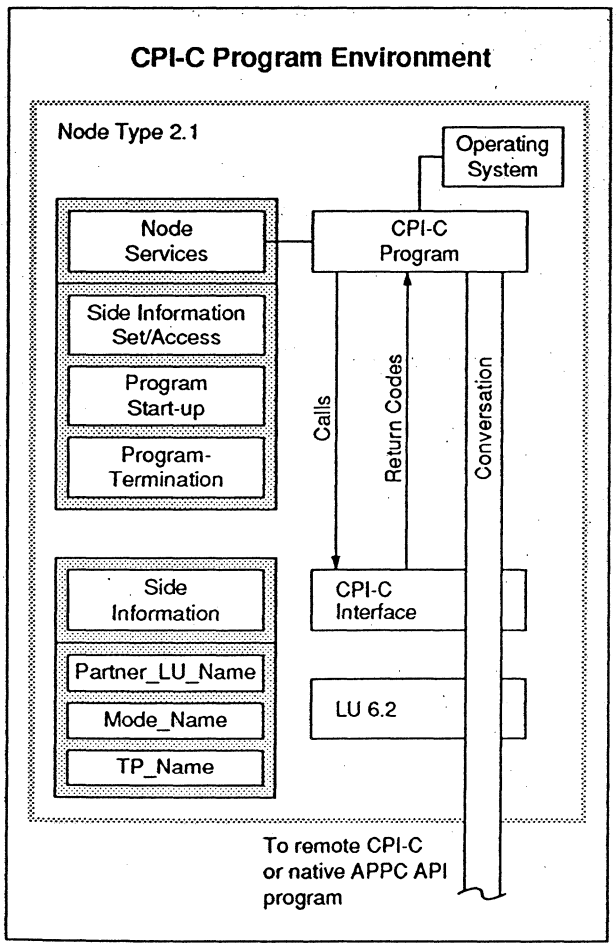


Figure 5

Side information is initialization information required for the local program to communicate with a partner (distributed) program. Side information is accessed by a symbolic destination name (*sym_dest_name*) and is described below.

- Partner_LU_Name; indicates the name of the LU with which the desired partner program is associated. This LU name can be any name for the remote LU that is recognized by the local LU for the purpose of allocating a conversation between the local and partner (remote) programs.
- Mode_Name; used by LU 6.2 to specify the properties of the session that will be allocated for the conversation. Session properties can include, for example, class of service. Modes names are shared between specific LU pairs and are unique at a given LU relative to a particular partner LU. Mode names for different partner LUs are independent in that the same mode name can correspond to different sets of session characteristics for different partner LUs.
- TP_Name; specifies the name of the remote transaction program.

This is simpler than the APPC API case given above. The procedure is as follows:

- A local program requests access to a conversation with a remote program by issuing a straightforward set-up call in the form INITIALIZE_CONVERSATION which is accompanied by a symbolic destination name. (Note: "Calls" are used for CPI-C where "verbs" are used for APPC API.)
- The symbolic destination name is used by local node services to look up all appropriate information necessary to establish a conversation.
- Once the lookup is complete, node services presents a conversation ID back to the local, requesting program.
- At that point, the local program is able to begin sending data on the conversation. ■

(continued from page 1)

This article focuses on IBM's OSI approaches and products across all of its major platforms. We emphasize specific means for OSI-SNA coexistence and integration. We also discuss challenges IBM has faced in seeking success in the OSI world.

In an upcoming issue of *SNA Perspective*, we will systematically explore the approaches and implications of IBM's TCP/IP products and their coexistence with SNA.

Several SNA users wanted to integrate SNA and OSI environments but have traditionally encountered several obstacles:

- **Host-centric.** IBM OSI products were, for several years, restricted to host platforms. These earlier products worked reasonably well for remote attachment of OSI/X.25 devices or OSI/asynchronous devices to host SNA applications. The OSI-SNA conversions occur in the host under VTAM. Unfortunately, these products emphasized host-centric networking and did not support the peer-to-peer connections characteristic of the OSI direction.
- **Non-"standard" implementations.** Various manufacturers and software developers have tended to create dissimilar OSI solutions. In Europe, for example, multiple, incompatible X.400 (electronic messaging) approaches exist. As a result, there are still no well-implemented "standard" standards.
- **Less functional, less efficient, more expensive.** Users have long perceived that standardization by committee is an arduous, time-consuming process at best and, when finally completed, tends to result in products that are, at least initially, less functional, less efficient, and more expensive than the proprietary products they are intended to replace. Users need solutions now and have tended to commit, at least in the near term, to proprietary architectures and to TCP/IP for multivendor connectivity.

IBM's Threefold Networking Strategy

For several years, IBM has steadily pursued a threefold networking strategy with SNA, TCP/IP, and OSI. Its current balance is to:

- Continue to develop SNA as a proprietary network architecture
- Support TCP/IP and related applications as one multivendor networking approach
- Develop OSI products as another multivendor networking approach

Develop SNA As A Proprietary Network Architecture

SNA was historically a host-centered and relatively closed architecture. Though still proprietary in the sense that it is internally controlled and developed by IBM, SNA has become increasingly open through various APIs. The architecture has also been evolving toward interprogram and peer-to-peer networking through LU 6.2, CPI-C, and APPN. (The other feature article in this issue of *SNA Perspective* is Part I of a two-part analysis of LU 6.2 and CPI-C.)

TCP/IP As One Multivendor Approach

IBM could not ignore the fact that users have been investing recently in TCP/IP much more than in OSI solutions. IBM stated clearly in 1991 that it has heard its customers say TCP/IP is no longer an interim protocol for them. Therefore IBM has made TCP/IP just as strategic to its multiprotocol direction as OSI and has significantly stepped up its investment in TCP/IP support.

IBM now regards SNA, OSI, and TCP/IP as three equally important networked application architectures. This commitment is systematically revealed in the rollout of products across its platforms.

OSI As Another Multivendor Approach

While several commercial, governmental, and university customers of IBM have adopted TCP/IP as the basis for heterogeneous network integration today, a large proportion of customers remain strategically aligned to OSI solutions.

In addition to expecting an increased OSI investment by the mid-1990s, *SNA Perspective* believes that, during the decade, many OSI components will be integrated into new iterations of both SNA and TCP/IP at several levels. For example, we expect the IP addressing structure to be "enhanced" with a structure much like the OSI approach, and we believe that NetView will increasingly be internally altered toward alignment with CMIS/CMIP. In this way, OSI migration will be taking place "under the covers."

IBM's OSI Approach

IBM continues to regard OSI as a significant strategy for integrating multivendor, multiprotocol networked environments. IBM has made significant research and development commitments for over ten years now to develop OSI and SNA-OSI integration products across its strategic processing platforms. IBM's OSI position has several elements:

- Recognition of the OSI business case
- Support for TCP/IP-to-OSI transitions
- Support for OSI-SNA integration
- Support for platform independence
- Endorsement of worldwide profiles
- Endorsement of conformance testing
- Provision of WAN and LAN interfaces
- Embracing of multivendor network management
- Support for multivendor directory

OSI Business Case

A significant and growing business case for OSI has emerged in enterprises strategically opposed to proprietary architectures. Most large corporations and government agencies are beset with multiple, competing network solutions. It is not uncommon for large and midsized enterprises to have more than ten internal, incompatible network architectures. Several users have adopted a two-phase strategy to reduce this incoherence to manageable proportions:

- Reduce the selected network approaches to SNA, TCP/IP, and OSI as a tactical solution
- Align with OSI as a single, strategic multivendor networking solution

Many users have extended their timeframe for continuing with the three-architecture tactical approach and have continued heavy investments in TCP/IP while keeping many OSI projects as pilot projects, finding TCP/IP standard enough for most current needs as well as being more available and less expensive.

To ensure successful OSI migration, most enterprise users either have their own internal standards organizations and/or are members of international standards organizations such as ISO, CCITT, CEC, CEN/CENELEC, ANSI, NBS, NIST, IEEE, COS, SPAG, and POSI.

TCP/IP-to-OSI Transitions

Through its support for transitions from TCP/IP to OSI, IBM intends to meet the needs of major corporate and government users who maintain their endorsement of an OSI strategy while continuing invest in TCP/IP applications and networks today. IBM must support its users' investment in TCP/IP applications as they migrate to OSI in the long run.

OSI-SNA Integration

Native, seven-layer, fully compliant OSI products have been developed by IBM while it maintains SNA as a proprietary network architecture. Whereas IBM OSI products have historically been SNA-integrated at the data link control level (SNA Layer 2), key directions include transparent OSI-SNA integration at higher layers through APIs and SAA CPIs. These interfaces could be accessed in all SAA and AIX programming languages.

Platform Independence

OSI interfaces will be developed consistently across IBM's SAA and AIX platforms. SAA Common Communications Support (CCS) supports consistent implementation of seven-layer OSI suites across the SAA platforms—MVS, VM, OS/400, and OS/2. The AIX family of products also supports SNA and OSI as well as TCP/IP.

IBM OSI Product Announcement Timeline

1977 First SNA/X.25 support.

1978 Initial X.21 support.

1980 X.25 interface through NCP Packet-Switching Interface (NPSI).

1983 Initial IBM support for OSI layer 3, 4, and 5 protocols (in Europe).

1985 Token-Ring Network (IEEE 802.2/ISO 8802-2 LLC and IEEE 802.5/ISO 8802-5 MAC).

1985 File Transfer Access and Management (FTAM, ISO 8571) for Series/1 on MAP 2.1 IEEE 802.4 token bus.

1986 Support for Ethernet/IEEE 802.3 CSMA/CD LAN on the 9370.

1987 In Europe, System/370 products providing VTAM and CICS interfaces into OSI, including:

- Message Transfer Facility (MTF). A VTAM facility that provides X.400 MHS message transfer agent entity (MTAE) to/from X.400 and DIA and SNADS.
- X.400 DISOSS Connection. CICS-based, maps X.400 MHS and DIA/SNADS protocols.
- General Teleprocessing Monitor for OSI (GTMOSI). Supports VTAM API and interconnects terminals and applications via OSI transport and session layer protocols.
- Open Systems Transport and Session Service (OTSS). Supports SNA application interface into ISO transport classes 0 and 2 as well as ISO session kernel.
- Open Systems Network Service (OSNS). Enables SNA applications to connect to SNA or non-SNA applications through an X.25. OSNS interworks with NPSI.

1988 IBM introduced SAA with the overall goal of openness.

1988 Comprehensive set of OSI services and protocols added to the CCS component of SAA.

1988 IBM announced OSI/Communication Services (OSI/CS).

1988 OSI/File Services provides an MVS/VM FTAM API and interworks with OSI/CS.

1988 X.400 support includes:

- Open Systems Message Exchange (MVS and VM hosts). OSME provides X.400 MHS services for multivendor messaging environments.
- X.400 DISOSS Connection (MVS/CICS) provides links between DISOSS and OSME.
- X.400 PROFS Connection (VM) provides links between PROFS and OSME.

1989 SAA Computer Integrated Manufacturing (CIM) architecture. In SAA and AIX, designed to support CIM implementation. Uses AD/Cycle CASE tools for automated application development. Conforms to PDES, PHIGS, MAP 3.0, and CALS.

1989 CIM Communications and Data Facility (CDF) provides CIM data management services for CIM data and enables support for the CIM operational repository and data store.

1989 MAP 3.0 SAA CIM products include:

- 3172 Interconnect Controller. Channel-attaches LANs to hosts for TCP/IP and MAP 3.0 access.
- OSI/Manufacturing Messaging Services (OSI/MMS) for OS/2 and VM. Supports ISO 9506.

1990-1992 SAA incorporates OSI standards at all seven networking layers for LANs as well as WANs.

■

Table 1

Worldwide Profiles

Users require OSI products that are consistent with worldwide profiles. These profiles account for national and regional implementation variations and include US GOSIP Version 1, UK GOSIP Version 3.0, and INTAP. IBM's OSI products conform to these, and its commitment to US and UK GOSIP remain clear.

Conformance Testing

Users require independent confirmation that OSI products from manufacturers and third-party vendors interface consistently. IBM OSI product interoperation and conformance testing is conducted through membership in EurOSInet, INTAPnet, and OSINET. IBM also maintains active roles in COS, SPAG, and POSI to ensure multivendor OSI product harmonization.

WAN and LAN Interfaces

Link connections for both SNA and OSI are being consistently implemented over LANs as well as WANs. These interfaces include X.25, Ethernet (ISO 8802/3 CSMA/CD), ISO 8802/4 token bus, ISO 8802/5 token ring, FDDI, and ISDN. These are, to a great extent, consistently implemented across IBM's SAA and AIX platforms.

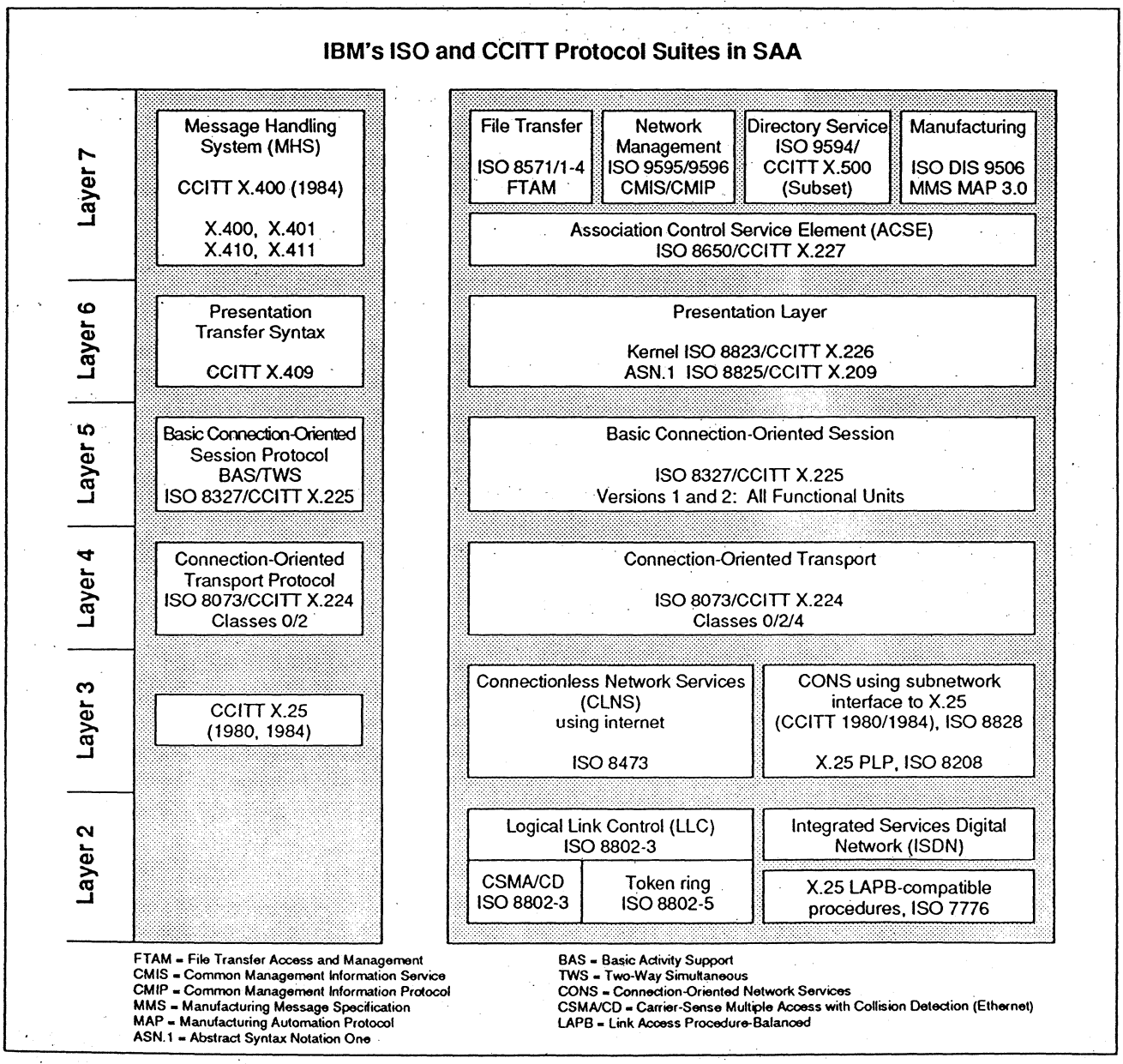


Figure 6

Multivendor Network Management

OSI multivendor network management is provided through the Common Management Information Service (CMIS; ISO 9595) and Common Management Information Protocol (CMIP; ISO 9596). TCP/IP multivendor network management is provided through the Simple Network Management Protocol (SNMP). IBM is supporting management of both CMIS/CMIP and SNMP networks and devices through NetView and the SystemView framework.

Multivendor Directory

Multivendor exchange of directory information is accomplished through the CCITT X.500/ISO 9594 Directory standard. IBM supports X.500 to provide multivendor directory support.

IBM OSI Products

A timeline of IBM OSI product introductions is provided in Table 1 on page 12.

IBM's OSI Protocol Suites

Figure 6 (see page 13) presents the OSI protocol suites that IBM has selected for inclusion in SAA. At layer 7, these provide a wide range of networking functions, including vendor-independent electronic messaging, file transfer, network/systems management, directory service, and distributed manufacturing. These services are well defined and are supported by a standard set of underlying services and protocols in layers 1 through 7.

Table 2 summarizes IBM's OSI networking products and shows that OSI products have been developed across SAA and AIX. This range of support is in marked contrast to host-only OSI support from IBM

IBM SAA and AIX OSI Platforms and Products

Functions Platforms:	MVS	VM	OS/400	OS/2	RS/6000 (AIX)
Applications					
NetView	OSI/CS, NetView	OSI/CS, NetView	OSI/CS/400	OSI/CS OS/2 EE	———
OfficeVision	OV/MVS, ONDS	OV/MVS, ONDS	———	——— (3/91)	———
Distributed Automated Edition	———	DAE	———	DAE	———
Remote Programming Interface	VTAM V3 OSI RPI	VTAM V3 OSI RPI	———	Future	———
Layer 7					
X.400 Msg. Handling System	ONDS/MVS, XDC	ONDS/VM, XPC R2	Future	Future	AIX OSIMF/6000
FTAM	OSI/FS R1 for MVS	OSI/FS R1 for VM	OSI/FS/400	OSI/FS/2	AIX OSIMF/6000
Manufacturing Message Services	———	OSI/MMS for VM	———	OSI/MMS for OS/2	Future
C/COBOL API	OSI/CS, RPI	OSI/CS, RPI	OSI/CS/400	OSI/CS OS/2 EE	———
Layers 3-7					
X.500 Directory	OSI/CS for MVS	OSI/CS for VM	OSI/CS/400	OSI/CS OS/2 EE	———
CMIS/CMIP	↓	↓	↓	↓	Future
APIs	↓	↓	↓	↓	AIX OSIMF/6000
ACSE	↓	↓	↓	↓	↓
Presentation	↓	↓	↓	↓	↓
Session	↓	↓	↓	↓	↓
Transport	↓	↓	↓	↓	↓
Connectionless Network Service	↓	↓	↓	↓	↓
Connection-Oriented Network Service	↓	↓	↓	↓	↓
Connectivity					
WAN X.25	OSI/CS for MVS	OSI/CS for VM	OSI/CS/400	OSI/CS OS/2 EE	AIX OSIMF/6000
ISDN	OSI/CS for MVS	OSI/CS for VM	———	Future	———
802.3 CSMA//CD (Ethernet)	OSI/CS for MVS	OSI/CS for VM	Future	OSI/CS OS/2 EE	AIX OSIMF/6000
802.4 Token Bus	Future	CCI MAP 3.0 Adapter	———	CCI MAP 3.0 Adapter	———
802.5 Token Ring	Future	Future	Future	OSI/CS OS/2 EE	AIX OSIMF/6000
FDDI	Future	Future	———	———	———

Table 2

in previous years. *SNA Perspective* believes this suggests that the company intends to provide for client-server capabilities in multivendor, multiprotocol networked application environments.

OSI/Communication Services

OSI/CS is IBM's most important single OSI product for several reasons:

- OSI/CS enables OSI connectivity across enterprise, departmental, and workstation platforms.
- OSI/CS provides end user APIs, which can be written to in C or COBOL, to OSI services.
- These OSI/CS APIs enable significant vendor-independent networking services such as directory, network/systems management, and file transfer.
- OSI/CS supports the US GOSIP and UK GOSIP profiles, which is especially interesting since the US Department of Defense (perhaps the largest user of TCP/IP) intends to eventually supplant TCP/IP with OSI.
- The OSI/CS APIs allow for a reasonable degree of underlying network transparency to enabling applications.

RPI: OSI Applications on One Host, OSI Stack on Another

The VTAM OSI Remote Programming Interface (RPI) represents a casual user approach to OSI. RPI allows MVS and VM SNA hosts to participate in OSI networks without the need to install OSI/CS on each system. At least one host must have OSI/CS; the other hosts can run OSI applications without a resident OSI stack. Any calls to the OSI network are intercepted by RPI, enveloped in LU 6.2, and routed across the SNA network to a host with the full OSI stack. RPI started shipping in 1990 for VTAM Version 3 in MVS/ESA/XA and VM/SP.

File Transfer

IBM intends to provide support for File Transfer Access and Management (FTAM), OSI's file transfer standard, across all SAA and AIX platforms. IBM's FTAM capability complies with ISO 8571 FTAM as well as NIST implementation agreements and CEN/CENELEC ENV 41204 Profile. Release 1 of OSI/File Services (OSI/FS), which implements FTAM on the host, shipped for MVS and VM in 1990. AIX OSI Messaging and Filing/6000 (AIX OSIMF/6000) provides both FTAM and electronic mail (X.400 MHS) services. These FTAM and X.400 capabilities can interconnect with other OSI FTAM and X.400 processing environments over Ethernet, token ring, or X.25 networks.

Electronic Mail

Table 2 also summarizes IBM X.400 support. MVS/VM X.400 products provide interchange between SNADS/DIA and MHS. Open Network Distribution Services (ONDS) for MVS and VM provide an X.400 feature that, in conjunction with OSI/CS, interoperates with other X.400 message transfer agents (MTAs).

ONDS has functionally replaced IBM's earlier Open Systems Message Exchange (OSME) and Message Transfer Facility (MTF) and provides X.400 interpersonal messaging (IPM) for OfficeVision/VM, PROFS, VM/CMS, OfficeVision/MVS, and DISOSS. This is supported through ONDS/VM and X.400 PROFS Connection on VM and ONDS/MVS and X.400 DISOSS Connection on MVS.

SNA-OSI Integration

IBM OSI solutions are now well established across the SAA and AIX platforms. However, coexistence is often not enough. Several users have existing SNA networks and require SNA-OSI interconnections transparently. These needs include:

- SNA traffic over OSI networks
- OSI traffic over SNA networks
- Dual-stack SNA-OSI protocols

SNA Traffic Over OSI Networks

CSFI

General Teleprocessing Monitor for OSI (GTMOSI) was introduced in 1987 by IBM France. In July 1990, a GTMOSI offering, Communications Subsystem for Interconnection (CSFI), was introduced and made available in Europe and Canada. CSFI evolved from requirements to interconnect heterogeneous SNA, OSI, and non-IBM processing environments in the French banking industry and Swiss public sector. CSFI is a VTAM application and, like GTMOSI, is a telecommunication monitor.

CSFI is supported by MVS, VM, and VSE VTAM (see Figure 7). CSFI provides RELAY and MAPPER programs to enable SNA sessions between the primary LU (PLU) and secondary LU (SLU) in the host. Connections can be established between SNA applications and 3270 devices/subsystems, between SNA applications and non-IBM devices/subsystems, and between SNA applications and OSI devices/applications. CSFI interacts with X.25 NPSI in a 37xx communication controller for X.25 connections. Perhaps the greatest CSFI asset is that it provides a vital bridge to strategic OSI offerings for users who choose not to migrate in one step. CSFI provides a means to protect user investment in host-based OSI applications.

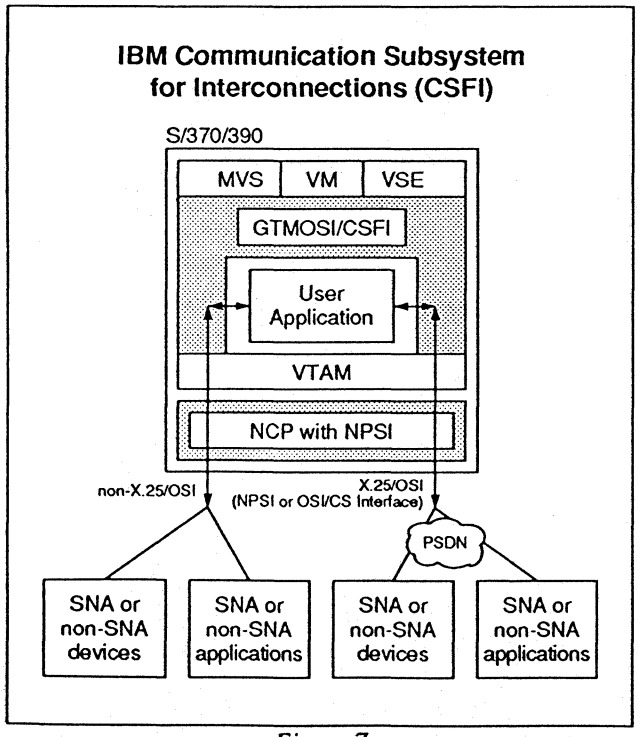


Figure 7

X.25

The general approach to relay SNA data traffic through OSI is to encapsulate SNA path information units (PIUs) in X.25 packets (see Figure 8). SNA PIUs are encapsulated in X.25 data packets. These packets are then routed through an X.25 interface. In this approach, OSI/X.25 connectivity is provided between a pair of SNA applications or between an SNA application and an SNA device through an intervening, non-SNA network. IBM products that support SNA over X.25 include NPSI, host integrated communications adapter (ICA), RS/6000, AS/400, System/36, System/38, System/88, 3174, and PS/2.

OSI Traffic Over SNA Networks

The next major scenario is to transport OSI data traffic over SNA backbone networks. From an OSI perspective, the requirement is to relay OSI traffic through a non-OSI subnetwork, in this case SNA. The general approach is to encapsulate OSI protocols in SNA headers and trailers. Again from the OSI viewpoint, interconnection systems designated as interworking units relay subnetwork independent convergence protocols (SNICP) over subnetwork dependent access protocols (SNAcP).

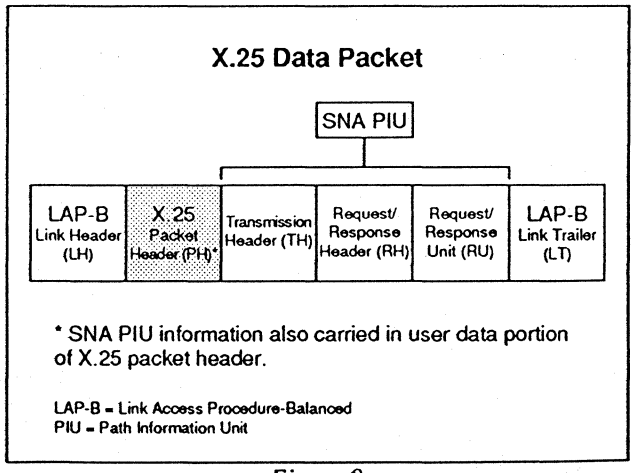


Figure 8

X.25 Interconnnection with XI

The principal SNICP protocols used by IBM include CCITT X.25/ISO 8208 for connection-oriented networks and Connectionless Network Protocol (CLNP) for connectionless networks. IBM products that provide this function at present include NPSI under NCP in the 3745 and X.25 SNA Interconnection (X.25 XI), also an NCP program.

XI supports connections between X.25 DTEs into NPSI as well as between each other through an intervening SNA subarea network. In essence, how X.25 XI functions can be thought of as the inverse of NPSI. That is, while NPSI provides SNA-to-SNA or SNA-to-non-SNA connections across X.25 networks, XI connects X.25 DTEs to X.25 DTEs through an intervening SNA subarea backbone network. The requirement for XI originally emerged in Europe in the mid-1980s. Several IBM SNA networks in Europe span multiple countries and users wanted to interconnect their X.25 terminals and hosts between multiple countries without having to transit X.75 gateways or deal with multiple European PTT government agencies. The opportunity for IBM, through X.25 XI, was to enable these connections through existing SNA subarea networks which were already multinational in scope.

SNA Perspective believes that the natural next step from supporting OSI over SNA will be to transport non-SNA traffic (e.g., OSI, TCP/IP, NetBIOS) across APPN as well as subarea SNA networks.

Dual-Stack SNA-OSI Protocols

Perhaps the most intriguing SNA-OSI interconnection solution is to locate multiple protocol stacks in the same operating platforms. Users would benefit from solutions that integrate SNA and OSI, for example, in common SAA and AIX platforms. Unlike the two previous scenarios, this one is not based on internetworking of SNA and OSI traffic but, rather, would be provided by architecture-independent APIs.

CPI-C and OSI TP

SNA Perspective believes that a workable approach for dual-stack SNA-OSI protocols would be to enhance the SAA CPI-C interface (shown in figure 9) to enable protocol-independent application calls to and from a variety of underlying networking architectures, including SNA, OSI, TCP/IP, and NetBIOS. The second part of the two-part CPI-C article begun in this issue of *SNA Perspective* will elaborate more on this approach. One significant advantage of a common, architecture-independent-API CPI-C approach would be that application developers could write applications that could utilize SNA, OSI, TCP/IP, and/or NetBIOS networking protocols transparently, that is, without the need to encode architecture-dependent syntax into communications modules.

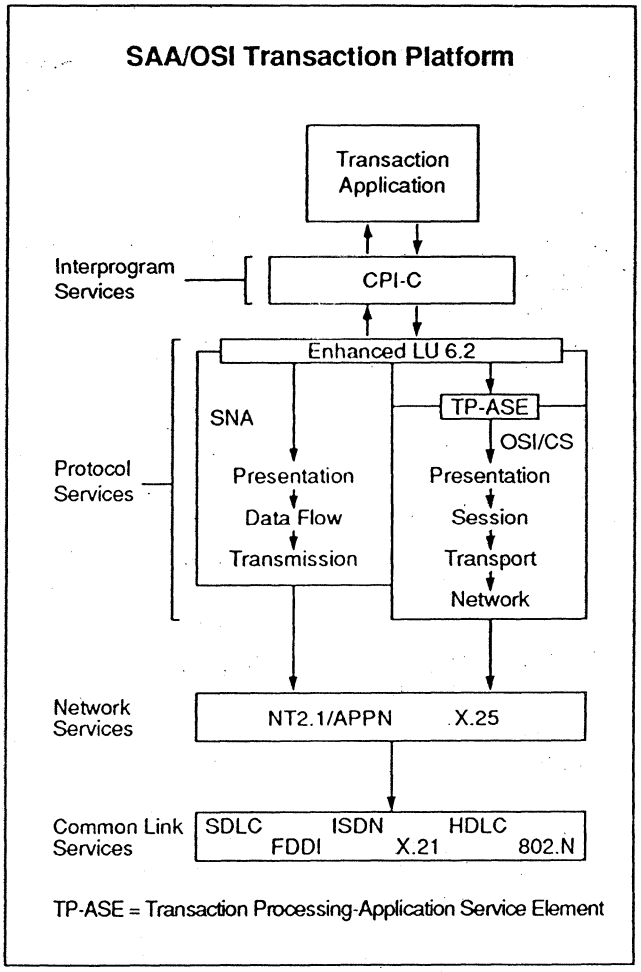


Figure 9

Figure 9 provides a view of this possible co-location of SNA and OSI protocol stacks that are specific to a transaction application environment. The approach shown here is based on a paper published by IBM in August 1990 called "Mapping IBM's CPI for Communications onto OSI Distributed Transaction Processing Services." This paper describes a mapping between CPI-C and the kernel, polarized control, and handshake functional units described in *OSI Distributed TP—Part 2: Service Definition,* September 1989. Local OSI applications would interconnect with remote OSI applications by exchanging interprogram calls across CPI-C.

Most CPI-C calls can be mapped to OSI TP. This is not surprising since OSI TP is based, to a great extent, on LU 6.2 logic. LU 6.2 could therefore support both SNA layers 6, 5, and 4 and an OSI TP-ASE interface. A common LU 6.2/OSI TP application interface could be provided across platforms and allow upper-layer SNA/OSI services to share common network and link services.

Remote transaction processing (TP, ISO 10026) defines TP and a framework for coordinating across multiple TP resources in an OSI environment. An IBM strategy for SNA/OSI transaction integration could, for example, incorporate Remote Database Access (RDA, ISO 9576), Remote Operation Service Element (ROSE, ISO 9072; the basis of RDA), ACSE (ISO 8649/8650), and Commitment, Concurrency and Recovery (CCR, ISO 9804/9805).

Shared SNA/OSI Links and Addresses

One immediate and significant result of achieving SNA and OSI integration through the above dual-stack approach would be that both architectures could share the same physical adapters, links, and subnetworks. This sharing could be accomplished by assigning (from an OSI perspective) each architectural protocol stack a unique link service access point (LSAP) and network service access point (NSAP) address. The resulting network architecture independence would also lend itself quite well to protocol-stack–independent network management.

Three Protocol Boundaries

One interesting result of this approach would be the generation of a few selected protocol boundaries. These would probably be located at the API, transport service, and link/subnetwork access levels. For example, an application program could issue architecture-neutral interprogram calls across an API such as CPI-C. The selected underlying network service (SNA, OSI, TCP/IP) would be kept transparent to the application. From this first selection, a transport service branch would be chosen that, for example, could include OSI transport and network layer protocols, SNA transmission control and path control protocols, NetBIOS, or TCP/IP. From each of these in turn, multiple possible link/subnetwork access branches could be selected, including SDLC, X.25, ISDN, frame relay, FDDI, Ethernet, token ring, or serial optical channel.

Significance of IBM OSI Products

The IBM OSI product line today is significant for several reasons:

- "Standard" standards
- Platform pervasiveness
- A single network infrastructure
- Common LAN interconnections
- Resistance to obsolescence

"Standard" Standards

IBM has selected a suite of OSI services and protocols that is consistent with the most significant international standards and national and regional profiles. The company is willing to submit implementing products to conformance testing to ensure compliance. Users would thus have the best chance of acquiring compliant products that are interoperable with implementations from other vendors.

Platform Pervasiveness

IBM OSI product development was, for several years, focused exclusively on MVS/VM host environments. The introduction of SAA/OSI in 1988 and subsequent SAA and AIX implementations have been a welcome departure from that narrow focus. SAA and AIX are positioned to provide processor-architecture–independent application access.

Single Network Infrastructure

Many enterprises have developed and must maintain multiple separate SNA, TCP/IP, and other vendors' networked environments. These multiple dissimilar solutions are costly and contribute to unacceptable performance. Emerging products will gradually eliminate dual SNA-OSI and TCP/IP-OSI networks through common APIs, bidirectional application gateways, shared connections, and shared adapters.

OSI-TCP/IP Gateways. AIX OSIMF/6000 provides a good example of a set of bidirectional OSI-TCP/IP application gateways. The MHS/Interpersonal Messaging to Simple Mail Transfer Protocol (SMTP) function translates messages between OSI X.400 MHS and AIX TCP/IP nodes. It also includes a set of FTAM to File Transfer Protocol (FTP) bidirectional translation services for OSI and TCP/IP network node file interchange. These TCP/IP-OSI application gateway services are significant in that they underscore IBM's intent to support TCP/IP-to-OSI communication as well as SNA-TCP/IP and SNA-OSI. They are further significant in that *SNA Perspective* believes they are the first of several TCP/IP-to-OSI interface approaches IBM will adopt across its platforms to position for large anticipated industry-wide need in the mid-1990s.

Common LAN Interconnections

As described above under "Three Protocol Boundaries," IBM's OSI software is designed to run over multiple link types, including the primary LAN technologies: token ring, Ethernet, and FDDI. Further, the 6611 multiprotocol router is well positioned to interconnect heterogeneous LANs across WANs, including SNA, TCP/IP, and OSI.

Resistance to Obsolescence

OSI products should address end-user requirements for multivendor integration through a consistent set of interfaces and profiles. OSI standards will be enhanced over time to express evolving user requirements. Therefore, IBM OSI products should support functional modularity without creating obsolescence. IBM's commitment to worldwide profiles, active participation in the standards development process, and strategic focus for SNA, OSI, and TCP/IP interconnection are likely to translate into resistance to obsolescence in its products.

IBM OSI Directions

SNA Perspective believes that IBM's OSI directions should include:

- A common API/CPI approach
- Common API/CPI languages
- Consistent interfaces and interoperability

Common API/CPI Approach

IBM could consider development of common SNA/OSI interfaces to satisfy a wide range of application requirements. One or a few common interface approaches would be useful if it or they could support a variety of applications including transaction processing, electronic messaging, file processing, database processing, terminal/printer applications, distributed processing, graphical design, object interchange, systems/network management, and job processing.

(continued on page 24)

Architect's Corner

Distributed Database: Fact Or Fiction?

by Thomas J. Routt

Some manufacturers have claimed for years to have a distributed relational database management system. Such systems are said to coherently integrate an enterprise's loosely knit repositories of information across dissimilar platforms, access/storage methods, and geographic locations. Today, this is more fiction than fact.

A Truly Distributed Database

Data are increasingly distributed—critical data components are scattered throughout the enterprise and around the world. Vast quantities of enterprise data are often tightly bound to their originating applications. And yet, distributed applications support a complex mix of data types, access techniques, storage techniques, and database or file models.

A truly distributed database must provide a *distributed request* function. Distributed request lets a user issue requests with the appearance that the application and data reside locally, though they may, in fact, be a series of database segments around the company. Distributed request capability is more important as applications are increasingly scattered among database servers and requesters on interconnected LANs.

There are many obstacles to creating distributed request. Associated issues to be conquered include distributed two-phase commit, security, data integrity, and reasonable throughput. However, the potential returns are enormous. Imagine the productivity benefits if users could locally issue standard requests to collect and connect data regardless of location(s) or processing platform.

IBM's Commitment

Distributed relational data is a top priority for IBM. Although the company has not yet delivered distributed request relational capabilities, no one else has either. I see the breadth of IBM's efforts to date in this direction as quite good. IBM is providing a phased rollout of increasingly distributed relational products across its platforms.

SAA Distributed Data

Data access has always been straightforward if the data were close at hand. Access to remote data is more complex, especially if they reside on a variety of servers that are geographically separated. Most enterprises still resort, therefore, to manual procedures to access data.

IBM seeks to remedy this situation through its SAA distributed relational database direction and, more recently, through the Information Warehouse framework which incorporates SAA relational database. Information Warehouse, unveiled in late 1991, is a set of database management systems (DBMSs), interfaces, tools, and facilities to manage and deliver reliable, timely, accurate, and understandable data to appropriate users.

The SAA relational database has four levels of increasing sophistication: remote request, remote unit of work (RUOW), distributed unit of work (DUOW), and distributed request (see Figure 10 on page 22).

Let's consider IBM's products and efforts at each of these levels and then look at some of the next steps.

Remote Request

IBM's Enhanced Connectivity Facilities Server-Requester Programming Interface (ECF SRPI) introduced remote request. ECF SRPI is a cooperative processing application pair in which the server resides in a host and the requester runs in a workstation. Requests are Structured Query Language (SQL) calls. A remote request consists of a single SQL statement. The workstation application sends the source SQL statement within SRPI verbs to the remote DBMS, such as MVS DB2 and VM SQL/Data System (SQL/DS), where optimization and access occur. ECF SRPI uses SNA LU 2/3270 sessions to convey requests and services.

Remote Unit of Work

RUOW was introduced by IBM as the Remote Relational Access Support feature of VM SQL/DS. An application in this environment (in conjunction with the SQL/DS database switching function) may search a local SQL/DS DBMS for a specified item, such as a part number in an inventory database. If the desired data item isn't local, the application can switch to the SQL/DS DBMS of the remote application, such as inventory, where the requested item is located. The data can then be allocated and dispatched and the remote database updated accordingly.

The Information Warehouse framework consists of an enterprise data element, a data delivery element, and applications and decision support systems. At present, the data delivery element uses the RUOW function as defined in the SAA Distributed Relational Database Architecture (DRDA). DRDA uses SQL as the application interface to distributed relational data.

I consider it positive that Information Warehouse includes RUOW across all SAA platforms: MVS DB2 V2R3, VM SQL/DS V3R3, OS/400 Database Manager V2R1.1, and on the OS/2 Database Manager as an RUOW client.

Distributed Unit of Work

IBM introduced DUOW with DB2 V2R2. Here, an IMS or CICS application can, in a unit of work, read any number of local or remote DB2 databases and write to the local DB2 database. A TSO/E application in the DB2 V2R2 environment can, in a single unit of work, read any number of local or remote DB2 databases and write to a single local or remote DB2 database. Transactions are based on underlying LU 6.2 sessions.

Distributed Request

Distributed request removes all data location restrictions inherent in the above three. Distributed request creates a distributed database environment which logically appears to be one large, local DBMS. IBM has not introduced distributed request functionality to date.

Directions and Implications

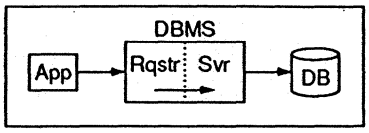
Based both on user needs and on IBM's statements of direction and announcements, I expect the following database directions for IBM:

- DUOW on all strategic platforms, with data integrity further ensured through consistent use of CPI-C. This would at once standardize calls to the underlying LU 6.2 session platform and provide a predictable, distributed basis for APPN connectivity.
- Distributed request on all strategic platforms. This most challenging piece of work represents the actual starting point of distributed relational DBMS function. The major obstacles: handling distributed two-phase commit, providing reasonable levels of security, ensuring data integrity across multiple, distributed platforms, and supporting reasonable throughput and response time profiles, especially for distributed transactions. While these are clearly enormous challenges, I expect IBM to accomplish this goal within three years in all SAA and AIX processing environments. When it has done so, we can say that IBM has reached its distributed relational database goal in fact.
- Object Database Management System (ODBMS) on all strategic platforms. SAA Common User Access '91 (CUA '91) provides object-based user interface enhancements to CUA '89. This suggests to me that SAA distributed data strategy will include development of an ODBMS, which would treat objects as collections of related procedures and data. In this approach, all relational DBMS services are preserved without the overhead incurred in disassembling and reassembling objects each time they are stored or retrieved. That is, distributed data would be stored as composite objects with component links represented as direct references. ODBMS would be a natural counterpart to CUA '91 as well as to IBM's participation in ongoing work with organizations committed to industry openness such as OSF. ■

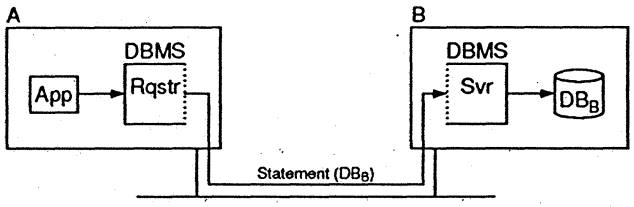
Note from Dr. John R. Pickens: This month, SNA Perspective brings you a guest architect. Thomas J. Routt is a consultant on SNA, SAA, network management, OSI, and TCP/IP and is a frequent contributor to SNA Perspective. Before becoming a consultant, he was the manager of Boeing Network Architecture.

Levels of Database Sophistication

Local Request

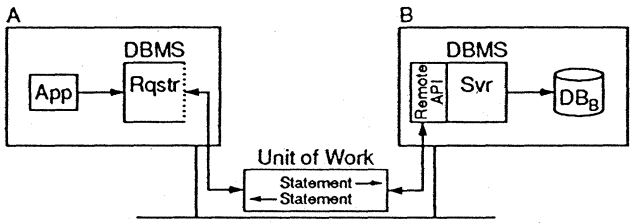


Remote Request



- A server DBMS is unaware that a distribution is occurring.
- No communication occurs while any DBMS resources are held.
- The requesting user knows which end system contains the desired data.
- All distributed data are copies of the original.
- Each statement is independent: failure of one request affects the results of only that request.

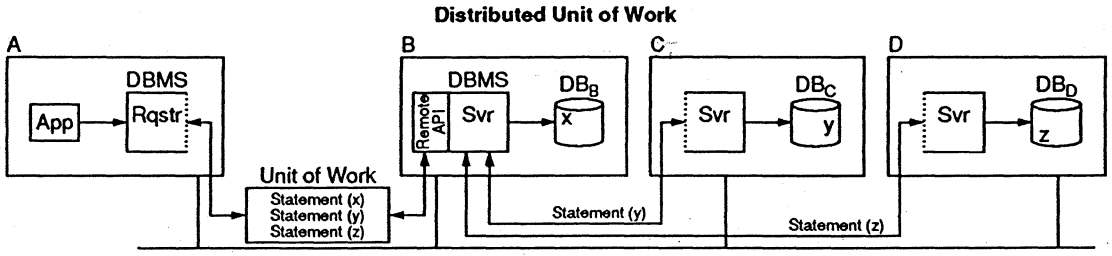
Remote Unit of Work



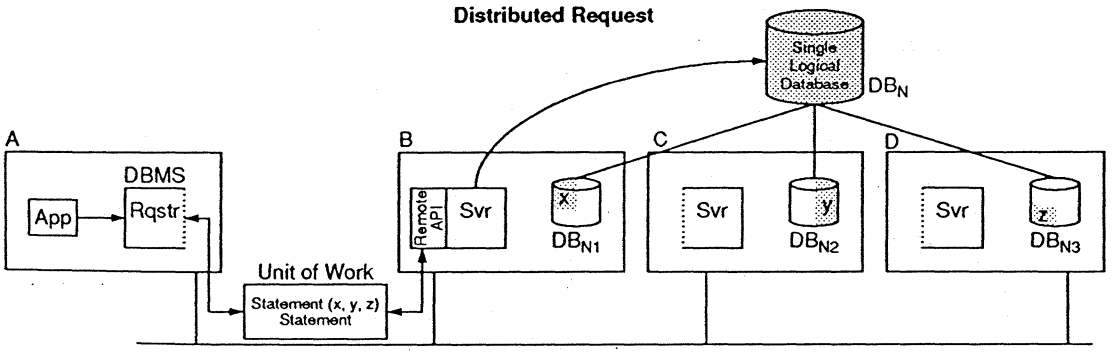
- A locally-executing application uses remote API provided by the remote system.
- Any DBMS facilities available to local applications are available to remote applications.
- A remote DBMS processes statements related by a unit of work (e.g., an SQL transaction).
- A remote DBMS updates database only when all unit-of-work statements are successfully completed.
- During application execution, database resources are protected with locks.
- Applications control boundaries of an active unit of work through COMMIT and ROLLBACK requests.

Figure 10

Levels of Database Sophistication (continued)



- The DBMS is aware which system manages the data to be read/modified by each request.
- The DBMS coordinates updates at several locations within a single transaction.
- DUOW provides access to multiple relational DBMSs within a single unit of work.
- Related updates at multiple locations can be performed as part of a single distributed transaction.
- No partial updates allowed: either all or none of the related changes are committed into DBMSs.
- DUOW requires a two-phase commit.
- Within a transaction, all operands of a single statement must refer to data from a single DBMS.



- Distributed request removes all data location restrictions inherent in the other levels.
- A single statement can be used to combine relational data from multiple DBMS locations.
- Distributed request creates distributed database environment which logically appears to be one large local DBMS.

Figure 10 (continued)

(continued from page 19)

API/CPI Languages

APIs and SAA CPIs enable users to write programs that communicate through OSI and are consistent across platforms. At present in SAA, C and COBOL can be used to write to APIs for certain OSI application, presentation, session, and transport layer services. It is likely that, in the near future, support will be provided for OSI APIs written in all SAA programming languages, which presently include C, COBOL, FORTRAN, PL/I, RPG, and REXX. Natural language APIs are also likely.

The key issue for any API that IBM may develop is to ensure that the requesting application program be concerned only with what it wants to accomplish and not with the underlying details of how to accomplish the task.

Consistent Interfaces and Interoperability

Users require common application and network interface solutions for SNA, OSI, and TCP/IP.

Common interfaces will assist in enterprise migration to a single infrastructure. Single-infrastructure networks will support transparent end-user interfaces, reduction of redundant resources, simplification of system/network management, improved performance, and improved security.

IBM needs to ensure consistent interfaces and consistent interoperability. Consistent application and network interfaces should be provided in all OSI products across SAA and AIX platforms. Provision must be made for OSI, SNA, and TCP/IP interoperability in all SAA and AIX platforms consistently and independent of product life cycles.

by Thomas J. Routt ■