# The Library for Systems Solutions Computing Technology Reference

Document Number GG24-4100-00

August 1994

International Technical Support Organisation
Poughkeepsie Center

**Take Note!**

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xiii.

**First Edition (August 1994)**

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. H52 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

# Abstract

This document is part of The Library for Systems Solutions, which is intended for technical professionals involved in defining solutions to data processing problems, in multiple configuration environments and multiple software platforms, including heterogeneous distributed environments.

Several hardware, software, and architecture developments are analyzed to establish a common base of understanding for concepts, such as software platform, distributed system structure, open environment, and a distributed systems environment. The most common IBM software platforms are also described on the basis of those concepts.

(242 pages)

# Contents

# Figures

# Tables

# Special Notices

This publication is part of the Library for Systems Solutions which is intended for technical professionals involved in defining solutions to data processing problems in multiple configuration environments and multiple software platforms. In particular, this publication contains information about both traditional and more recent developments in computing technology, and also contains information that is preparatory to the understanding and use of the other publications in the Library. When a product is mentioned, the information in this publication is not intended as the specification of any programming interface that is provided by that product. See the PUBLICATIONS section of the IBM Programming Announcement for each named product for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program, or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbour Drive, Stamford, CT 06904 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (vendor) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ACF/VTAM
Advanced Function Printing
AIX
AIX/6000
AD/Cycle
Advanced Peer-to-Peer Networking
AIX/ESA
AIXwindows

AOEXPERT/MVS
APPN
AT
CallPath CICS/MVS
CICS
CICS/ESA
CICS/VM
CICS/400
COBOL/2
COBOL/400
CUA
DATABASE 2
Dataspace
DB2
DB2/6000
DFSMS
DFSMS/VM
DFSMSdss
DFSMSrmm

Distributed Relational Database Architecture
Enterprise Systems Architecture/370
Enterprise System/3090
Enterprise System/9000
Enterprise Systems Connection Architecture
ES/4381
ES/9370
ESA/390
ESCON XDF
FORTRAN/400
Hiperspace
IMS CS/2
Micro Channel
MVS
MVS/ESA
MVS/XA
NQS/MVS
OfficeVision/MVS
OfficeVision/2
OPC/ESA
Operating System/400
OS/400
Personal Computer AT
Personal System/2
PowerPC
PR/SM
Processor Resource/Systems Manager
PS/2
RACF
RISC System/6000
RT Personal Computer
S/360
S/370-XA
SAA
SQL
SQL/400
Sysplex Timer
System/370
System/370-XA
Systems Application Architecture

Application System/400
AS/400
CallPath
CallPath CICS/VSE
CICS OS/2
CICS/MVS
CICS/VSE
CICS/6000
COBOL/370
Common User Access
Data Propagator
DataHub
DataTrade
DB2/2
DFDSM
DFSMS/MVS
DFSMSdfp
DFSMShsm
DISTRIBUTED DATABASE CONNECTION SERVICES/2
DRDA

Enterprise Systems Architecture/390
Enterprise System/4381
Enterprise System/9370
ES/3090

ES/9000
ESA/370
ESCON
FORTRAN/2
Hardware Configuration Definition
IMS Client Server/2
IMS/ESA
Multimedia Presentation Manager/2
MVS/DFP
MVS/SP
NetView
OfficeVision
OfficeVision/VM
OfficeVision/400
Operating System/2
OS/2
Person to Person/2
Personal Computer XT
POWER Architecture
PowerPC Architecture
Presentation Manager
PROFS
QMF
RISC
RT PC
RT
S/370
S/390
SNA
SQL/DS
SYSPLEX
System/360
System/370-Extended Architecture
System/390
SystemView

| | |
|---|---|
| ThinkPad | Virtual Machine/Enterprise Systems Architecture |
| Virtual Machine/Extended Architecture | VM/ESA |
| VM/XA | VSE/ESA |
| VTAM | Workplace Shell |
| 3090 | |

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

| | |
|---|---|
| Apple | Apple Computer Inc. |
| AT&T | American Telephone and Telegraph Corporation |
| Best/1 | BGS Systems Inc. |
| BSD | Berkeley University, California |
| Berkeley Software Distribution | Berkeley University, California |
| cc:Mail | Lotus Development Corporation |
| DCE | Open Software Foundation, Inc. |
| DEC, DECnet | Digital Equipment Corporation |
| DEC VT52, DEC VT100 and DEC VT220 | Digital Equipment Corporation |
| EASEL | Interactive Images, Inc. |
| EASEL Workbench | Interactive Images, Inc. |
| EDA/SQL | Information Builders, Inc. |
| Encina | Transarc Corporation |
| Ethernet | Xerox Corporation |
| Fujitsu | Fujitsu Limited |
| Hewlett-Packard | Hewlett-Packard Company |
| Hitachi | Hitachi Ltd. |
| HP | Hewlett-Packard Company |
| HPUX | Hewlett-Packard Company |
| IDMS | Computer Associates, Inc. |
| IEEE | Institute of Electrical and Electronics Engineers |
| INFORMIX | Informix Software Inc. |
| Ingres | Ingres Corporation |
| Intel | Intel Corporation |
| Intel386 | Intel Corporation |
| Intel486 | Intel Corporation |
| Kerberos | Massachussets Institute of Technology |
| Lotus,Lotus 1-2-3 | Lotus Development Corporation |
| Lotus Notes | Lotus Development Corporation |
| Intel486 | Intel Corporation |
| Macintosh | Apple Computer Inc. |
| Microsoft, Windows | Microsoft Corporation |
| Motif | Open Software Foundation, Inc. |
| Motorola | Motorola, Inc. |
| MS, MS-DOS | Microsoft Corporation |
| NCS | Hewlett-Packard Company, Apollo Systems Division |
| NEC | NEC Corporation |
| Network File System, NFS | Sun Microsystems Inc. |
| NetWare | Novell, Inc. |
| Novell | Novell, Inc. |
| Omegamon | Candle Corporation |
| Open Software Foundation | Open Software Foundation, Inc. |
| Oracle | Oracle Corp. |
| OSF | Open Software Foundation, Inc. |
| OSF/1 | Open Software Foundation, Inc. |
| Pentium | Intel Corporation. |
| POSIX | IEEE |
| SCO | Santa Cruz Operation, Inc. |
| Sun Microsystem | Sun Microsystems, Inc. |

| | |
|---|---|
| Sybase | Sybase Inc. |
| Synon | Synon Corporation |
| Texas Instruments | Texas Instruments Inc. |
| UI-Atlas | UNIX International |
| ULTRIX | Digital Equipment Corporation |
| UniTree | General Atomics |
| UNIX | UNIX System Laboratories, Inc. |
| UTS | Amdahl Corporation |
| VAX | Digital Equipment Corporation |
| VMS | Digital Equipment Corporation |
| Windows | Microsoft Corporation |
| XPG3 | X/Open Company Limited. |
| XPG4 | X/Open Company Limited. |
| Xenix | Microsoft Corporation |
| X-Open | X/Open Company Limited |
| X-Windows | Massachusetts Institute of Technology |
| X-Window System | Massachusetts Institute of Technology |

# Preface

This document is part of The Library for Systems Solutions. It contains technical information about both traditional and more recent developments in computing technology. In addition, this document also contains information that is preparatory to the understanding and use of the Library.

## The Library for Systems Solutions

*The Library for Systems Solutions* is intended for technical professionals involved in understanding and selecting the components that provide solutions to data processing problems. Because the solutions may vary with the actual configuration, multiple configuration scenarios or, simply, environments, are considered in the Library. Environments range from the totally centralized to distributed environments with several levels of distribution. When distributed environments are considered, the involved systems are assumed to be homogeneous or heterogeneous as far as the hardware, the architecture, and the software are concerned.

There are various types of books in the Library: a *Computing Technology Reference* book, *Function Reference* books, and *Technology Reference* books.

The *Computing Technology Reference* contains technical information about both traditional and more recent developments in computing technology. Several hardware, software, and architecture developments are analyzed to establish a common base of understanding for concepts, such as a software platform, a distributed system structure, an open environment, and a distributed systems environment. The most common IBM software platforms are also described on the basis of those concepts. In addition, the book contains information that is required for understanding and using the library.

The *Function Reference* books contain the solutions available, at this date, in various areas of data processing and business related functions. The areas, or functions, defined at the time of the initial publication of the *Computing Technology Reference* and covered by the individual books are:

- Application Development
- Workload Management
- Data Access
- Security
- Directory, Naming, and Timing
- Systems Management
- Printing and Viewing
- Image
- Office

For the convenience of the reader, each function reference book has a complimentary structure and is organized in two sections, with an additional summary section when needed and appropriate.

Section 1 is a description of the current state of technology for that function. It contains, at a high level of technical detail, all the information that is required to approach the task of building a solution for that function in various environments.

Section 2 describes the solutions, or solution approaches, available for the function with current technologies and products. To address the problem of having to deal, in real life, with multiple environments and multiple software platforms per environment, this section is organized into several chapters, each of which addresses a specific environment.

Five environments are defined in the *Computing Technology Reference* book, and it is expected that they will cover most of the environments that will be encountered in real life situations. In general these environments are used in other function books to provide the reader with a consistent reference framework. If required, additional environments of special interest for a specific function will also be addressed in a separate chapter called, “Other Environments.”

Each chapter addressing a specific environment describes the solutions currently available in that environment for the different software platforms that might be involved.

The *Technology Reference* books cover technologies that are an important component of any solution in any environment. There are currently three technology reference books in the Library covering the End-user Interface, Open Networking, and LAN technologies.

The structure of the Library for Systems Solutions allows it to be used in several ways by technical professionals with different types of skill.

A reader looking for a solution in a specific, familiar, function area is expected to go directly to Section 2 of the appropriate function reference book to find the products implementing the requested solution in the applicable environment and for the applicable software platforms.

Users looking for a solution in an unfamiliar function area are expected to go to the applicable function reference book and use Section 1 first to get acquainted with the technology aspects of that function, and then to go to Section 2 of the book to look for the solution.

The user that requires information on current computing technologies prior to looking for a solution in a specific environment is expected to use the Library by reading the Computing Technology Reference book and the Technology Reference books, before going to the specific Function Reference books.

The objective of the Library for Systems Solutions is to identify the elements that provide solutions to specific data processing problems in several environments.

It is not the objective of the Library to cover two additional activities that are also relevant to the process of building a solution. These activities are the *design* of the solution, starting from the business requirements of the customer, and the *implementation* phase of the chosen solution.

The Library for Systems Solutions is expected to be extended and to be updated periodically as products and technologies evolve.

## How This Document is Organized

The document is organized as follows:

- Chapter 1, “Introduction,” summarizes the elements of technology and business evolution that have produced substantial changes to the traditional paradigms of information systems in the last few years.
- Chapter 2, “Data Processing Technology,” describes several developments that have characterized the technology evolution in the last few years and are prerequisite knowledge to define solutions to data processing problems in today′s environments.
- Chapter 3, “Distributed Data Processing Technology,” describes how the structure of software is evolving to meet the requirements of a distributed data processing system based on multiple hardware and software platforms (heterogeneous systems).

  At the time the book was initially developed the Open Blueprint had not yet been completed. As this work evolved it became clear that there was significant value in utilising Open Blueprint material in a complementary way. To this end the original chapter is now replaced by substantial elements from the Open Blueprint Technical Overview which directly meets the author′s original objectives for the chapter.

  The chapter concludes with a descriptive approach, derived from the Open Blueprint, identifying the elements that provide solutions to specific data processing problems in several environments. This descriptive framework is generally used as a model throughout the Library.
- Chapter 4, “Configuration Environments,” describes several types of data processing configurations, or environments, starting from environments with no or minimal distribution up to environments with multiple levels of distribution. Those environments will generally be referred to and used by documents in the Library for Systems Solutions to help the user locate the solution to a specific problem.
- Chapter 5, “Software Environments,” describes the most common types of software environments available in the industry today, starting with a traditional mainframe and concluding with a desktop processor. For each environment, the qualifying software elements are described. A list of items is also provided that deserve special consideration when a data processing environment is being expanded to assume the characteristics of a heterogeneous distributed environment.
- Chapter 6, “IBM Software Platforms,” maps IBM software platforms against a common structure of software services ranging from support of the processor architecture to the support of applications and systems management. Based on the level of service provided, the various software components are classified as local or distributed.

## Related Publications

In addition to the publication you are now reading, *The Library for Systems Solutions* contains the following publications:

- *Application Development Reference,* GG24-4101
- *Workload Management Reference,* GG24-4102
- *Data Reference,* GG24-4103
- *Directory, Naming, and Timing Reference,* GG24-4104
- *Printing and Viewing Reference,* GG24-4105
- *Security Reference,* GG24-4106
- *End User Interface Reference,* GG24-4107
- *Image Processing Reference,* GG24-4109
- *Open Networking Reference,* GG24-4110
- *LAN Reference,* GG24-4111
- *Office Reference,* GG24-4112
- *Systems Management References,* GG24-4113 through GG24-4117

The following publications were referenced and provided input during the writing of this publication. Some of these publications may no longer be available.

- H. Lorin, *System Architecture In Transition. An Overview* IBM Systems Journal, Vol.25, Nos.3/4, 1986, G321-0084.
- *IBM Journal of Research and Development, Vol.27, N.3, 1983*, G322-0129.
- *IBM Journal of Research and Development, Vol.36, N.4, 1992*, G322-0181.
- *IBM Journal of Research and Development, Vol.34, N.1, 1990*, G322-0169.
- *IBM System Journal, Vol.27, N.3, 1988*, G321-0091.
- *IBM System Journal, Vol.28, N.1, 1989*, G321-0093.
- *IBM System Journal, Vol.28, N.3, 1989*, G321-0095.
- *IBM System Journal, Vol.32, N.4, 1993*, G321-0114.
- *The Networking Blueprint*, SX33-6090.
- *Client/Server Computing: The Design and Coding of a Business Application*, GG24-3899.
- *Client/Server Computing Application Design Guidelines: A Distributed Relational Data Perspective* , GG24-3737.
- *Client/Server Computing Application Design Guidelines: A Transaction Processing Perspective*, GG24-3728.
- *Basic Electronic Mail*, GG24-3878.
- *SAA SystemView Concepts*, SC23-0578.
- *Open Blueprint Introduction*, G326-0395
- *Open Blueprint Technical Overview*, GC23-3808

## International Technical Support Organization Publications

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

*Bibliography of International Technical Support Centers Technical Bulletins,* GG24-3070.

To get listings of ITSO technical bulletins (redbooks) online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

**How to Order Redbooks**

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy redbooks individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order redbooks in online format on CD-ROM collections, which contain the redbooks for multiple products.

## Acknowledgments

This publication is the result of a residency conducted at the International Technical Support Organization, Poughkeepsie Center.

The coordinator of this document is:

Fulvio Capogrosso, IBM Italy

The authors of this document are:

Chuck Poland, International Technical Support Organization, Poughkeepsie Center

Franco Meroni, IBM Italy

William Ogden

Special thanks are also due to:

- Deborah Carr
- George Hutfilz
- Lou Thomason
- Jim Colosimo

of IBM Systems Software Structure for their contribution in providing concepts, documentation, and review for this document.

In particular thanks are due to Matt Schein of the Distribution Systems Architecture group for allowing the substantial use of his Open Blueprint Technical Overview to replace the original material in chapter 3.

Thanks are also due to the many members of the International Technical Support Organization who reviewed and improved this document. In particular, thanks are due to Adam Jollans, ITSO Boca Raton, for improvements made to the Personal Computer elements of the book.

Additional coordination was provided by

Ian Crane, International Technical Support Organization, Poughkeepsie Center

# Chapter 1. Introduction

The evolution of data processing technology and the changes in business organizations and disciplines have both produced substantial changes to the traditional paradigms of information systems (IS).

Data processing technology has evolved in such a way and in so many directions that many processor types, architectures, and software offerings are commercially available today through multiple vendors. Moreover, the level of performance and the price/performance available with new technologies are continuously improving and opening areas to data processing that were not conceivable or affordable in the past.

To quantify the scale of the evolution, recall that in the 60′s, the number of data processing systems functioning worldwide was approximately in the hundreds and mostly of the type that is today called the traditional commercial mainframe. There were a handful of vendors. Processing power, data, and applications were all concentrated in only one place.

To make those resources available to geographically dispersed users, terminals and communication lines were used. Terminals had no processing power (non-intelligent) but only input/output capability through a keyboard and a display. Accommodation in only one processor of several types of workloads with different characteristics such as commercial and scientific batch, online transactions, and interactive users required the processor architecture to be general purpose and the control program to be increasingly sophisticated to satisfy the requirement for optimized performance. That type of data processing organization was essentially dictated by the cost of the hardware (compared to the cost of communication lines) and by the business organizations which, being mostly hierarchical, related well to a centralized data processing structure.

Today, the number of commercial mainframes installed worldwide is in the tens of thousands. In addition, the number of intermediate and departmental processors is in the order of the hundreds of thousands, and personal computers and workstations range in the order of the tens of millions. There are literally thousands of hardware and software vendors. These numbers clearly illustrate the choices available to data processing organizations today. In addition to centralized configurations, where the processing power, data, and applications are concentrated in only one place and on one system, it is now possible to have configurations where the processing power, data, and applications are distributed among several interconnected locations (or nodes) and can still operate as a single data processing system.

A system of this type is called a distributed system. A further qualification of heterogeneous, as opposed to homogeneous, might also be added when the individual systems have different characteristics as far as the hardware, the software, or the architecture are concerned.

Distributed physical resources do not necessarily mean distributed management and distributed organization. Various types of logical organizations can be implemented for a distributed system to suit different business requirements. Examples range from a mostly centralized organization where a system with full control over data and applications is connected to a periphery of workstations or personal computers dedicated to individual activities with minimal interaction

with the center, to more de-centralized and complex organizations where multiple data processing locations, each independently managed, share data, applications, and processes.

Together with data processing technologies, business organizations and management philosophies have also changed, and management models addressing needs beyond the traditional hierarchical and centralized structures have emerged. Distribution of management responsibilities may derive from expansion of a business beyond the point where centralized management can still maintain the required level of efficiency and control. Distribution of responsibilities often generates the requirement to distribute data processing resources. Similarly, common business practices, such as intercompany partnerships, mergers, and acquisitions, also generate the requirement to integrate multiple, independent data processing organizations into a single distributed data processing system.

Common business drivers to data processing distribution are:

- The requirement for alternate locations for backup and security purposes.
- The requirement to increase high availability for some applications by physically isolating them from other applications and still maintaining some level of logical connection.
- The requirement to improve response times for applications with remote users that are geographically dispersed. In this case, centralized data can be a performance bottleneck due to communication delay. The distribution of data and processing power can remove the constraint.

The evolution of data processing technology and the increasing number of products that are available every day to address new application areas represent a challenge for the technical professionals whose responsibility is to provide solutions to data processing problems. If the environment is also distributed and the processors in the environment are heterogeneous as far as the architecture and the software are concerned, it adds an additional level of complexity to the problem.

This book and its associated volumes in the Library for Systems Solutions is intended to help technical professionals face these challenges in translating the increasing number of technology products into solutions for multiple heterogeneous environments.

**Note:** In this document, the following terms are used to indicate quantities:

K=kilo=$2^{10}$=1,024
KB=kilobytes=$2^{10}$bytes=1,024 bytes
MB=megabytes=$2^{20}$bytes=1,048,576 bytes
GB=gigabytes=$2^{30}$bytes=1,073,741,824 bytes
TB=terabytes=$2^{40}$bytes=1,099,511,627,776 bytes
PB=petabytes=$2^{50}$bytes=1,125,899,906,842,624 bytes

# Chapter 2. Data Processing Technology

A data processing system is described, in general terms, by its main components. They are users, applications, system software or simply software, architecture, hardware, and the relationships among these components. An important type of relationship between the components is the way they communicate among themselves or, in other words, the type of *interfaces* that they have.

For example, the interfaces used by applications to communicate with software are generally qualified as application programming interfaces (API). The interface to communicate with the hardware is generally called the machine-level interface, or machine interface, and the interface used by the user to access the system is generally referred to as the end-user interface (EUI).

This high-level description of a data processing system is represented in Figure 1 and will be used throughout this document as a model for a data processing system. When no ambiguity exists, the term *system* will also be used in place of data processing system to represent any hardware equipment capable of executing applications and system software.

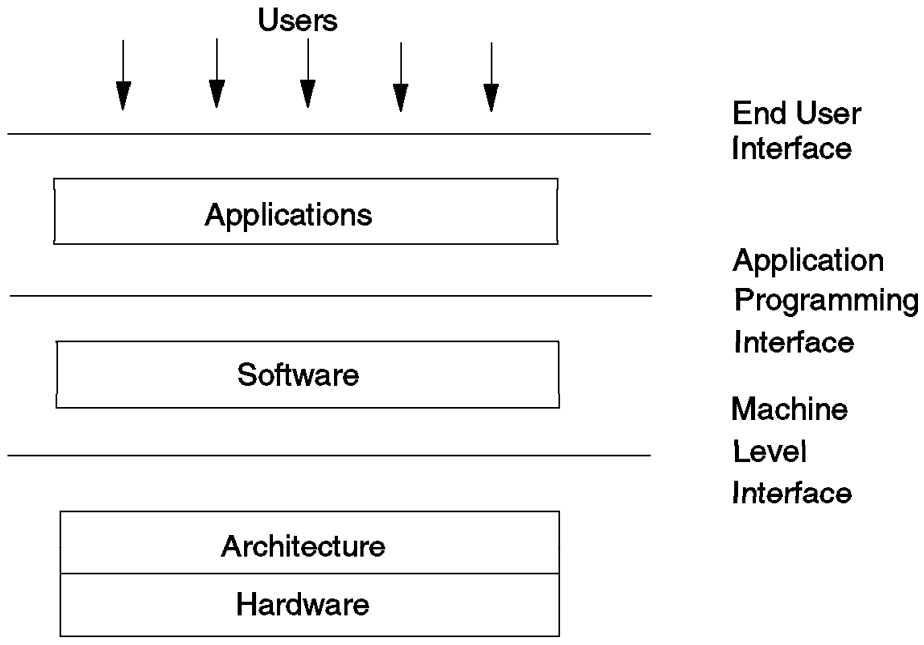


*Figure 1. System Model*

## 2.1 Data Processing Elements

Figure 1 on page 3, though quite general, provides most of the elements that will be used to describe various types of systems available in the industry today.

*Architecture* is the name traditionally used to describe that set of rules, principles, and behaviors that a programmer or a code generator has to know and follow to utilize the hardware components of a computing system. Therefore the architecture includes the instruction set, the storage addressing scheme, the interruption structure and priorities, the communication interfaces among the system components, and the communication interfaces between the hardware and the software.

For a long time, the word *architecture*, with no further qualification, has been closely associated with the processing unit where the instructions are executed. Because no ambiguity existed, the term was used in place of the more precise *hardware architecture*. Today, due to the evolution of the design of data processing systems, the concept has been extended to other elements of the system.

Terms like *I/O architecture* and *communications architecture*, up to the more general *system architecture*, have become common. This extension of the concept has not altered the original meaning of the term *architecture*, even though sometimes it is also used to include elements of hardware and software design that, according to a strict definition of the word architecture, should be kept separate.

The *hardware* represents the physical resources available to the system.

Hardware includes the central processing units for the execution of the programming instructions; the input, output and storage devices for the management of data; and the communication lines and controls for the management of remote units. The structure, components, and internal organization of the hardware are the result of a design effort aimed to map an architecture into an available technology in order to achieve the price/performance targets established for that data processing system. Elements of design are, for example, the internal structure of each processing unit (sequential, overlapped, pipeline, parallel, and so on), the arrangements of the processing units (uniprocessors, dual, n-way, multiprocessor), the structure of the other elements such as internal buses, storage hierarchy and I/O controllers. The design determines whether a microcode layer exists between the hardware and the software. In Figure 1 on page 3, a microcode layer, if present, is considered to be part of the hardware layer.

The *software*, or system software, or operating system as it was called in the early days of data processing, allows applications to execute on the hardware according to the rules defined by the architecture. The software performs this function by accepting requests for services from the applications and honoring them either directly (internally) or by having them serviced by hardware components.

Directly serviced requests include, for example, requests for physical or logical resource allocations. Requests passed to the hardware for service include, for example, requests for access to external storage devices.

To some extent, it might be useful to think of the software as the *interpreter* of requests for services coming from the applications and directed to the hardware and, with a further level of abstraction, as a *shield* to protect the applications from the complex technical implications of having to interface directly with the hardware and architecture.

Based on individual objectives, software can be designed and developed for a single machine level interface and, therefore, be capable of executing on a single hardware platform/architecture. It can also be designed and developed to be portable across multiple hardware platforms and architectures.

MVS/ESA*, OS/400*, and DOS* are examples of the first category, while UNIX** systems and OSF/1** are examples of the second.

Due to the rather complex relationships among software, hardware, and architecture, a data processing system is clearly identified when all three elements are known. For example, the single term S/390* processor identifies an IBM ES/9000* processor, the ESA/390* architecture, and the ESA/390* software components. Similarly, a personal computer is normally identified by the type of processor and the operating system it executes.

The *applications* represent the programs to be executed and managed to achieve the business objectives defined for the data processing system. From a technological point of view, the distinction between applications and software is quite arbitrary because both the applications and the software are executable programs. The reason behind the distinction is to highlight some major differences.

Applications are designed to request services from the software, and the software is designed to provide services to the applications. Application evolution is driven by the business strategy of the customer. Software evolution, however, is driven by information technology. Applications tend to be tailored to the individual business, while the software services tend to become industry commodities.

A characteristic that both the applications and the software have in common is the very fast rate of change they are subject to driven by improvements in business disciplines and the fast pace of technology innovation. This creates the requirement for both applications and software to be easily adaptable to changing requirements.

There is a lively debate in the industry today about the best way to achieve this adaptability, but a consensus appears to be growing about a structure where the role of the applications is limited to that of service *requesters*, and the role of software is defined as that of service *provider*.

It appears, therefore, no longer necessary for the business environment to develop applications where some sort of system services are built in. Those types of applications, also known as Roll-Your-Own (RYO) applications, were popular in the past when the spectrum of system services provided by the software was not as rich as today.

Whenever it is necessary to make a distinction, we will use the term *data processing system*, or *system*, for the hardware, software, and architecture, and use the term *information system* for the hardware, software, architecture, and the applications. This definition implies that the data processing system provides

support to the applications, while the information system provides support information to the user (or to the business).

## 2.2 Data Processing System Evolution

The data processing system model in Figure 1 on page 3 has the structure of the early implementation of commercial computers where the software component was simply called the *operating system* and consisted of limited workload scheduling functions, limited resource management, basic access methods, and few I/O drivers and programming languages. As an example, we might think of the early implementation of IBM S/360* systems.

The model is the common origin of several evolutionary paths that have produced the various data processing systems and technologies available in the industry today. Among others, we will discuss the following technology evolutions:

- Software layers
- S/390* mainframes
- UNIX** systems
- RISC* technology
- Open systems
- AS/400* systems
- Personal systems

The software layers are described first because the structure of software is of primary importance when defining the capabilities and the characteristics of a data processing system.

In addition, when dealing with heterogeneous distributed systems, it is the software that primarily determines the ability of the individual systems to communicate and to interoperate.

Mainframes (using the S/390* mainframe as a reference) are described next from an historical perspective and also because most of the data processing evolutions that follow are better understood by describing how they diverge or what they intend to change relative to mainframe technology.

From this perspective, UNIX** systems are described as an attempt to achieve application independence from the hardware, and RISC* technology is described as an attempt to improve the raw performance achievable with the mainframe processor technology. AS/400* systems are then described as an attempt to optimize the structure of a system to the requirements of midrange application environments, and personal systems are described as an attempt to export traditional business-oriented data processing technologies to everyday private and work activities.

Finally, open systems are discussed as an attempt to solve the issues that so many different types of systems and technologies create when they are combined to form a single, heterogeneous, distributed system.

## 2.3 Software Layers

In section 2.1, “Data Processing Elements” on page 4, the software has been described as an interpreter between applications and the machine-level interface. In fact, software, in addition to providing services to the applications, also allows the applications to be less involved with the technical details of the architecture. Software translates the requests for services into the appropriate sequence of architectural instructions and commands.

The concept of having a layer of software providing services to the applications and at the same time relieving the applications of the technical language of the architecture can be replicated n times using additional software layers between the applications and the operating system. The role of those additional layers, sometimes called *subsystems*, is to provide the applications with services of increasingly higher level, where the term high-level implies greater functional content and better user orientation. Examples of those layers are transaction managers, database managers, workload managers, communication managers, and other service providers.

With the addition of software layers, the structure of a data processing system has evolved over the time from the model represented in Figure 1 on page 3 to the model of Figure 2, where the software component is now represented by a series of overlapping software layers.

As each technology matures, it explores both the relationships between layers and also between elements within the layers. This is evident today in the development of personal computer software.

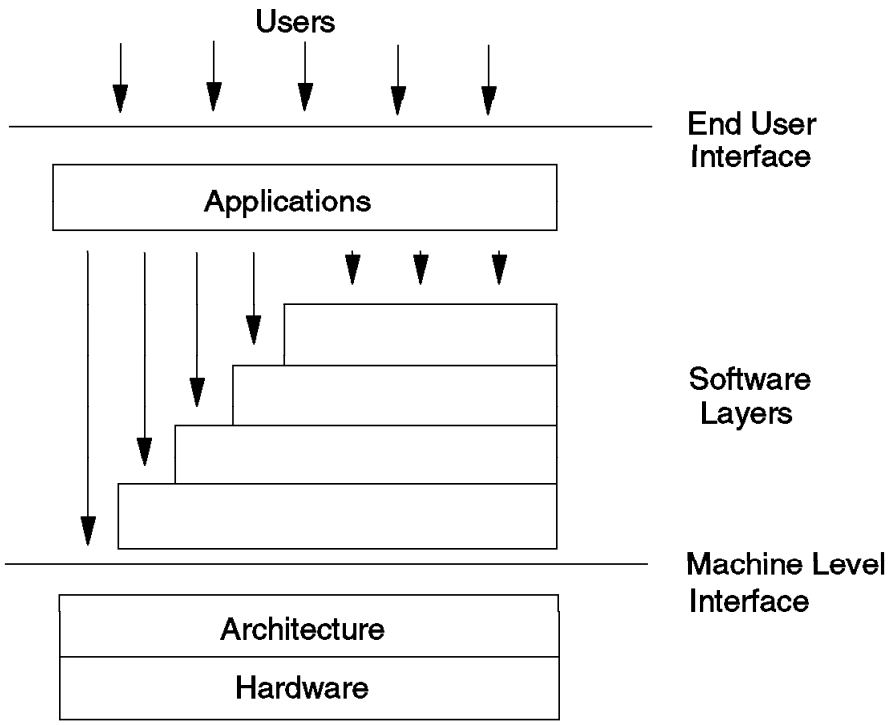


*Figure 2. Software Layers*

The number of layers and the number of independent software products in a layer are variables that do not affect the rest of this discussion, and, for simplicity, we will assume at this point that the terms *software layer* and *software product* are synonymous.

### 2.3.1 Programming Interfaces

An important element for understanding a data processing system with a layered software structure is the mechanism used by the layers to communicate among themselves.

Software elements communicate, horizontally or vertically, through programming interfaces, and each software layer provides an interface to its users and uses the interface(s) provided by the layers it intends to use.

If, for convenience, we consider a vertical structure and we number the software layers progressively starting from the lowest level, we might say that software layer n provides a programming interface to layer n+1 and above, and uses the interfaces provided by layers n-1 and below, as shown in Figure 3.

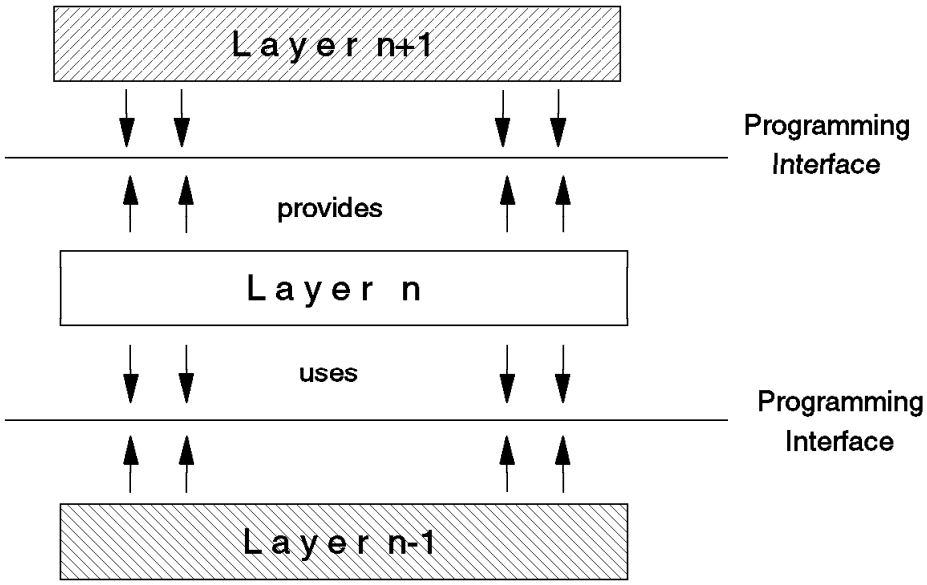


*Figure 3. Communication Between Layers*

To describe the process by which an application or a software layer uses a programming interface, it is common practice to use expressions such as *applications written to an interface*, or *the layers communicate across or through programming interfaces*, with the same meaning.

The interface to a software layer contains all the information required by a program to access the services provided by that software layer. The attributes of the programming interfaces have an important role in determining the usability of a data processing system.

A programming interface may be public, proprietary, or standard. A programming interface is public when it is fully documented by the owner and is provided for unrestricted use. Examples of a public interface include the SQL* language to access a DB2* relational data base, the TCP/IP Socket interface, and the CICS* command level programming interface. A public interface should contain some level of explicit or implicit commitment to maintain the interface across product evolution.

The term proprietary is today used to qualify a public interface developed by a software vendor according to its own private standards. Any application or software component that makes use of a proprietary interface can execute only in data processing environments where the vendor product providing that interface is also available.

If the term *porting* is used to describe the activity of transferring an application from one data processing system to another, an application depending on a proprietary programming interface can be ported only across systems where the product providing that interface is also available.

A data processing system in which the software layers provide and use only proprietary interfaces to communicate among themselves and with the applications is said to be a proprietary system.

An interface is standard when it is defined by a body entitled to define standards. Depending on the charter of the standards body, there are national standards, international standards, industry standards and others. There are also proprietary interfaces that, intentionally or not, have become accepted as standard by a vast number of users. Those standards are called *de facto* standards to distinguish them from the so called *de jure* standards. As examples, in the world of communications protocols, OSI is a *de jure* standard defined by the ISO organization; SNA* is a proprietary interface and a *de facto* standard developed by IBM and accepted by vast numbers of users, and TCP/IP is a public interface and a *de facto* standard.

An application written to a standard interface can be executed or ported to any data processing system adhering to that standard or, more precisely, to any data processing system where that standard is available. In current terminology, a data processing system that provides standard programming interfaces is also said to be, or is accepted as being, *open* because it allows portability of the applications to and from other systems where the same programming standards are available. The nature of the standard interface provided, national, international, de jure, or de facto, can also be used to qualify the level of openness of the data processing system. See section 2.9, “Open Systems” on page 36 for a more complete discussion and definition of the open system concept.

There is a common misconception about open and proprietary being mutually exclusive characteristics of a data processing system. Any proprietary system with a layered software structure can also be or become an open system with respect to a standard interface by developing a layer of software that provides that interface.

As an example, the MVS* platform, traditionally a proprietary platform with several *de facto* standard interfaces, has recently taken steps to qualify as an open system by adding layers of software to support, among others, the

standard POSIX** 1003.1 interface (see section 2.9, “Open Systems” on page 36 for a more complete discussion on open systems and POSIX** standards).

Another important characteristic of an interface is the way it changes when the software layer is subjected to a technological advance or enhancement. The changes may be compatible or incompatible.

Changes are compatible when they do not require applications be modified to continue to function. Incompatible changes force the applications to be modified. Compatibility can also be further qualified as *upward compatibility* when the evolution takes the form of an extension to an existing interface that causes no alterations or modifications to the existing one. In this case, the applications can continue to function without modifications unless there is need to make changes to take advantage of the new features available with the extensions.

### 2.3.2 Software Layers Visibility

Figure 2 on page 7 shows the software layers piling up in such a way that each layer does not completely overlap the layer below. This graphical representation is intended to address an important characteristic of software layers and programming interfaces which is the visibility of the interface.

An interface is said to be visible when it can be used by other software layers or applications. In a software structure where several layers overlap, leaving visibility of interfaces, it is left to the application to determine at which level to interface with software. In other words, when the software structure provides visibility of several software layers, the use of each layer is not mandatory for the applications, but it is an opportunity for the application to use the services provided by the software instead of having to develop them.

As an example, assume a situation where the application is a query to a relational database, and the software structure provides the services of a query manager, a transaction manager, a database manager, and the operating system with its access methods.

The following options are available to the application designer as in Figure 4 on page 11 (some of the options are purely theoretical and are used only for the purpose of the example):

- Develop a query, and provide an interface at the level of the query manager.

  The query manager will interface with the lower levels of software to process the query. As part of the process, the query manager might also provide a graphic display to the response.

- Write a transaction, and access the database through the services of a transaction manager.

  In this case the application has to develop the required graphic display services previously provided by the query manager.

- Interface directly with the database manager.

  In this case, the application has to provide its own logic to access the database manager and to qualify as a manageable entity.

- Interface directly with the access method and the operating system, or interface directly with the machine-level interface.

The additional responsibilities for the applications in these cases are obvious.

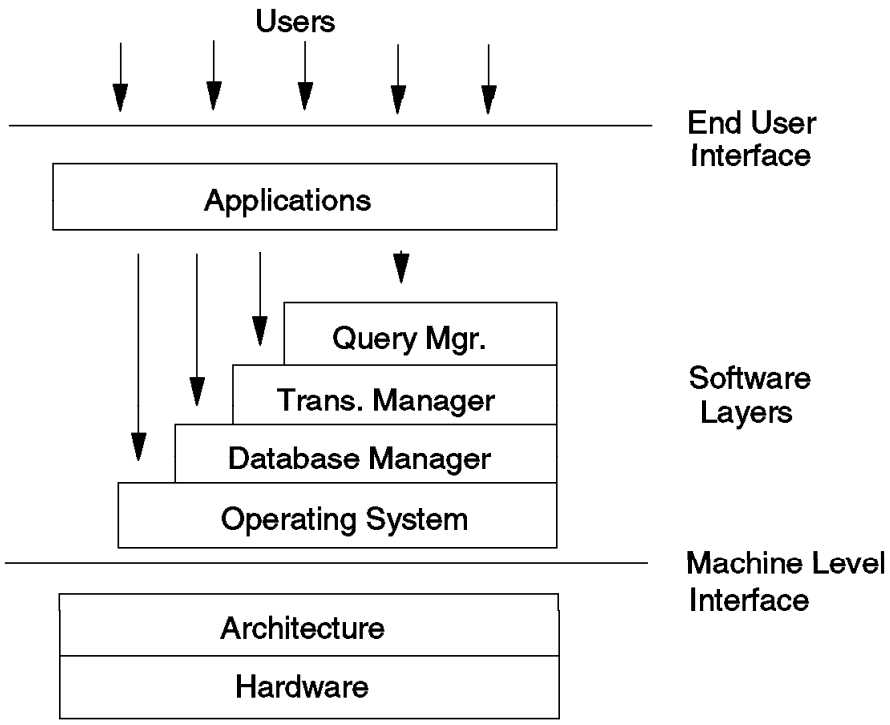


*Figure 4. Software Layers Visibility*

The example clearly demonstrates the role of software as service provider and the service provider/service requester relationships between the software and the applications. The higher the level of the software interface used, the lesser the development effort for the application. This characteristic is the basis for the term *high-level* or *low-level* interface. Among others, the advantages of high level interfaces are:

- Ease of use
- Investment protection

Ease of use derives from the fact that the higher the level of the interface, the closer it is to everyday language.

The software layers beneath the interface have to interpret the request and translate it into the technical language of the lower-level layers. Ease of use can easily be translated into both user productivity of the user and usability of the entire information system. With a high-level interface, the application is easier to design; faster to develop, test, and become productive; and easier to maintain.

Similarly, less skill and training is required for the user to be able to access and use the system.

High-level interfaces also provide application investment protection, because by shielding the applications from the technicalities of lower layers, they also provide protection from changes in the lower levels of software due to technology evolution. Provided that the applications are based on programming interfaces that can be maintained across technology evolution, the applications will not automatically become obsolete because of a technology advance in the

hardware, the software, or the architecture. The higher the level of the interface, the more likely it is that the software will provide the changes required to convey the benefits of the technological advances to the application through a compatible programming interface.

### 2.3.3 Architecture Layer

The architecture layer in Figure 2 on page 7 is in a unique position with respect to the software layers. In fact, the software layers overlap in such a way as to guarantee the applications the visibility of several layers, while the architecture layer completely overlaps the hardware layer, leaving no visibility of it. The graphical representation indicates that any hardware component can be used only according to the rules, principles, and protocols defined by its architecture.

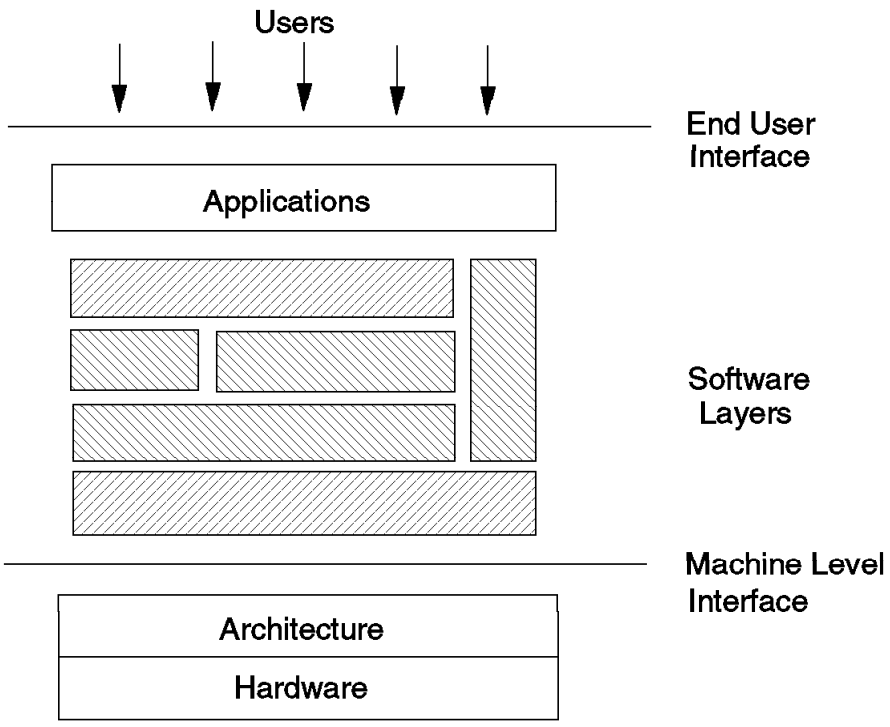


*Figure 5. Data Processing Software Structure*

### 2.3.4 Additional Considerations

Finally two general considerations about the layered structure of system software of Figure 2 on page 7 and the components of a data processing system as in Figure 1 on page 3.

#### 2.3.4.1 Layered System Software

The graphical representation of Figure 2 on page 7, with several horizontal layers, illustrates the concept of the software as a service provider for the applications, the role of the programming interfaces, and the meaning of the terms high-level and low-level interface. The figure does not intend to imply that a software structure has necessarily to be horizontal. We have already mentioned in section 2.3, “Software Layers” on page 7 that the same software

layer may in fact include more than one product and it might well happen that products functionally belonging to the same layer communicate among themselves horizontally. Similarly, software services such as security and systems management, can be invoked by any layer, independent of their position or level in the structure.

Once the implications of the interfaces, visibility, and structure are understood or implicitly accepted, other graphical representations can be used to illustrate other properties of the software structure. For example, Figure 5 on page 12 could be used (like Figure 2 on page 7) to represent the concept of a system where the applications can benefit from several software services provided by several software components whose physical relationships among themselves and between themselves and the applications are not relevant.

#### 2.3.4.2 The components of a Data Processing System

Applications, software, hardware and architecture are the main components of a data processing system. While it is obvious that complete independence of the four components is highly desirable for a solution, it is also a fact that technical constraints and practical considerations are often a limit to that objective.

Several research efforts currently being undertaken in data processing technology are aimed at removing these constraints and making it practical to provide a higher level of freedom of choice.

At the level of the architecture-hardware relationship a fair degree of independence has already been achieved to the extent that the same architecture (ex. the ESA/390*, or Intel 80X86**) is often available on more than one hardware implementation. Different hardware implementations providing the same architecture interface are usually called *software compatible* to illustrate that they all provide the same machine level interface and are therefore all capable of running the same software.

The relationship between the software and the architecture is dependent on the objectives given to, or defined by, the software developer. As already discussed, software may be written to exploit a specific architecture, or to be portable across multiple architectures.

Software portability allows the same software to be executed on various hardware platforms and, as a consequence, allows applications written for that software to be executed on various hardware platforms (for example UNIX** applications). In this environment we can say that there is a tight relation between the application and the software and a loose relation between the software and the architecture.

An additional step towards independence of choice for the solution architect may be achieved by loosening the relationship between the application and the software and allowing an application, written for specific software, to execute on different software (for example an Apple** MAC** application to execute under OS/2* or DOS Windows** or UNIX**).

In addition to the old and traditional method of software emulation, the most recent developments in software technology address this requirement by allowing specific software to run *foreign* applications or, to use current terminology, to have multiple *personalities*. WindowsNT**, OS/2* and the Workplace OS* family of products are examples of this multi-personality approach.

A software implementation that provides multiple *personalities*. towards applications, and portability across architectures and hardware platforms, allows the highest level of independence between the components a data processing system.

## 2.4 IBM S/390* Mainframes

The IBM S/390* system is used to describe that path of the evolution of data processing that, starting with the system shown in Figure 1 on page 3, has today arrived at what are usually identified as *traditional mainframes* or *mainframes*.

The drivers of the evolution that started in 1964 with the IBM S/360* systems include:

- Unconstrained growth, or the ability for the customer's workload to grow in terms of total applications, concurrent number of users, and volume of data with no loss in performance due to structural constraints in the hardware, software, or architecture.
- High level of service and performance, where service includes security, privacy, and integrity, in addition to programming services. Performance includes reliability, availability, and serviceability, in addition to application and processor performance.
- Fully compatible technology evolution to protect customer investments in corporate applications and data processing skills.
- Fully documented, functionally rich, general-purpose architecture to allow the development of several operating systems such as MVS/ESA*, VM/ESA*, VSE/ESA*, TPF*, AIX/ESA*, and several types of applications such as commercial, scientific, numeric intensive, I/O intensive, interactive, real time, and time sharing to suit the requirements of a vast population of users.

Among the several technology elements that can be highlighted to demonstrate the characteristics of a S/390* mainframe are:

- High multiprogramming capability - the ability to manage and control a large number of concurrently active users requesting system services and sharing system resources.
- Multiprocessing - the capability of the hardware, software and architecture to provide processing power in incremental steps without being technologically constrained by the maximum processing power obtainable with a single processor.
- Multisystem - the capability of the hardware, software and architecture to provide processing power in incremental steps without being technologically constrained by the maximum processing power obtainable with a single system.
- High bandwidth data paths for data transfers inside the processors and for data transfer with external storage resources.
- Data sharing - the capability of the system to maintain total data integrity with many users and applications concurrently accessing the same data.
- High availability - the capability of the system to continue to provide services to the applications even in the presence of some component failures.
- Resource management algorithms - to allow system resources to be over-committed by the user and managed by the system in priority order.

- Systems management services - to allow a high level of system management at a cost that is not impacted by the size and complexity of the installation.
- System integrity - by which system users have visibility and access only to resources for which they are authorized, and any deviation from this principle is considered a design error to be eliminated.

The evolution of IBM mainframes has taken place through a continuous update process to the hardware, software, and architecture of which only the major steps are discussed here.

The hardware for the S/390* systems is provided, today, by the large family of Enterprise System/9000* (ES/9000*) processors, including the 9221 low-end models capable of executing up to a few million instructions per second (MIPS), the 9121 midrange models, the 9021 high-end models capable of hundreds of MIPS (average for commercial workloads), and the parallel processing models: the S/390 Parallel Query Server, and the S/390 Parallel Transaction Server. Both deliver full data sharing.

This vast range in performance is achieved through several hardware technologies and through the use of several processor designs. Hardware technologies vary from low-power CMOS chips to the high-performance bipolar chips packaged on Thermal Conduction Modules (TCM) and TCM boards. Processor design includes several uniprocessor designs, several multiprocessor designs (n-way designs, and symmetric and asymmetric multiprocessors), and new parallel configurations.

The performance of the high-end processors is achieved through state-of-the-art design including:

- Superscalar design with multiple instruction execution elements and virtual register management for parallel instruction processing. (See section 2.6, “RISC* Technology and IBM RISC/6000*” on page 21 for more on superscalar design.).
- Sophisticated branch instructions that benefit from advanced technology branch prediction algorithms.
- Large first-level cache storage for each processor, with separate arrays for data and instructions.
- Very large, shared, second-level cache storage.
- High internal path bandwidth and high I/O path bandwidth to maintain a balanced flow of data in and out of the system.

The hardware structure of the high-end members of the ES/9000* family of processors provides for several units:

- A storage subsystem (storage controller and storage cards) for management of central storage.
- An Interconnect Communication Unit (ICU) for the management of the channel subsystem and expanded storage.
- Multiple Central Processors (CP), each with its own first-level cache storage or High Speed Buffer (HSB).
- A System Controller (SC) to which all the previous units are connected and that also contains a large, second-level cache storage shared by all CPs.

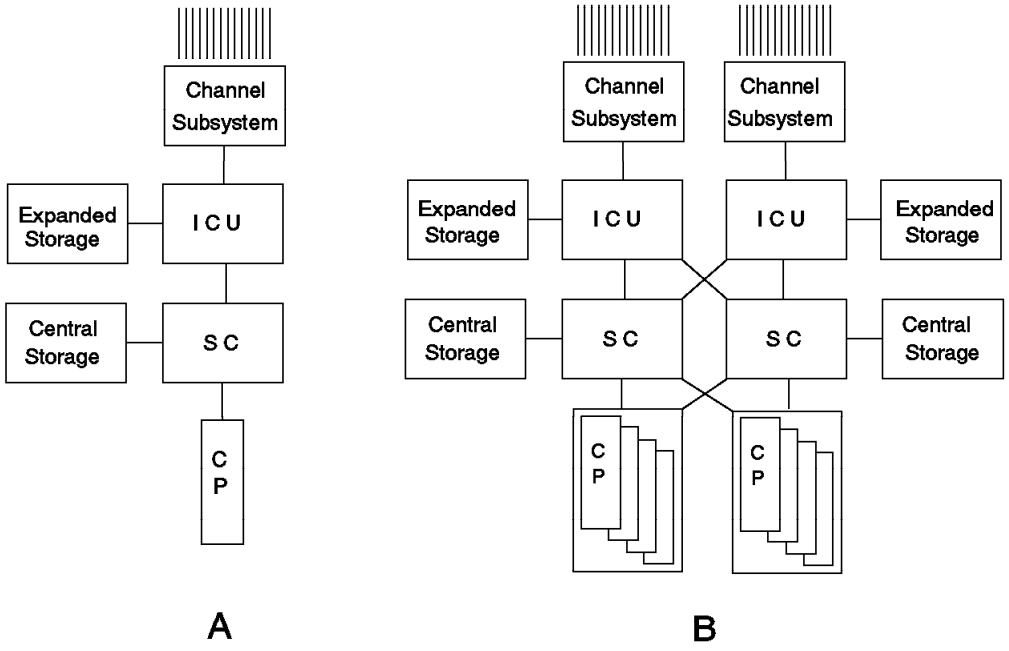


*Figure 6. High End ES/9000 Hardware Structure*

The minimum configuration is represented in part A of Figure 6 while part B represents a maximum configuration. Each ICU can handle up to 128 I/O channel paths.

The processor and I/O architecture for the S/390* systems is IBM ESA/390* architecture derived, through a long, compatible evolution, from the IBM S/360* architecture made available in 1964. The major steps of that evolution (to mention only the key components of each step) are:

- The System/360* (S/360*), defining the processor architecture (S/360* instruction set) and the I/O interface protocol (Standard S/360* I/O Interface).
- The System/370* (S/370*), introducing virtual storage and multiprocessing.
- The System/370-Extended Architecture* (S/370-XA*), introducing extended storage addressing and the independent channel subsystem for connection to the I/O subsystem.
- The Enterprise System Architecture/370* (ESA/370*) further extending the storage addressing capability, and introducing Dataspaces* and Hyperspaces*, in addition to the traditional address spaces.
- The ESA/390* architecture introducing the Enterprise System Connection* (ESCON*) I/O architecture for extended I/O connectivity, and the SYStem comPLEXes (Sysplex*) facility to increase the total system capacity.

- The S/390* Parallel Sysplex introducing the Coupling Facility technology [001] to implement full data sharing, subsystems workload balancing, and rapid recovery from failures.

Virtual and central (real) storage uses 31-bit addressing to provide addressability to a maximum of two gigabytes. Up to 16 terabytes of expanded storage are also accessible by the processor, the actual amount available being processor dependent. Central and expanded storage are also called processor storage. The operating system and the architecture allow multiple, independent, 2-GB spaces to be managed concurrently. The processor architecture provides facilities for multiprocessing operations, including atomic storage update instructions, to allow multiple processors to operate concurrently on the same shared storage, interprocessor hardware and software communication instructions.

Each evolutionary step of the architecture has been accompanied by a concurrent evolution of the hardware and software platforms to allow full compatibility for the applications and smooth (when not transparent) migration paths.

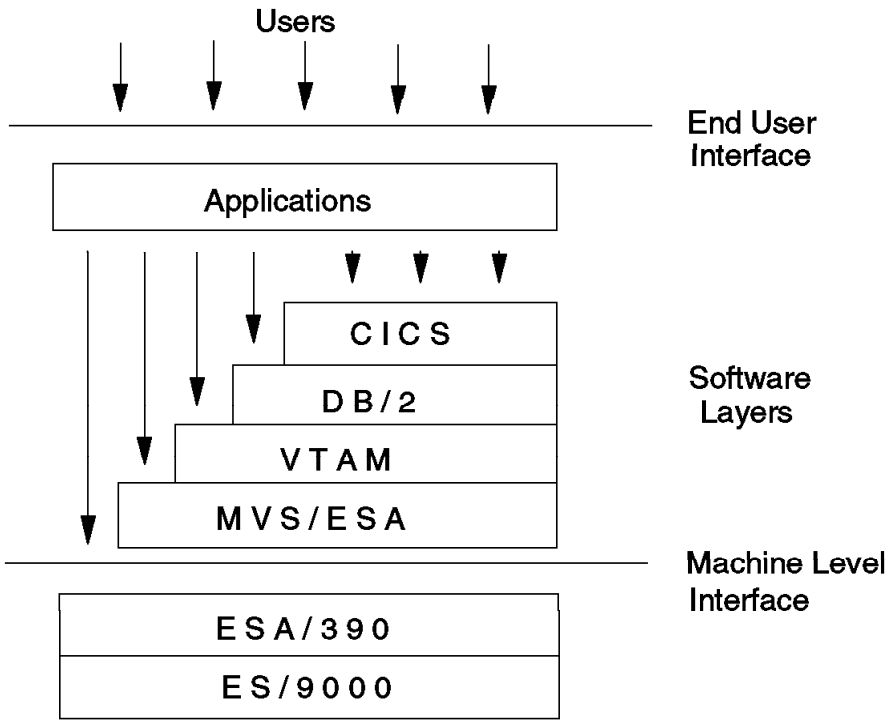


*Figure 7. S/390 Software Example*

Multiple software platforms are available to operate on a S/390* system including the MVS/ESA*, VM/ESA*, VSE/ESA*, and AIX/ESA* operating systems, plus a long list of software layers or subsystems available on each platform. (TPF, the specialized, high-performance operating system is not included in this document). Figure 7 is an example of Figure 2 on page 7, personalized to an

[1] The Coupling Facility technology is a combination of hardware and software functions supported by MVS/ESA*.

IBM S/390* mainframe, where, for simplicity, the software layers are limited to a minimum and include: the operating system (MVS/ESA*), a telecommunication access method, (VTAM*), a database manager (DB/2*), and a transaction manager (CICS*). In the figure, the vertical arrows represent the various levels of programming interfaces available to the applications.

More information about the main steps of the evolution of the S/390 mainframes can be found in the following publications:

- IBM Journal of Research and Development, Vol.27, N.3, 1983, for a discussion of System/370 Extended Addressing* architecture and the channel subsystem design.
- IBM Journal of Research and Development, Vol.36, N.4, 1992, for a discussion of IBM S/390* architecture and design.

Other vendors who manufacture products that fit into the category of mainframes are described here. It is useful to distinguish among two families of vendors:

- The software compatible vendors (SCV) that manufacture processors with some level (or full) compatibility with the IBM mainframes at the level of the machine interface. In this context, the terms compatible and compatibility refer to a processor that can execute the software and the applications of the corresponding IBM mainframe with identical results (the level of performance achieved by the individual processor does not affect compatibility).
- The full system vendors (FSV) that, as the name implies, manufacture processors that have their own architecture, operating system, and software products. Among the FSVs, it is possible also to distinguish between those that provide some level of compatibility with the IBM architecture and those that do not.

Among the SCVs are Amdahl Corp. and Hitachi Ltd. Among the FSVs that provide their own hardware and software systems with various levels of compatibility with IBM architecture are Hitachi Ltd. and Fujitsu. FSVs who manufacture mainframes not compatible with the IBM architecture are not discussed, because when a specific architecture reference is abandoned, the same definition of a mainframe becomes questionable and of little practical use.

The levels of performance of the mainframes in the class of the high-end IBM ES/9000 are the highest achievable by general purpose processors.

## 2.5 UNIX** Systems

UNIX** is a data processing system that evolved from the model of Figure 1 on page 3, but along a quite different path from the traditional vendor-provided proprietary systems. The term UNIX** in this discussion does not represent any of the multiple implementations of UNIX** available today in the industry, but rather describes those characteristics that are common across all the implementations.

UNIX** research started in 1969 at the Bell laboratories in the USA. The focus was on the structure of the operating system and particularly on the machine level-interface. The objective of the research was to design an operating system that would allow independence between the applications and the hardware. In other words, the UNIX** research was aimed at an operating system that could

be used on any hardware processor and architecture, and at the same time maintain the same programming interface to the applications.

Referring to Figure 1 on page 3, the UNIX** research dealt mainly with the software component of the system and the relationships between the software and the applications on one side, and the software and the hardware on the other.

The solution to the problem was an operating system, or *kernel*, with a defined set of services or system calls for the applications to use (the UNIX** application programming interface), and another defined set of services for the hardware to provide. Today, the kernel is generally written in *C* language. To execute the kernel, an additional layer of software is required between the kernel and the hardware.

Such a layer, here called the *low-level kernel*, is local or private to each processor and implements the services required by the kernel on that specific processor, according to the rules of the local architecture. Figure 8 shows that where the same kernel operating system could be executed on multiple systems, each is provided with its own local low-level (LL) kernel.

In this design, a UNIX** kernel can be executed on any hardware platform and on any architecture, provided that an LL kernel is developed and, of course, a C compiler is available for that hardware and architecture.

In addition to the kernel, the UNIX** system also includes a command interpreter called the *shell* and a library of utility programs called *utilities*. In UNIX** terminology, everything above the level of kernel, shell, and utilities is considered an application.

It is important to observe at this point a conceptual difference that exists between the UNIX** approach to a data processing system and the traditional proprietary systems. In traditional proprietary systems, the hardware, software, and architecture all cooperate to achieve the system objectives, and each component relies on the characteristics of the others. Major advances in hardware, architecture, or software technology are delivered to the applications through the complex software structure of the operating system and subsystems, and always maintain compatibility at the application programming interface level. As a result, in such environments the applications depend heavily for their execution and performance on the features, functions, and capabilities provided by the hardware and the architecture through the software.

A UNIX** system, on the other hand, has the objective of making the applications independent of the hardware and the architecture. In order to achieve that independence, the kernel and the applications are not expected to make assumptions about features being present in the hardware or in the architecture.

Due to the free circulation of the UNIX** source code in the early days of development and use, several implementations of the UNIX** system exist in today's market. Among the most popular are System V** from AT&T Corporation; Berkeley Software Distribution** (BSD**) from University of California, Berkeley; HPUX** of HP**; ULTRIX** of DEC**; UTS** from Amdhal**; and OSF/1** from the Open Software Foundation. The IBM implementation of UNIX** is the AIX* family of operating systems.

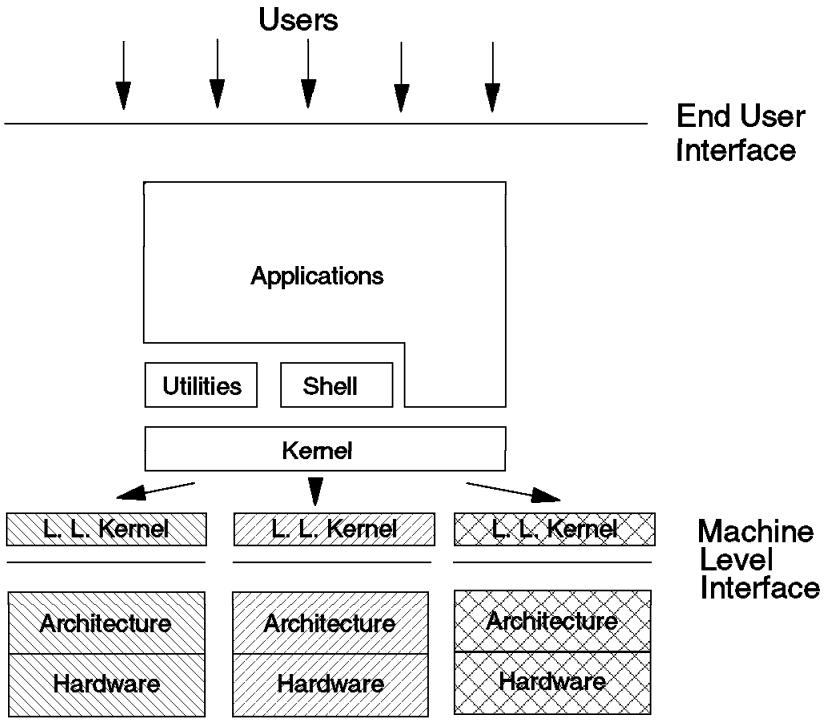


*Figure 8. UNIX** Model*

The development of various implementations of the UNIX** system, similar but not identical at the application programming interface level, has impacted one of the primary characteristics of UNIX**, namely the portability of programs and applications across different hardware platforms. The increasing need to invest in porting activity across UNIX** platforms raised, in the UNIX** community, the requirement to define a standard version of the UNIX** system.

The task has been undertaken by the Technical Committee on Open Systems (TCOS) of the Institute of Electronic and Electric Engineers (IEEE**) with the mission to define a standard for the application programming interfaces for UNIX** and, more generally, to define a set of standard interfaces to satisfy the requirements of Open Systems (see section 2.9, “Open Systems” on page 36).

As part of the TCOS activities, the standard for the kernel programming interface has been defined as POSIX** 1003.1, and the standard for the programming interface of shell and utilities has been defined as POSIX** 1003.2, where POSIX** stands for Portable Operating System Interface for Computer Environments (see section 2.9, “Open Systems” on page 36). UNIX** applications written according to the POSIX** standards do provide portability across hardware platforms where those standards have been implemented by the local version of the UNIX** system.

The portability of applications resulting from the UNIX** design is sometimes used to draw the conclusion that to achieve application portability it is necessary to adopt a UNIX** system. This conclusion is not correct because, as we have already discussed in section 2.3.1, “Programming Interfaces” on page 8, the portability of the applications does not depend on the design of the operating

system, but on the programming interface used by the applications and provided by the operating system.

Once an application is written using a standard programming interface, that application can be executed on any platform where that standard interface is available. For example, an application using the standard POSIX** 1003.1 programming interface can be ported on any UNIX** system where POSIX** 1003.1 is available, and also to an IBM S/390* system running MVS* or VM, or to an IBM AS/400* system when that interface is available.

## 2.6 RISC* Technology and IBM RISC/6000*

The RISC* (Reduced Instruction Set Computer) technology is the result of a project started by IBM in the '70's to define the architecture and design of a processor much faster than available in those days.

The project showed that it was possible to improve the raw speed of a processor by reducing the complexity of the instruction set, and possible to design a processor optimized to execute the new instruction set.

**Note:** The R of RISC* stands for *reduced* and refers to the complexity, not the size, of the instruction set.

To describe the basis of RISC* technology, consider that the speed at which a processor can execute instructions depends on:

- The cycle time of the processor, or the time required by the processor circuitry to go from a known state to another state.
- The number of cycles required, on average, to execute the instructions of the instruction set.
- The number of instructions required to execute a specific task.

RISC* research, starting from the then-current mainframe technology of the S/370* systems, addressed all three areas and defined ways to improve the performance in each of them to achieve the objective of an instruction execution rate of one instruction per machine cycle.

Referring to the model of Figure 1 on page 3, RISC* research is concerned with the components below what we call the machine level interface - the hardware and architecture. The design of the compiler is also affected by RISC* technology.

RISC* research reduced both the cycle time of the processor and the number of cycles per instruction by reviewing the S/370* architecture and reducing certain levels of complexity that affected processor performance. A new data flow model of the architecture reduced the levels of logic required to perform a cycle and, consequently, the cycle time.

Some lengthy S/370 instructions, such as storage-to-storage and data moving instructions with protection key handling, were eliminated and replaced with faster register-to-storage and storage-to-register instructions.

It was then left to the compiler or to the program to replace the functions and the services provided by the eliminated instructions, using both new instructions and new programming techniques.

The extensive use of registers in the instruction set as fast storage, the hardwired processor implementation to avoid the delays introduced by microcode, and the pipelined processor design to increase the instruction execution rate are additional key elements of the first generation RISC* machines. The reduced amount of logic required to implement the architecture also made a compact and efficient processor design possible.

The emphasis on *reducing the complexity* of the then-current S/370* architecture has remained in the name RISC* given to the new processors.

Among the first IBM products commercially available that used the RISC* technology were the IBM RT* System, the internal microprocessors of the IBM ES/9370* system, and the IBM ES/3090* channel subsystem processor. Those early implementations of the RISC* technology are known today as first generation RISC*.

By the time they were put on the market, in 1985, the IBM research on RISC* technology had resumed to further improve the performance and usability of RISC* technology. The objective was a superscalar processor capable of executing more than one instruction per machine cycle.

An additional objective was to extend the architecture to include important functions not included in the first implementation, such as the floating point arithmetic. The main results of this second phase of research are the IBM POWER* (Performance Optimization With Enhanced RISC*) and POWER2* architectures, the IBM RISC System/6000* (or RISC/6000* or RS/6000*) processor, also called second-generation RISC*, and the Scalable POWERparallel System.

Among the most important features of the second-generation RISC* research are:

- Superscalar implementation with multiple, specialized execution units and multiple instruction dispatch for simultaneous execution
- Separate instruction and data caches to maintain data and instruction bandwidths sufficiently high to allow simultaneous instruction execution
- Specialized branch architecture to allow highly efficient pipelining
- Floating point unit specialized for high performance.

The RISC System/6000* hardware is distributed among a CPU planar, an I/O planar, and a standard I/O planar card.

The CPU planar card contains the processor and the storage cards. The number and size of the storage cards is model dependent.

The I/O planar card contains the slots for attaching MicroChannel* adapters, such as tape drive adapters, LAN (Ethernet and Token Ring) adapters, display and graphic adapters, coprocessors, and printer adapters. The I/O planar card also contains other devices for system control functions, such as the CPU initialization at IPL, the operator display, the system status, and others.

The standard I/O planar card contains the interfaces and connectors for non-MicroChannel* devices, such as the keyboard, the mouse, the diskette, and others.

The RISC System/6000* employs the POWER* architecture for the processor. Storage addressing provides for a single virtual storage space of 4 PB and a real storage space of 4 GB. Actual implementation is software dependent.

The I/O architecture is MicroChannel* for most of the I/O operations plus two high-speed optical serial link adapters (SLA) for high-I/O bandwidth.

The characteristics of the RISC/6000* processors make them suitable and competitive in the powerful market for intelligent workstations and servers. As this market is predominantly a UNIX** market, the software support for the IBM RISC/6000* is provided by the AIX* V.3, which is the IBM version of UNIX** for the RISC/6000* platform. AIX* V.3 is derived from UNIX** System V, conforms to the IEEE** standard POSIX** 1003.1 of 1990, and is compatible with Berkeley Software Distribution** 4.3 (BSD** 4.3).

The Scalable POWERparallel System 9076 SP2 is based on the RISC/6000* technology and can have up to 128 nodes, where each node is a RISC/6000* processor. The nodes are connected through a High-Performance Switch (HPS) that allows any-to-any communication. The Scalable POWERparallel System enables high performance parallel processing for computational intensive and UNIX** business critical data query and transaction processing.

More reference information on RISC* technology and the IBM RISC System/6000* can be found in the IBM Journal of Research and Development, Vol.34, N.1, January 1990.

Today, the term RISC* is commonly used in the industry to identify a wide range of microprocessors that share the characteristics of limited size and high instruction execution rate. Among the various non-IBM processors that are known as RISC* processors are:

- DEC** Alpha
- MIPS R3000, R4000, and R6000
- HP** Precision Architecture
- Intergraph C4
- Intel i860, and i960
- AMD 29000
- Cypress Pinnacle
- Motorola** 88000
- National 32000
- SGS-Thompson T9000
- SUN Sparc**

The operating system and the software structure of these processors is mostly UNIX** based.

When the RISC* technology became widely known on the market, the acronym CISC (Complex Instruction Set Computers) was also adopted to identify the traditional complex architecture based processors.

## 2.7 IBM Application System/400* (AS/400*)

AS/400* is used in this discussion to refer to the whole IBM Application System/400* family of processors.

The AS/400* is the result of a research and development effort by IBM aimed at building a family of general purpose, mid-range data processing systems especially suited to support customer applications and to be the successors of the System/36 and System/38 product families.

In absence of a more precise and commonly accepted definition, we will assume, for the purpose of this discussion, that mid-range refers to a data processing environment where the requirements and the size of the installation (not necessarily the processing power) are between those of a full sized mainframe and those of a workstation environment.

A typical characteristic of mid-range environments is that the majority of customers tend to buy ready-made applications from software vendors instead of developing them internally. As a consequence, vast quantities of ready-to-use applications are available from thousands of independent, worldwide, software vendors for those types of systems.

This situation more and more extends the requirement for an easy to use and efficient application programming interface and for a commitment to compatibility. Actually, the requirement to maintain compatibility with the applications running on the hundreds of thousands of System/36 and System/38 processors around the world has been one of the major challenges for the development of the AS/400*. The strong emphasis on the application orientation of the AS/400* is reflected in the name *Application System*.

The most relevant elements of innovation introduced by the AS/400* in terms of system structure are:

- High-level machine interface
- Object-oriented design
- Single-level storage

The AS/400* system is *visible* to the operating system and to the users through a high-level machine interface (MI), where *high-level* has the meaning described in section 2.3.2, “Software Layers Visibility” on page 10, and is relative to the machine interfaces (instruction sets) provided by other industry systems.

Unlike the traditional systems (see Figure 2 on page 7), the AS/400* Machine Interface implements functions that traditionally are provided by the operating systems and higher-level software layers. Such functions include, among others, I/O management including the low-level layers of network communication, security, integrity, selected office services, database management, storage management, and task management. As usual, nothing below the level of the machine interface is visible to the operating system or to the applications.

The implementation of the machine interface is accomplished through two layers of microcode (or licensed internal code). The hardware and the first layer of microcode implement a first level of machine interface called the Internal MicroProgramming Interface (IMPI). The IMPI instruction set is similar to the S/370* instruction set with the addition of some composite instructions for branch

procedures, such as compare-and-branch and test-and-branch, and other extensions. The IMPI interface is not directly available or visible to the user. The IMPI interface is completely overlapped by an additional layer of microcode. This layer of microcode implements the higher-level machine interface (Machine Interface or MI) that is used by the OS/400* operating system, and is the lowest level programming interface available to the application programmer.

The multilayered structure of the AS/400* architecture provides a high level of flexibility because it provides instructions that are hardware independent. At execution time, the MI instructions are *translated* into IMPI instructions by an internal microcoded component called the *translator*. With this structure, it is possible to modify the technology of the low-level layers in the system (hardware components, instructions implementation) with no impact on the MI interface or on the applications.

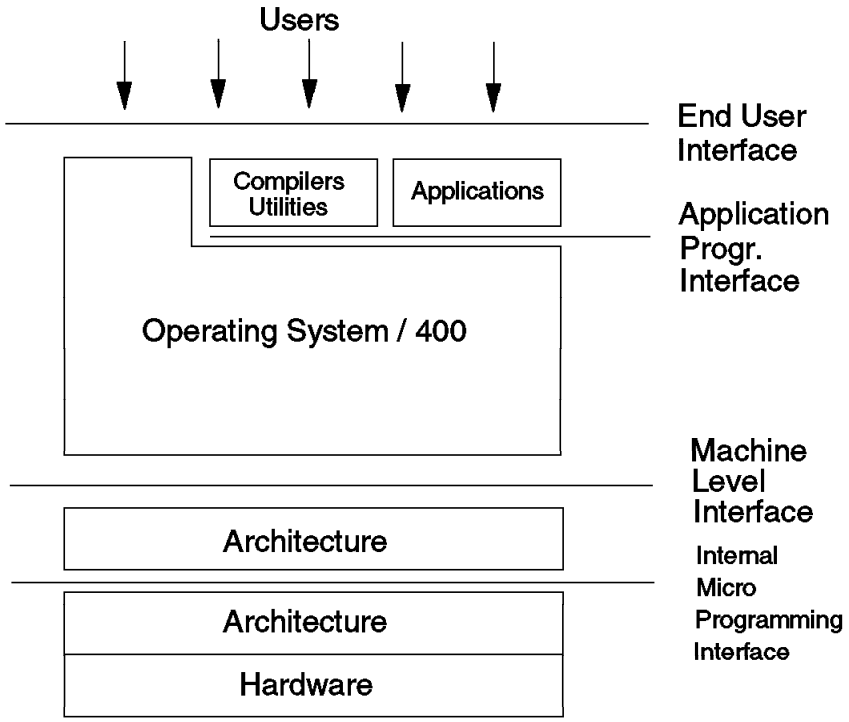


*Figure 9. AS/400* System*

In AS/400* terminology, *objects* are the means by which information is stored and processed. Everything that the user can store, retrieve, or process is known as an object. The high-level machine interface (MI) allows every object to be treated the same through the use of a generic object structure that is common to all the objects (for example, each object has a type identifier that determines how the object can be used, once retrieved). The internal representation of the data and attributes of an object are not visible to the user. This approach makes users independent of the addressing structure defined by the architecture and the implementation techniques, providing ease of use through consistent handling of interfaces, and hiding the individual object's internal complexity. Other elements that provide a description of the objects include: the owner, the object size, the creation date, the last reference date, the last update date, the last medium where the object has been saved, and text describing the object.

The AS/400* system can handle 6-byte virtual addresses, thus providing addressability to any byte within a $2^{48}$ byte addressing space or about 281,475 GB of contiguous addressable space. This enormous addressable space is called single-level storage and is used by the system to hold every program or data object in the system. The relationships between virtual storage, real storage, and auxiliary storage (magnetic disk space) are completely transparent to the user. Whenever an object is addressed by the user (for example the call to a program), the system ensures that the object is accessible by bringing it into real storage. This storage addressing scheme permits the applications to be independent of the stored location of the objects, the storing devices, and the system I/O configuration. (Actually, the processor architecture has the potential to expand the addressing capability to 64-bit addressing with no impact on the applications, should such requirement appear in the future.)

Figure 10 represents the single-level storage of the AS/400* system. The physical resources (central storage and auxiliary storage) are separated from the logical object's databases and programs by the single-level storage mechanism.

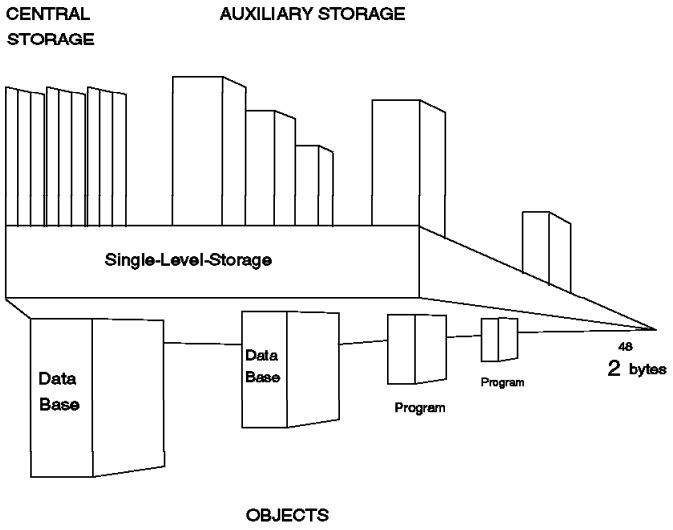


*Figure 10. OS/400* Single-Level Storage*

The software structure above the machine level interface is also layered and provides services for the end users, applications, system compilers, and utilities. The end-user interface, the highest level of access to the system, provides access to the software and machine-level services through panels and menus designed for ease of use. As far as the user interface is concerned, the AS/400* provides the user with an easy to use interface, even towards complex functions, such as database management systems, communications, security, resource management, and problem determination.

Two primary user interfaces are provided. They are a menu-driven, interactive, full- screen, display interface for accessing objects and performing actions on them, and a command interface for fast-path access to frequently used functions. Online tutorial and help information, and support for national languages are additional ease-of-use items.

As far as the hardware structure of the AS/400* is concerned, the system processors or, simply, the processors, communicate with multiple, independent, I/O processors over high-speed buses for direct data access. A bus control unit (BCU) that handles functions such as arbitration error handling and initializations is located within the processor. The number of processors, I/O buses, and I/O processors is model and configuration dependent. Figure 11 provides a simplified representation of the hardware structure of an AS/400* n-way model.

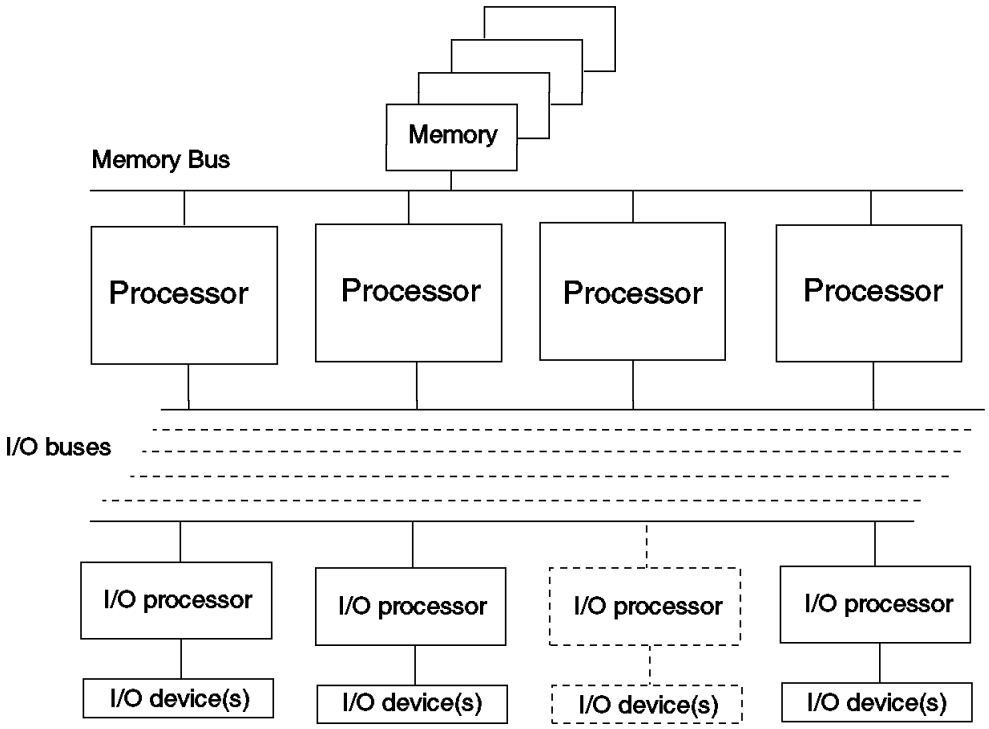


*Figure 11. AS/400* System Structure*

Depending on the devices to be managed, several types of I/O processors execute specialized code loaded at initialization time. For example, the communications I/O processors are initialized for specific protocol tasks and execute data link control functions according to the communications architecture in use, and the workstation I/O processors provide end-user oriented functions, such as field editing, keystroke processing, text processing, and national language support.

The system processor is also directly connected to main storage, and access is provided by a hardware component, the Virtual Storage Translator (VTA), that converts virtual storage addresses to main storage accesses. The I/O processors have full, Direct Memory Access (DMA) capability to main storage through the bus.

The processor architecture, as already described, is defined by the Machine Level interface (MI), which is the lowest level of interface visible to the user. How MI provides high-level functions that normally are provided by programs, and not by an instruction set, has also been described. An additional internal level of processor architecture is provided by the Internal Microprogramming

Interface (IMPI), not visible to the user. MI is implemented over IMPI by two layers of microcode.

Virtual storage uses 64-bit addressing of which only 48 bits are used at execution time (48 bits provides addressability to about 281,475 GB). The addressing capabilities of the AS/400* provides for a single level of (virtual) storage to maintain all system data, user data, and programs. The MI provides the services required to map the allocated virtual storage into the existing real storage and to back up data on auxiliary storage devices.

The operating system for the AS/400* is the OS/400*.

More information on the AS/400* characteristics and technology can be found in the IBM Systems Journal, Vol 28, No. 3, 1989.

## 2.8 Personal Computers

Personal computers are the result of the data processing evolution that extended the computing technology to everyday activities. The appearance of the first models of personal computers (as they were named at that time) in the middle *70*s produced a step forward in the world of data processing comparable in size to, and perhaps superior to, the appearance of the first computers for commercial use back in the ′50′s. As the traditional mainframes allowed the industry to automate everyday commercial processes, such as billing, accounting, personnel, and order entry, the personal computers allowed individuals to automate their everyday activities such as, text editing, mail, messaging, spreadsheet, and games. The qualifier itself used for those processors, *personal*, reflects the individual orientation of a computer intended for the personal use of a single user.

Today, the PC marketplace is characterized by fast growth, intense competition, a fast rate of technology development by the hardware industry, and a software industry attempting to keep up with these hardware improvements.

Today there are two major groups of PCs:

- **IBM PCs and compatibles**. These are all based on the Intel x86 series of processors. This market is very competitive between a large number of manufacturers, including IBM, Compaq, and Dell.
- **Apple Macintosh**. These are based on the Motorola 68xxx series of microprocessors and are incompatible with the IBM PCs. Apple is the only manufacturer of this group.

We will discuss the PC arena in the following six areas:

- User Interface
- Applications
- System Software
- Hardware
- Distributed Systems
- Application Development

### 2.8.1.1 User Interface

One of the reasons for the success of PCs has been their capability to provide a highly interactive and visual user interface. The standard for PC user interfaces has become the **Graphical User Interface (GUI)**, which is characterized by a large bit-mapped graphics display, a mouse as well as a keyboard, and a style of interaction using icons, pointing, and direct manipulation.

End users have found this considerably more productive and usable than character-based terminals. Since the display is local to the PC, the bandwidth required to transfer large quantities of graphical data between processor, memory and display is available, and low-cost video adapters and display hardware have become standard.

Three GUIs currently have significant market-share for PCs

- Apple′s **Macintosh System 7** user interface brought the power and usability of GUIs into the mainstream of computing. System 7 runs on Apple Macintosh hardware which may now include POWERPC microprocessors.
- Microsoft′s **Windows 3.1** provides a GUI on top of the DOS operating system, running on IBM PCs and compatible hardware.
- IBM′s **OS/2 2.1 Workplace Shell** provides a GUI as an integral component of the more powerful OS/2 2.1 operating system for IBM PCs and compatible hardware.

New technologies currently being absorbed into GUIs include:

- **Object-Oriented User Interfaces (OOUIs)**, which enable the user to focus on real-world objects, such as documents and printers, instead of the application focus of GUIs. Both Macintosh System 7 and the OS/2 Workplace Shell are OOUIs. Microsoft has indicated their intention to add OOUI capabilities to future versions of Windows and Windows NT.
- **Multimedia**, such as digital audio and software motion video, is standard in OS/2.1 and available as extensions to both Macintosh System 7 and Windows 3.1.

Other user interface technologies are emerging and are expected to enter the mainstream as the hardware becomes sufficiently powerful and affordable. These include Pen-based Computing, and Speech Recognition. Both pen and speech are currently available as extensions to all three GUIs.

### 2.8.1.2 Applications

In the PC world, application packages are the major type of application software. Tens of thousands of application packages have been developed by a wide variety of Independent Software Vendors (ISVs), such as Lotus, Borland, Wordperfect, and Microsoft. Application packages typically provide functions, such as spreadsheets, word-processing, graphics, and electronic mail.

Applications can also be classified by the original operating system they were written for. This includes:

- **DOS applications**, the major group of PC applications, including both character-based and graphics applications.
- **Windows applications**, the most common GUI applications.
- **Macintosh applications**, the GUI applications for the Apple Macintosh. It is particularly strong in the desktop publishing and and graphics areas.

- **OS/2 applications**, an emerging class of 32-bit GUI applications able to exploit the powerful OS/2 system facilities, such as large flat memory address spaces and multiple threads.

As new operating systems, such as Windows, Windows NT, and OS/2 have been developed, it has become essential for them to continue to support the legacy applications of previous operating systems such as DOS and Windows. Emerging technologies called **Personalities** enable applications written for one operating system to run under another operating system - sometimes better than under the original operating system.

**Interoperability** between applications is also becoming increasingly important. This can be divided into three areas:

- **Clipboard** provides the ability for the end user to cut and paste data between applications.
- **Dynamic Data Exchange (DDE)** provides the ability for applications to establish program-to-program links to dynamically exchange data.
- **Compound Documents** enable documents composed of mixed text, graphics and other components, to be treated as a single document. The appropriate application is invoked automatically to process each part of the document. Two standards for compound documents exist:
  - **Online Linking and Embedding (OLE)** has been developed by Microsoft and is used between Windows applications.
  - **OpenDoc** has been developed by a consortium which includes Apple, IBM and others, and can be used between applications of different types, and also across distributed systems.

#### 2.8.1.3 System Software

The system software layer is dominated by a small number of products from a few vendors, primarily Microsoft, IBM, Apple, and Novell (Digital Research).

For IBM PCs and compatibles, the main system software products today are:

- **DOS**, which provides a basic set of operating system services, primarily a file system, a program loader, a command interpreter, a set of basic utilities, and more recently, task-swapping and menus. Various memory management extensions are also available to enable DOS applications to use more than the 640KB memory limitation of the early versions of DOS. Microsoft, IBM, and Novell (Digital Research) all provide versions of DOS. DOS, a 16-bit operating system, is the most common operating system.
- Microsoft **Windows 3.1**, which provides a GUI environment running on top of the DOS base. Windows also replaces many of the operating system services of DOS and provides its own memory management, along with limited cooperative multitasking. Windows operates within the 16-bit environment of DOS, and the DOS/Windows combination is the most common GUI operating system.
- IBM **OS/2 2.1** provides an advanced 32-bit operating system with integrated GUI and full multitasking between applications. OS/2 2.1 is the most common 32-bit GUI operating system.
- Microsoft **Windows NT** also provides an advanced 32-bit operating system with integrated GUI and full multitasking, built on a kernel that can be ported to new hardware.

For Apple Macintosh, the major system software product is:

- Macintosh **System 7**, which provides an advanced operating system with integrated GUI and limited cooperative multitasking.

PC system software typically consists of:

- A user interface
- One or more personalities that provide support for the applications
- The main operating system kernel
- A hardware abstraction layer of BIOS/ABIOS and device drivers.

This is shown in Figure 12.

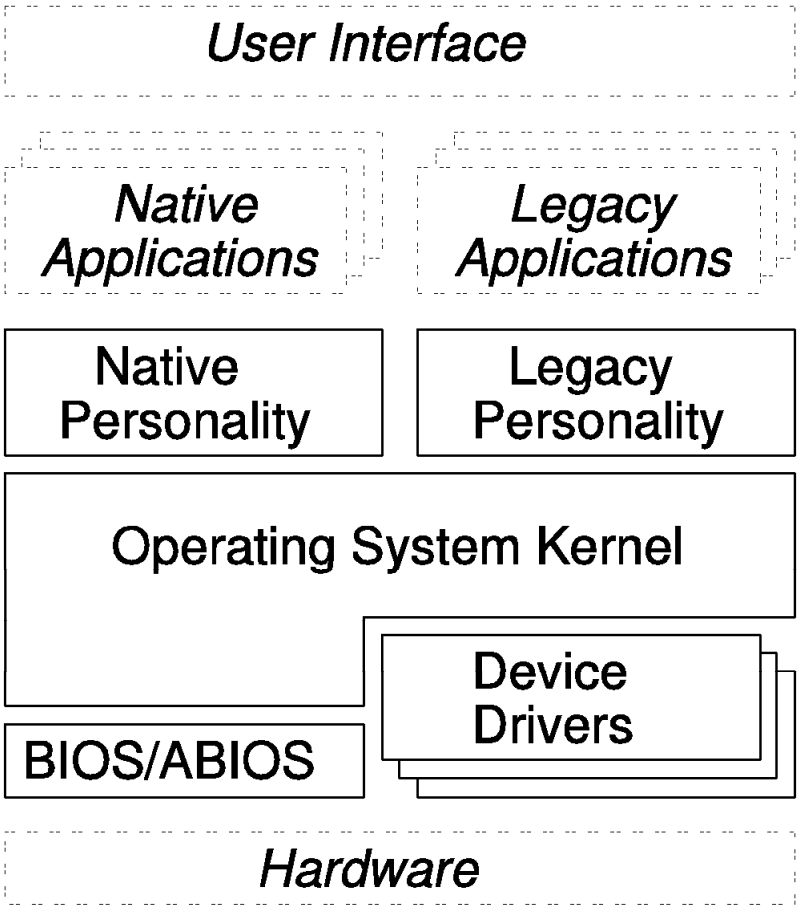


*Figure 12. Example PC System Software Structure*

The personality support for different application environments can be provided either by emulating the legacy application environment on top of the native application environment, as in Windows NT, or by providing a native legacy environment alongside the native application environment, as in OS/2 Windows applications.

The operating system kernel provides the fundamental system software capabilities, such as starting and multitasking programs, memory management, interprocess communication, and interrupt handling.

New processors, such as the emerging RISC processors discussed in the next section, have created a need for the operating system to be portable to different hardware architectures. This can be solved in two ways - rewriting the operating system for each new hardware platform, or restructuring the system software so most of it is portable and only a few components need to be rewritten for each new hardware platform. **Microkernel** technology is emerging as a key technology that can be used to reimplement current operating systems, such as OS/2, on a portable foundation.

Support for new hardware devices is provided by the device driver layer, enabling disks, CDs, and video adapters to be added to the system without rewriting or rebuilding the operating system.

The emerging use of **object-oriented** technology promises to provide solutions to the challenges of fast development of complex system software. IBM, Apple, and HP have sponsored a joint company, called **Taligent**, which is developing a fully object-oriented operating system. Technologies from Taligent are expected to be absorbed into existing IBM, Apple, and other operating systems over time.

#### 2.8.1.4 Hardware

The PC hardware itself can be divided into three areas:

- **Processor**
- **Hardware architecture**, such as the adapter bus and BIOS firmware
- **Hardware devices**, such as disks, video displays, and printers

An example PC hardware structure is shown in Figure 13 on page 33.

***Processor:*** The IBM PC and compatible hardware industry is dominated by the Intel x86 series of processors, whereas the Apple Macintosh is based around the Motorola 68xxx series of processors.

The **Intel x86** processor family today consists of the 386sx, 386dx, 486sx, 486dx, and Pentium processors. IBM also manufactures the 386slc and 486slc2 variants of this family, under license. This family has the following characteristics:

- 32-bit CISC (complex instruction set computer) processor.
- Uses an extended ASCII character set and little-endian byte-ordering mode.
- Hardware support for multitasking and for protected virtual memory.
- Able to operate in real 8086 mode, protected mode, and virtual 8086 mode (this is especially useful as a basis for emulating DOS and Windows sessions).
- 386sx processors use a 32-bit instruction set and 24-bit external addressing. All higher members of the family use a 32-bit instruction set and 32-bit external addressing.
- 486dx and Pentium processors include integrated floating-point units. For the other processors, floating-point coprocessors can be added.
- 486sx, 486dx, and Pentium processors include internal memory cache.
- The Pentium processors includes dual instruction paths.

The **Motorola 68xxx** processor family today consists of the 68020, 68030, and 68040 processors. This family has the following characteristics:

- 32-bit CISC processor

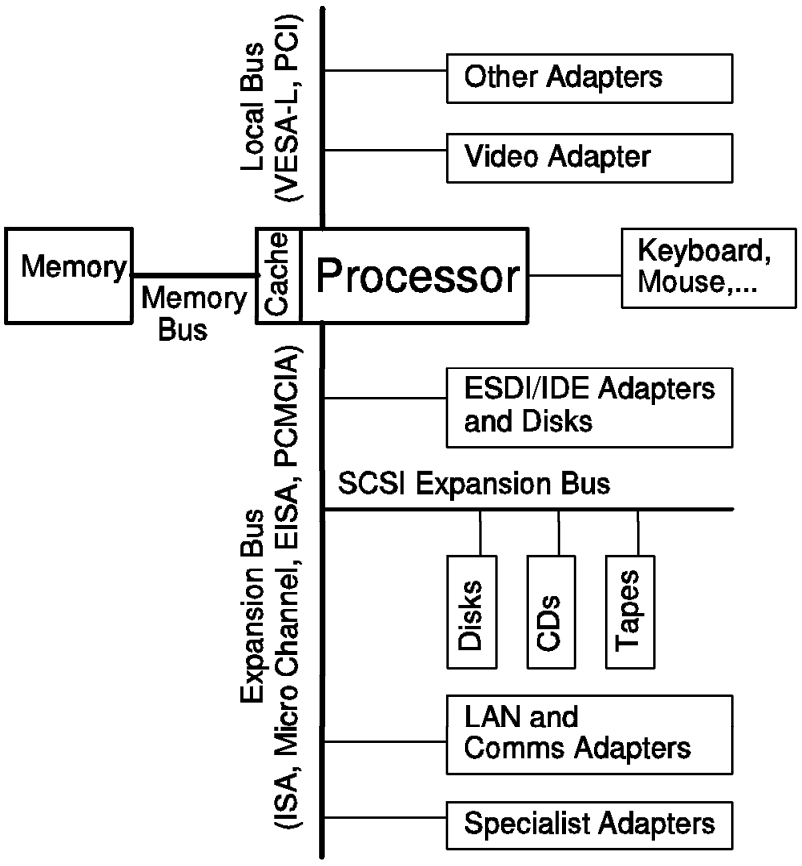


*Figure 13. Example PC Hardware Structure*

- Uses an extended ASCII character set
- Hardware support for multitasking and for protected virtual memory.

The recently announced **PowerPC** family of low-cost RISC processors are a joint development effort by IBM, Apple, and Motorola. The PowerPC processor family consists initially of the 601, 603, and 604 processors, and all of these have the following characteristics:

- 32-bit RISC processors, derived from the POWER processors developed and used by IBM in the RS/6000 workstation family.
- Able to boot in either little-endian or big-endian byte-ordering mode.
- Hardware support for multitasking and for protected virtual memory.

Symmetrical multiprocessing **(SMP)** PCs are also starting to emerge, containing more than one processor. Although they use standard microprocessors, such as the Intel Pentium, other elements of the SMP hardware architecture vary between manufacturers.

***Hardware Architecture.:*** Apart from the processor, the component that characterizes the PC most is the bus used to interconnect components and into which adapter cards can plug. The most common bus today is the **ISA** bus, which was originally used in the IBM PC AT. This bus has limitations for more advanced PCs, but compatibility considerations have enabled it to survive since 1984. Two newer buses, the **Micro Channel** bus, developed by IBM and used in most PS/2s; and the Extended ISA **(EISA)** bus, developed by Compaq and others and used especially in servers, have had some success. Since then, two new factors have emerged which are driving the development of the bus; the need for smaller and interchangeable adapter cards in laptops has spawned the **PCMCIA** bus, and the need for very fast video adapters to support GUIs has spawned the **VESA local bus** and the **PCI** bus, both of which provide very fast data transfer between adapter cards and the processor and memory. The Apple Macintosh uses its own advanced bus, the **NuBus**.

On IBM PCs and compatibles, the operating system is insulated from specific hardware implementations by a defined interface (the **BIOS**), and by installable **device drivers** which either provide an interface to the BIOS or interact directly with the operating system. The BIOS is a level of firmware, usually provided in read-only memory (ROM) on the system. BIOS is present on all IBM PCs; in addition IBM PS/2s using the Micro Channel implement a variant of the BIOS called **ABIOS**, which is optimized for use with multitasking operating systems, such as OS/2.

The BIOS does not provide sufficient insulation from the hardware for SMP. Therefore, the operating system currently has to be specifically tailored for each new SMP system. Microsoft provides SMP support in Windows NT, and IBM has beta-tested a SMP variant of OS/2 2.1.

***Hardware Devices:*** A wide variety of hardware devices are produced by a large number of Independent Hardware Vendors (IHVs). These devices are normally connected into the bus either through their own adapter card, or by sharing an adapter card. Hardware devices include:

- Peripheral buses, such as **SCSI**. SCSI is a standard bus that enables disks, CDs, and other storage devices to share the same adapter card.
- Fixed Disks, such as **ESDI**, **IDE**, and **SCSI**. Fixed disks provide fast-access, read-write permanent storage, with disk sizes typically of a few hundred MB, and fast access times.
- **CD-ROMs**, which may be connected using SCSI adapters or their own adapter or may share another adapter. CD-ROMs provide access to interchangeable CDs, typically 600MB and containing data or multimedia information. CDs are usually read-only and have slower access times than fixed disks.
- Video adapters and displays, such as **VGA**, **XGA**, **SVGA**, and the emerging **accelerated SVGA**. Video adapters and display provide high-quality color bit-mapped graphics, typically up to 1024x768 pixels and 256 colors. Photo-realistic adapters and screens which can display up to 16 million colors are emerging, as are higher-resolution and larger displays.
- Printers, such as **Postscript** and **LaserJet**, provide letter-quality printing at low prices. Color printers are also starting to emerge.
- Multimedia devices, such as **digital audio** or **video** adapters, are able to record or playback high-quality sound or video. Video can also be played

back on standard high-resolution video adapters, if it is stored in an appropriate format.

Support for new hardware devices is provided by the device driver layer of the system software.

### 2.8.1.5 Distributed Systems

Today, PCs have matured and can be a major component in distributed systems. Their main benefits are the powerful user interfaces possible, and the computing power now available on the desktop dedicated to one user, or in a server shared between multiple users. However, PCs introduce major new problems in distributed systems, such as distributing and synchronizing both code and data across many thousands of nodes.

The motivation for involving PCs in distributed systems comes from three distinct areas:

- PC users want to share data, share resources (such as printers), and communicate with other PC users.
- Mainframe or mini users want to provide a better user interface, a faster response time, or local reliable access to data, while maintaining the advantages of the host for data security and integrity.
- Mainframe or mini users want to transfer their applications to a PC and LAN based solution (downsizing).

In the PC area, the primary interconnection method is the Local Area Network **(LAN)**, which provides very fast connection between PCs. Typically, most of the PCs will be end-user workstations, and one or more PCs will be used as servers, providing shared files, printers, and data. The main LAN technologies currently are **Token-Ring** and **Ethernet**.

On top of the LAN hardware technology runs the LAN software. **Novell NetWare** is the leading LAN software for IBM PCs and compatibles, along with IBM′s **OS/2 LAN Server**, Microsoft′s **Windows NT Advanced Server**, and **TCP/IP** which comes from a Unix background. NetWare uses a communications protocol called IPX, and OS/2 LAN Server and NT Advanced Server use the NetBIOS protocol.

When PCs need to communicate with other computer systems, such as mainframes, they can either do so directly if the system can participate in one of the above networks (for example, a Unix server can participate in the TCP/IP network), or they can provide an SNA or OSI connection into a Wide Area Network **(WAN)**. These connections can be for terminal emulation (such as 3270, 5250 or ASCII), shared data, and program-to-program communication.

Emerging technologies and standards will make it easier to interconnect PCs with other systems. The Distributed Computing Environment **(DCE)**, originated by OSF, provides standards for interoperation of both homogeneous and heterogeneous systems. This includes the Remote Procedure Call **(RPC)**, which can be used for developing distributed procedural applications.

The Distributed System Object Model **(DSOM)** technology enables distributed object-oriented applications and systems to be built, in a similar way to the use of RPC for building distributed procedural applications. DSOM is built upon the **SOM** object technology used for building mixed-language object-oriented

applications, and also upon the RPC technology used for building distributed procedural systems.

#### 2.8.1.6 Application Development

Today, development of GUI applications is slow, complex, and costly. This is due to the new application structure that GUI applications written in the C programming language must conform to, and also due to the complex APIs that GUI application programmers must use.

Development of client-server and distributed applications adds to the complexity of the whole application development process.

Despite this, both application development and system software development are unable to keep up with the demand for new systems and for modifications and enhancements to existing systems.

**Object-oriented** technologies are now emerging into the mainstream, and it is expected that they will provide powerful tools to simplify the software development process. Some of these technologies are:

- **C + +** , which is an object-oriented version of the C language
- **Class libraries**, which are the object-oriented equivalent of procedure libraries. They enable standard functions to be written once and shared by many applications
- System Object Model **(SOM)**, which provides a language-neutral and compiler-neutral way of enabling objects and classes to be shared between different applications
- **Visual builders**, such as IBM′s VisualAge and Microsoft′s Visual C++, enable applications with major user-interface components to be programmed interactively on the screen.
- **Object-oriented frameworks**, which are a development of class libraries enabling them to be easily used by application programmers, by collecting them in coordinated structures called frameworks, and by providing a rich set of default behavior.

## 2.9 Open Systems

The term *open systems* identifies a complex set of requirements that are constantly growing in the data processing industry. They are strictly associated with the industry trend towards distributed data processing environments that are multi-vendor and heterogeneous as far as both the processor′s architecture and the software platforms are concerned.

The following are just a few of the requirements for an open system:

- Data processing users want to access and share information across several data processing systems without being constrained by hardware, software, or architecture. The more the customer installation tends to be distributed and heterogeneous, the more critical becomes the open system requirement. The more the data processing technologies become vital to the enterprise activities, the more the relationships among enterprises are dependent on inter-systems communications or, in other words, on systems interoperability. The ideal solution to this requirement is a technology that allows any application to access and share its own local data and any type

of data residing on a remote heterogeneous system, while maintaining the same level of performance regardless of the location of the data.

- Data processing users want the freedom to choose the best solution that meets their requirements without being constrained by the existing hardware, software, or architecture. The ideal solution for this requirement is a technology that allows any application solution to run at its best performance on any type of data processing system.

Several definitions exist today for an open system or open-system environment because many independent bodies are working on definitions.
In addition, multiple versions exist for the definitions developed by the same body reflecting the long iterative work required to reach an agreement on matters covering multiple requirements.

For example, in July 1991, the IEEE** draft definition of *an Open Systems Environment*, as reported in *Guide to the POSIX Open Systems Environments*, was:

> "The comprehensive set of interfaces, services, and supporting formats, plus user aspects, for interoperability or for portability of applications, data, or people, as specified by information technology standards and profiles."

This definition is also known as the draft for the POSIX** 1003.0 standard.

In September 1991, that IEEE** draft definition for an open system had evolved to:

> "A comprehensive and consistent set of international information technology standards and functional standard profiles that specify interfaces, services and supporting formats to accomplish interoperability and portability of applications data and people."

IBM adopted that definition in the announcement of *The Open Enterprise*.

At the same time, X/Open also had developed a definition for open systems that refers to a:

> "Software environment designed and implemented in accordance with standards that are vendor independent and that are commonly available."

While waiting for the work on the definition to proceed and reach an agreement widely accepted in the data processing community, it is possible to focus on a few undisputed elements of open systems namely:

- Open systems is a software concern.
- Open systems require standards.
- Open systems are expected to provide interoperability among heterogeneous systems
- Open systems are expected to provide portability of applications, people, and data among heterogeneous systems.

We have already discussed the implications of standard programming interfaces in section 2.3.1, "Programming Interfaces" on page 8, and how the various types of standards available in the industry (de jure, de facto, national, international, and others) might be used to indicate the level of openness of a system. In the same place, we also discussed how standard programming interfaces affect the

portability of an application from one system to another. The portability of an application has an immediate affect on the portability of the data and people associated with that application.

An important consideration about application portability derives from the layered structure of software and applications already discussed in section 2.3, “Software Layers” on page 7. For an application to be moved or ported from its local environment to a remote heterogeneous environment, it is necessary to make the programming interfaces used by the application available in the remote environment. This might be accomplished in two ways:

- Porting to the remote location the application AND the products providing the required programming interfaces,
- Porting to the remote location ONLY the application, and having the remote software products provide the required programming interfaces.

For example, the requirement to port an existing application that makes use of the programming interface provided by transaction manager A, for example a CICS* application, from its native, or local, environment to a remote environment can be satisfied either by porting the application and the transaction manager to the remote environment, or by porting the application only and having the remote transaction manager provide a programming interface compatible with that of transaction manager A.

The example provides the basis for describing some of the directions taken by different institutions to satisfy the portability aspects of the open systems requirement. The following discussion is not intended to cover every organization involved in open systems standards and specification but only to show the different approaches to the problem.

The Technical Committee on Open Systems (TCOS) of the Institute of Electrical and Electronics Engineers (IEEE**) is dedicated to define a set of standard programming interfaces intended to satisfy the open system requirements. These standards are known as Portable Operating System Interface for Computer Environments (POSIX**). POSIX** standards are numbered from 1003.0 upwards, where 1003.0 is the definition of an open system.

Due to the quite complex and lengthy internal TCOS procedures, POSIX** standards are often quoted as being in the draft position, which means defined but not yet officially approved.

X/Open**, a nonprofit organization founded in 1984 to solve problems caused by software and systems incompatibilities, has also started to define standard programming interfaces. If POSIX** standards are available, they are adopted. Otherwise, the most widely accepted standard in the industry is selected. The suite of X/Open** standard programming interfaces define the X/Open** Common Application Environment** (CAE**) that is currently being implemented by the X/Open** members and other vendors. X/Open** CAE** is documented in an X/Open** Portability Guide updated and edited regularly, and commonly referred to as XPGn, where n is the number of the edition (XPG4 for the fourth edition). IEEE** standards for Open Systems and X/Open** standards tend to have a high degree of overlap.

The Open Software Foundation** (OSF**) is a non-profit organization that has taken a different direction towards open systems. Instead of defining standards, OSF** has the objective of developing an operating system, OSF/1**, and several

other software layers, or environments, based on existing programming interfaces that the OSF** members and the users accept as standards. The OSF/1** operating system is not written for any specific machine interface and must, therefore, be adapted by any interested hardware manufacturer to run on its hardware base. For example, IBM AIX/ESA* is based on the OSF/1** operating system.

Among the best known software layers implemented or in plan by OSF** is the Distributed Computing Environment** (DCE**), a set of standard interfaces for distributed computing.

Software vendors have also taken steps to address open system requirements. The most common approaches are:

- Defined public programming interfaces (as described in 2.3.1, “Programming Interfaces” on page 8) for unrestricted use by other software vendors like, for example, the Distributed Relational Database Architecture* (DRDA*) of IBM.
- Software products that can be executed in several software and hardware environments (for example CICS* and Oracle**).
- Standard interfaces in software products, in addition to the traditional proprietary interfaces (for example the POSIX** 1003.1 interface on MVS, the socket interface in CICS*, the SQL* interface in Oracle**, the selected OSF**/DCE** interfaces on the IBM software platforms, and others).
- Do both things at the same time. For example, CICS* and Oracle** both have taken the direction to provide multiple standard interfaces to applications and at the same time to be able to execute in more than one hardware and software environment.

Most of what has been said about portability also applies to the interoperability aspects of open systems.
In fact, the enforcement of standard interfaces on software products is also beneficial to interoperability if those standards also cover communications among systems.

An additional interoperability consideration also involves the layered structure of the software and the multiple options available to the application to operate with a partner system. For example, if interoperation is required to access some remote data from a local application, the following alternatives exist, at least in theory, to do the same thing:

- The two systems interoperate at the communication level (for example, the two systems both have SNA LU6.2 capabilities),
- The two systems can communicate at the transaction manager level (for example, the two systems both have CICS capabilities),
- The two systems can interoperate at the database manager level (for example, the two systems both have distributed RDBMS capabilities).

In the first case, the user has to write two applications (one per side) utilizing a *low level* interface, such as the SNA LU6.2 interface. In the second case, the user has to write two CICS* transactions. In the third case, the user has to write one query. All three options solve the user problem, but the implications of each alternative are extremely different.

# Chapter 3. Distributed Data Processing Technology

The subject of distributed systems is greatly simplified by using a single framework for discussion of all types of systems. The specific framework to be used in the remainder of this document is based on the Open Blueprint (which is illustrated in section 3.3.3, “The Open Blueprint” on page 50) and is described in Figure 40 on page 100. However, before examining this framework, we will discuss some prior examples of system structures. We will then examine in detail the *Open Blueprint* which is the base that will be used to develop the framework.

## 3.1 Early Data Processing System Structures

In early data processing systems, an application was responsible for providing all system functions. In the earliest examples, the programs were written directly at the machine level interface. Somewhat later, assemblers and compilers were developed to provide some simplification in writing applications. In either case, each customer had to develop applications with specific logic that was unique to each model of processor. If a customer had more than one application, each required processor-specific logic. These systems are represented by Figure 14.

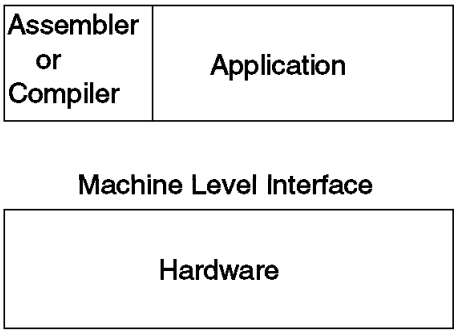


*Figure 14. Early Data Processing Systems*

This approach to application development was costly, and simply could not keep up with the rate of change in computing technology. Two major new concepts in system structure were developed to address these limitations: operating systems and processor architectures.

### 3.1.1 Operating Systems

Operating systems were developed to provide basic management of a processor and its resources, such as processor time, storage, and I/O equipment, so applications did not have to perform that task. As operating systems evolved, they allowed multiple applications to run at the same time. Further extensions to base operating systems were offered over time, and the collective set of services became the application programming interface (API) for that operating system. Examples of differing operating systems from IBM* are OS/2*, MVS/ESA* with CICS/ESA*, and VSE/ESA*. Examples of other vendor operating systems are DEC**, VMS**, and Microsoft** DOS with Windows**.

### 3.1.2 Processor Architectures

Processor architectures define a set of hardware functions that are common across several processor models. This allows an application to be isolated from the details of the differences between processor models. An early example of this is the IBM* System/360 architecture. Some other examples of processor architectures are the Intel** x86, IBM* RISC/6000* and AS/400*, and DEC** VAX**. In some cases, several hardware vendors offer processors that conform to a single processor architecture. These systems are represented by Figure 15.

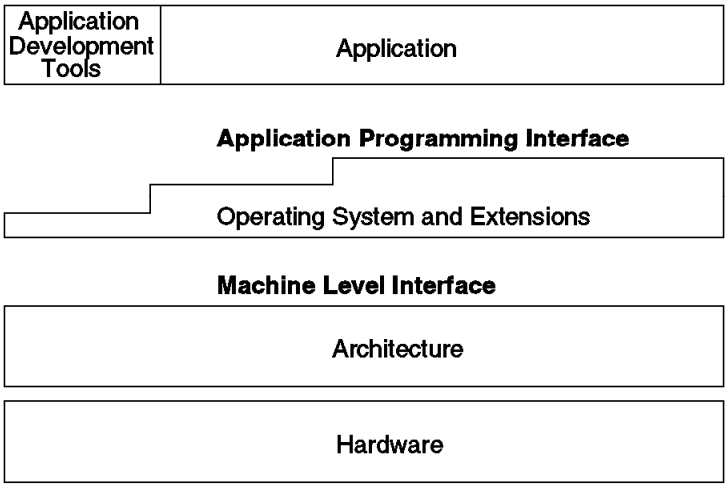


*Figure 15. Homogeneous Architecture*

## 3.2 Historical Heterogenous System Structures

Having operating system functions and APIs greatly simplifies application development. However, many manufacturers have their own set of proprietary interfaces and, in some cases, have more than one set of interfaces. Applications are required to adjust to variances between operating system interfaces. Standards have been developed to minimize the impact of differing operating system interfaces. Examples of this are ANSI** compilers, POSIX** operating system interface definitions, and X/OPEN** Portability Guides. The UNIX** systems and the more recent POSIX** 1003.x operating system interface definitions also address this problem. Nevertheless, many differences between systems are still visible to application programmers and end users. As computing technology changed, radically different types of structures developed, and millions of processors came into use.

### 3.2.1 Systems Application Architecture* (SAA*)

In response to the growing number of processor architectures and the increasing volumes of systems, IBM* developed SAA*. SAA* was intended to enable consistent APIs and end-user interfaces across:

- Heterogeneous operating systems:
  - OS/2*
  - OS/400*
  - MVS/ESA*
  - VM/ESA*.

  VSE/ESA* and DOS also offered a subset of SAA* functions, as do DEC** HP** and other vendors.
- Heterogeneous processor architectures:
  - PS/2*
  - AS/400*
  - System/390*.

  Other hardware vendors offer processors that are largely compatible with PS/2* and System/390* architectures.

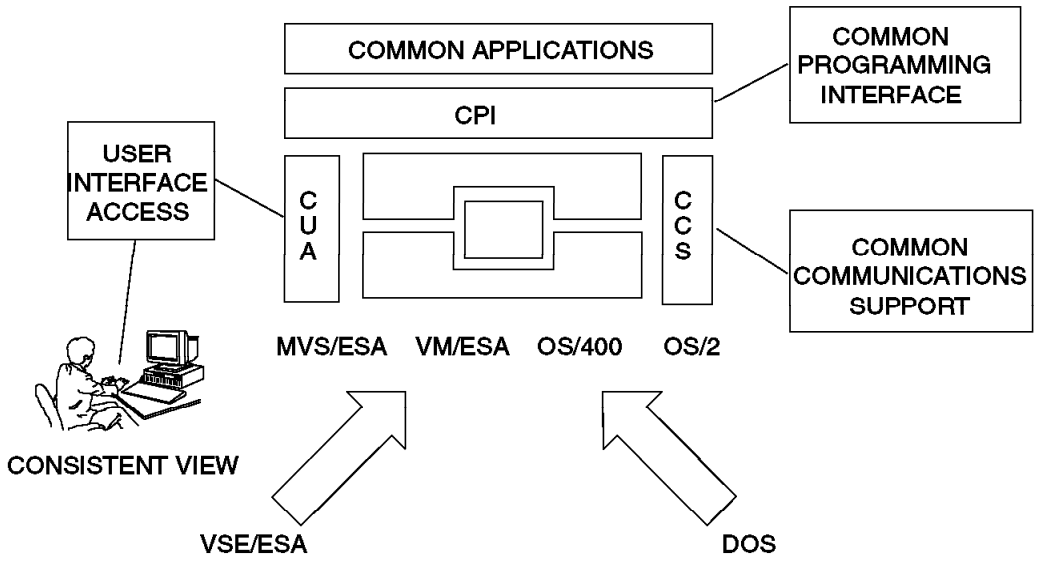


*Figure 16. Systems Application Architecture* (SAA*) Structure*

### 3.2.2 Advanced Interactive Executive* (AIX*) Family

IBM* also developed a set of UNIX** based AIX* systems that provide for consistent APIs and end-user interfaces across:

- Heterogeneous, but similar, operating systems:
  - AIX* PS/2*
  - AIX* Version 3 for RISC System/6000*
  - AIX/ESA*.
- Heterogeneous processor architectures:
  - PS/2*
  - RISC/6000*
  - System/390*.

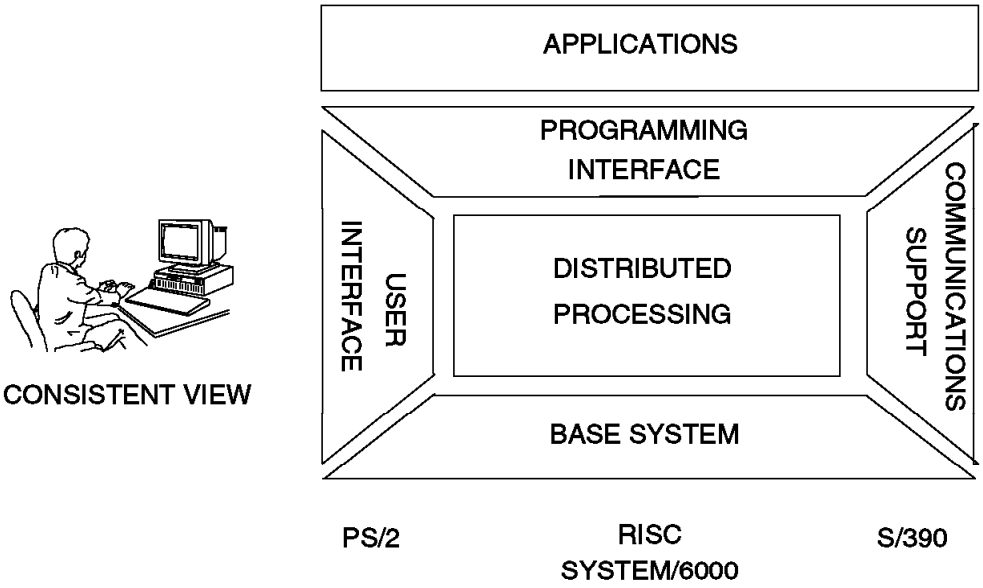


*Figure 17. Advanced Interactive Executive* (AIX*) Structure*

### 3.2.3 UI-ATLAS**

In contrast to SAA*, which covered several dissimilar operating systems, the UNIX** SV R4 family is heterogeneous but similar, as is the case with the IBM* AIX* family. The same basic operating system has been recompiled and modified (ported) to run on several heterogeneous processor architectures. The systems are similar, but not identical. There are also several other UNIX** type systems that have considerable overlap with SV R4, but they were not developed by porting SV R4.

The UI-ATLAS** structure, defined by Unix International, is intended to provide a common structure of higher level functions beyond SV R4 itself. They hope that this structure will be adopted by many UNIX** type systems, in addition to those based on SV R4. This structure is represented by Figure 18.

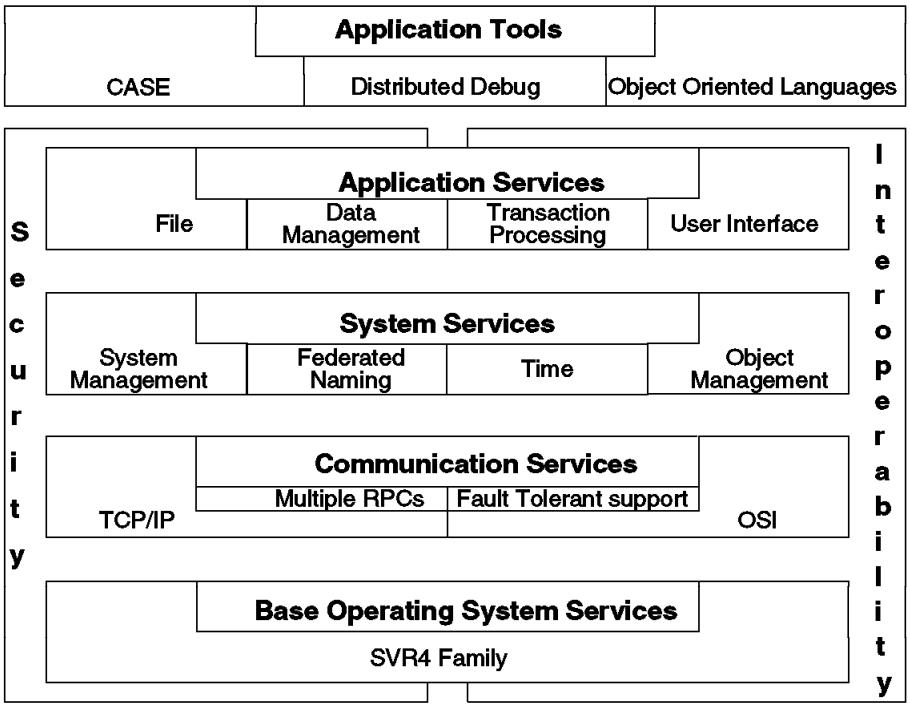


*Figure 18. UI-ATLAS** Structure*

## 3.3 Distributed System Structure Evolution

The realities of today's computing technology demand the ability to address a wide variety of operating systems and processors. Most customers require a broader scope than that addressed by SAA*, the AIX* family, or UI-ATLAS**. The IBM* response, over time, to these needs is shown in Figure 19 culminating in the identification of a requirement for an open, distributed system structure.

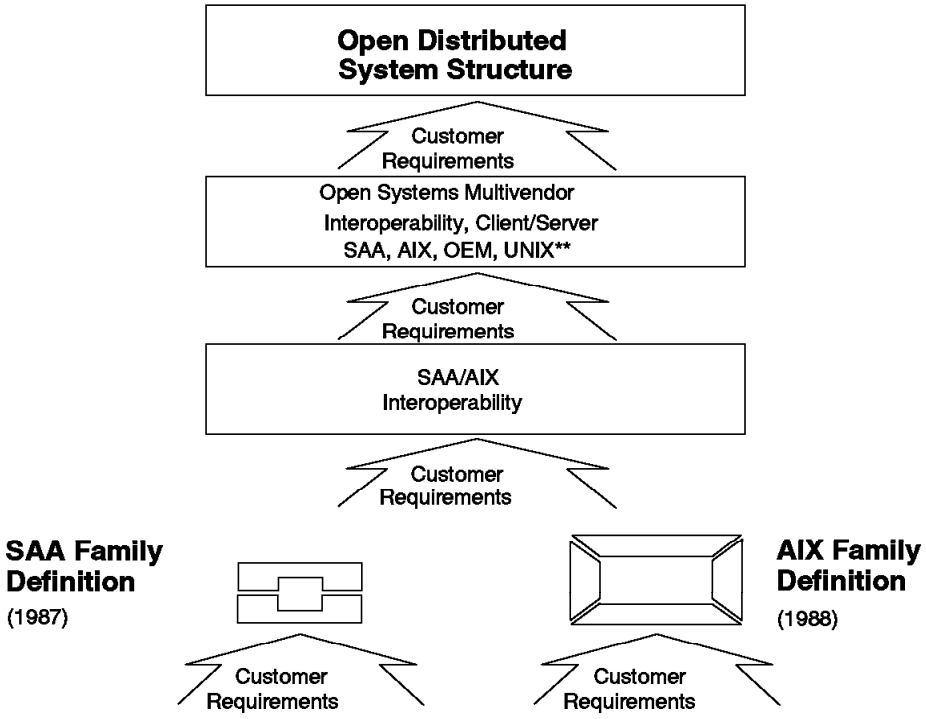


*Figure 19. Roadmap to a Strategy*

This roadmap has led to what is now known as the IBM* *Open Blueprint*. The characteristics leading to and associated with this structure are explored in the following section. Some definitions associated with these characteristics which describe an open, heterogeneous, distributed system are:

**Open** The system defines interfaces that are standard, relatively stable and publicly described. Users need to be able to choose elements that make up the system from various suppliers based on cost, performance, functions, packaging, terms and conditions, and other factors.

**Heterogeneous** Users expect to compose systems with offerings from many different vendors at the same time and may change the components that make up their system over time. Also, multiple systems may be connected together forming even more diverse configurations.

**Distributed** System elements are separated from one another and are connected by a communication network.

The structure must accomodate an evolution from existing system structures and product implementations.

### 3.3.1.1 Client/Server Terminology

The industry uses the terms client and server with both very specific and very general meanings, which causes considerable confusion in communicating structural differences. Our use of the terms is consistent with the industry, but we will define them carefully to avoid confusion:

- Roles

  In general, the terms describe roles that entities can play when a system performs some work.

  - Client refers to the entity on whose behalf the work is done.
  - Server refers to the entity that does the work.

  The terminology is useful when these roles are in different, and often, physically separate entities. Often, but not necessarily, the server provides services to more than one client, either simultaneously or serially, providing shared or multiplexed access by many clients to a single resource.

  The term client is not used just for the end user of work, but is applied to any entity that is requesting work. Therefore, a single entity often acts in the client role for some work and the server role for other work. An application program provides services to a human being, or its client, but is, itself, the client when it requests work of the operating system.

- Machines

  In some machines, particularly workstations used directly by people, the client role predominates. In others, the server role predominates. Such machines are often referred to as client machines and server machines, or physical clients and physical servers, or, confusingly, as just clients and servers.

- Resource manager

  When a resource manager is distributed, we describe the portion of the resource manager that does the work as the resource manager server and the portion on whose behalf the work is done as the resource manager client.

  Where necessary, we distinguish the entity that requests services by interfacing to the resource manager client by calling it the requester.

### 3.3.2 Distributed System Characteristics

Many of the characteristics of an open, heterogeneous, distributed system are essentially the same as those of any computing system. Appropriate performance for applications, high reliability, a flexible configuration, and system integrity are all desirable attributes of any system. But the nature of an open distributed system puts emphasis on certain characteristics that affect its fundamental design and specific implementation details.

The distributed system must be:

**Accessible** Users of a distributed system expect to see it as a single, coherent computing facility, with a single point of access and a unified logon. Within the distributed system, resources are accessible to users and to programs through a consistent universal naming scheme that is independent of location and method of access.

**Transparent** Functions should support well-defined, simple, functional interfaces, where implementation details are not apparent. End users and their applications do not see the overall complexity of the distributed system. They view the distributed system as an extension of their individual workstation.

Many different networking mechanisms are used to support the interconnection of distributed system. Programs in a distributed system are provided with several levels of communication mechanisms and interfaces that insulate them from the protocols or semantics of the particular network transport being used. This enables the transparent support of multiple networks.

**Scalable** The distributed system can accommodate a choice of configurations from relatively low cost and few functions, to many functions and high reliability and performance.

Scalability requires that there be no major design discontinuities as network size increases.

**Manageable** Systems management includes the planning, coordination, operation, and support of a heterogeneous network across a user′s enterprise. This includes monitoring systems functions, scheduling hardware and software changes, configuring system resources, and tracking system problems and resolutions.

Regardless of how systems management applications monitor and control system resources, the possible system administration actions should be available anywhere in the network.

As the size of a network system increases, the only acceptable solution to many systems management problems is to have systems that are self-managing and have automated logic to recognize and respond to exception conditions, failures, and changes in system load.

The geographically dispersed nature of large-scale distributed systems makes *hands on* management and operation of most resources impractical. In these cases, the tools for managing

and controlling the resources function entirely with remote operators and allow unattended operation.

***Additional Characteristics***

**Security** Security facilities are available across a distributed system, and mechanisms exist to permit user authentication, secure communications, information integrity and confidentiality, resource access control, security administration, and auditing of security events.

**Reliability and Availability** Large distributed systems that serve widespread populations must be continuously available to be business justified. The distributed system is designed to degrade gracefully during failures, and single points of failure of critical resources are avoided.

**Serviceability** In large distributed systems, the hardware and software resources are specifically designed to be serviceable. Built-in mechanisms are incorporated for monitoring, problem diagnosis and repair. In large-complex systems it is difficult or impossible to re-create problem situations. Therefore, system elements are designed to support the capture of such fault data as trace, logs, and dumps at the first point at which problems are detected.

**Accounting** The diverse population served by large-scale distributed systems requires that the cost of delivering services needs to be attributed to distinct consumers of the service. The elements of the distributed system maintain sufficient information to correlate processing to specific user activities no matter where the work is performed in a distributed system.

### 3.3.3 The Open Blueprint

A significant part of this chapter, including the previous section, is drawn directly from the Open Blueprint Technical Overview, GC23-3808. This has replaced a large part of the original material intended for use in this chapter. The ′Technical Overview′ is current, provides a valuable analysis of the issues, and a thorough description of the *Open Blueprint* which provides the descriptive base on which the body of this publication is constructed.

Some condensation has taken place in making use of the ′Technical Overview′ and the reader is requested to consult the original to fully appreciate its examination of the *Open Blueprint*.

#### 3.3.3.1 Introduction to the Open Blueprint

The Open Blueprint addresses the challenges of the open environment by viewing a system as part of a distributed network and viewing the network as if it were a single system. It is represented in Figure 20.

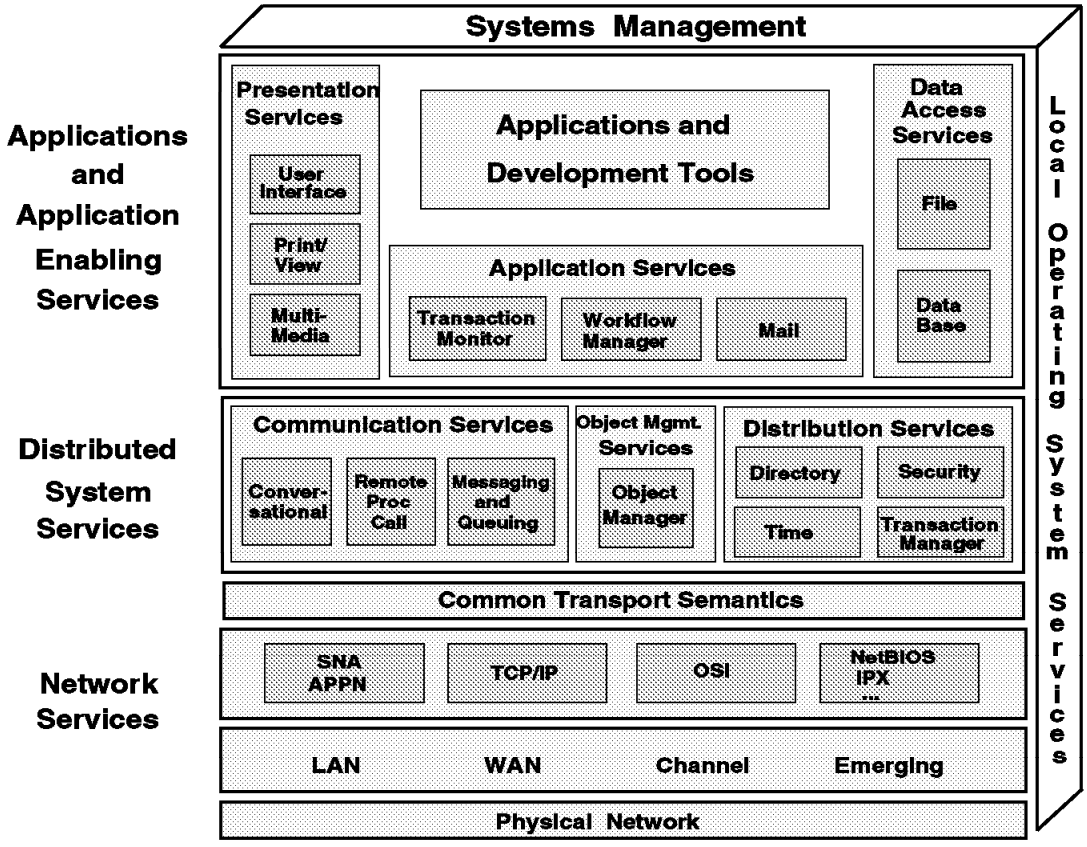


*Figure 20. The Open Blueprint*

The Open Blueprint serves four major roles:

- It helps users think about, discuss, and organize products and applications in an open, distributed environment.
- It describes IBM*′s directions for products and solutions in the open, distributed environment.

- It guides developers as they meet users′ needs by supplying products and solutions that include the appropriate function and that can be integrated and can interoperate with other installed products.
- It provides a context for the incorporation of new technologies into a distributed environment.

A goal of the Open Blueprint is to provide consistency among IBM products and related products such that they work together to achieve a high level of systemic value. Since the wants and needs of users include openness and product/vendor heterogeneity, the Open Blueprint is based on a combination of existing and emerging industry standards. The fundamentals that allow this support are heterogeneous network support as per the IBM Networking Blueprint; the security and directory protocols and usage as per the Open Software Foundation OSF** Distributed Computing Environment DCE**; participation in systems management as per the defined set of standard interfaces and protocols; and, for object-oriented implementations, adherence to the Object Management Group Common Object Request Broker Architecture (OMG CORBA).

It is a structure that enables a network of operating systems to function as a unit, as a *network operating system* comprising multiple systems separated from each other and connected by a communication network.

Just as an operating system provides the management of resources on a single system, a network operating system provides for the management across the network of the same types of resources: files, databases, printers, transactions, software packages, documents and jobs.

It promotes the integration of multivendor systems and simplifies the more cumbersome aspects of distributed computing. This integration improves the single system image that the end user and application developer perceive for the distributed system.

The Open Blueprint describes technical attributes and characteristics of supporting software, reflects desirable functional modularity, provides software principles and guidelines, and specifies important boundaries and interfaces.

Much of the function described in the Open Blueprint exists and is being developed and used in product form. Over time, the Open Blueprint will be expanded with additional function, and additional product implementations will be provided. However, it expresses technical direction only. None of it should be construed as a committment to deliver any of the functions described, nor should any inference to that effect be made.

In summary, the Open Blueprint is a structure that will help IBM and others deliver integrated, interoperable products and solutions.

### 3.3.4 Open Blueprint Concepts and Resource Managers

This section describes concepts and terminology that apply to open, heterogeneous, distributed systems and, in particular, to the IBM* Open Blueprint.

The dictionary definition of a system is “a regularly interacting or interdependent group of items forming a unified whole,” or “an organization forming a network, especially for distributing something or serving a common purpose.” In computing, the term is appropriate at many levels.

In this document, the term system is frequently used for smaller computing environments, such as single CPUs, CECs, multiprocessors, operating system images, clusters, and Sysplex*es.

#### 3.3.4.1 Resource Manager Concepts

The resource manager is the principal structuring element of the Open Blueprint. Resource management is a logical concept. Thus, a specific resource manager should be though of as a set of programs that maintains the state of a set of resources. Resource managers provide a set of formal interfaces through which operations may be performed on their resources. Resource managers support distribution with separable support for client and server functions. Only resource managers can directly access the resources they control; that is, they encapsulate access to their resources. Resource managers request services from other resource managers through their functional interfaces. In order to achieve proper integration, resource managers use Open Blueprint networking, security and directory services, and object management services.

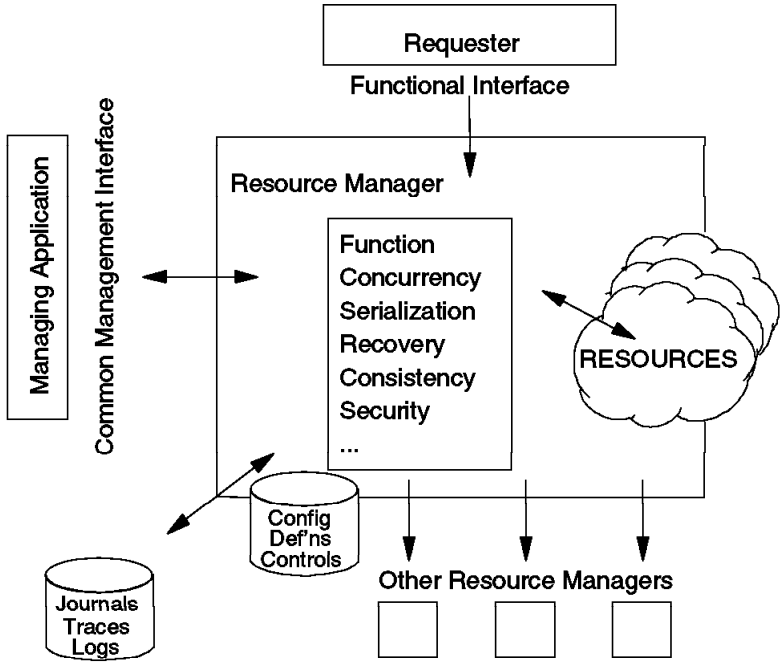


*Figure 21. Resource Manager Characteristics*

Resources may be distributed and replicated across many systems in the network. A file system, print server, and database manager are examples of typical resource managers.

Figure 21 on page 52 depicts a schematic representation of a resource manager.

***Resource Manager Interfaces:*** Resource managers provide programming interfaces for the operation, control, and administration of their resources. Programming interfaces support all required resource manager capabilities; no operations require human intervention.

Functional interfaces are either application programming interfaces or protocol boundaries.

- Application Programming Interfaces (APIs) are well-defined and portable interfaces that are used by user and vendor-written application programs, and by other resource managers. In the last case, these functional interfaces are sometimes called system programming interfaces.
- Protocol boundaries are well-defined interfaces for which only the operations and information passed in the interface are defined, and in which the syntax is implementation-dependent. Protocol boundaries are generally supported where performance demands override portability requirements.

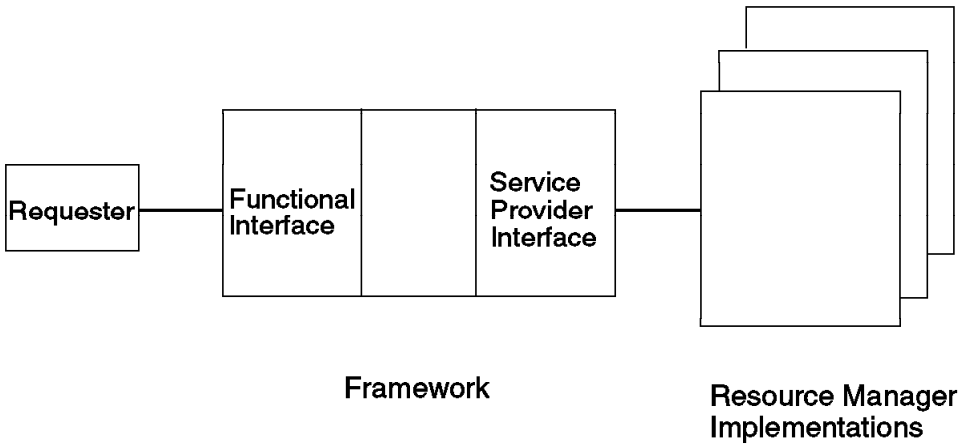


*Figure 22. Resource Manager Interface Frameworks*

In the Open Blueprint, the resource manager interfaces can be structured as a framework. Frameworks provide a mapping from the functional interface (an API or protocol boundary) to a service provider interface. The framework permits a particular implementation of a resource manager to be replaced without changes to the program that uses it. Frameworks support heterogeneity by allowing different implementations of resource managers that support the same service

provider interface to exist at the same time. Figure 22 depicts a schematic representation of a resource manager interface framework.

The service provider interface defined by a framework should be a distributed interface so that the service providers can be in different systems than the requesters. This makes it possible to support configurations where a minimum amount of function is required on a small, lightweight hardware platform.

Currently, resource manager frameworks are based on traditional procedural interfaces, but they will evolve over time into object frameworks. The evolution of these object frameworks will be consistent with the emerging industry standards, including those from the Object Management Group (OMG).

***Resource Managers and Systems Management:*** Resource managers support management functions by:

- Defining their management functions and externalizing those functions through the management interface, so an external entity can monitor and control their function
- Exploiting the common management services of systems management.

In the management structure, the resources managed by a resource manager are managed objects. The resource manager, itself, is a managed object. Management operations on this object include: resource manager initialization and termination, restart, work prioritization and control, accounting, problem determination, tracing, configuration management, and performance tuning.

Some of these operations require information that does not flow through the management interface, such as journals, logs, trace files, and configuration tables, but may use other systems management services. See section 3.4.5, “Systems Management Services” on page 90 for more information.

***Resource Manager Distribution Support:*** A resource manager that operates in a single system is a local resource manager. A distributed resource manager operates across multiple systems. Distributed resource managers include parts that support the interface that requesters use, called the client parts, and parts that perform functions on resources, called the server parts.

The client program may do some of the processing of the request (for example, validation), and is responsible for determining what instance of the resource manager′s server code should process the request. The client program supports multiple protocols as needed to deal with a heterogeneous environment.

The client and server communicate through an agreed protocol, using one of the interprocess communication services or the distributed object management services supported by the Open Blueprint.

A server program of a resource manager may process a request entirely itself, or it may transparently access other instances of its server program through a server-to-server protocol.

It is expected that different implementators could supply the client and server parts of a resource manager. Thus, functional and systems management protocols are based on standards. Also, each implementation would use the fundamental Open Blueprint facilities that enable integration, interoperability, and a single-system image.

***Resource Manager Characteristics:*** Resource managers typically maintain state information for requesting applications and other resource managers across invocation. They must maintain the consistency of the resource state information with underlying system state information.

Resource managers can access environment state information that represents their requesters. They may, if suitably authorized, change the environment state information. Communication resource managers support the distribution of the environment state information by passing it between systems when interprocess communication occurs.

Resource managers support the serialization and concurrency control needed to allow multiple requesters to use their resources. This may require multithreading within the resource manager.

Resource managers manage the integrity, consistency, and reliability of their resources, including recovery from physical and logical damage. Recovery from logical damage involves coordination and synchronization with other resource managers through the use of transaction managers.

Resource managers employ the necessary security mechanism to protect resources and information from unauthorized use or unintended disclosure. The access control resource manager provides this function based on information provided by the identification and authentication resource manager. It is the responsibility of the resource manager that owns the resource to determine when the check is to be performed, and the granularity of the resource and operation to which the check applies.

A resource manager′s responsibility for problem determination includes detection of failures and capture of relevant failure data when a failure first occurs.

***Resource Manager Relationships:*** Resource managers typically depend on other resource managers for the provision of services. There are two types of relationships between resource managers:

- Those that are dictated by the structure because they have structural significance.

  For example, resource managers are required to depend on the directory to present a single namespace to the user, and on the transaction manager to present a single scope for a logical unit of work.

- Those that are an implementation convenience and are of no structural significance.

  For example, a particular product implementation of the directory may choose to store its information using the relational database resource manager. This is transparent to the rest of the distributed system.

### 3.3.4.2 Protocol Layering and Gateways

Each program in a pair of programs that work together must have some understanding of how the other program operates. That is, they must define the syntax and semantics of the parameters and responses passed between them. This definition and its encoding is called protocol.[2]

Protocols are typically nested or contained within each other. For example, the functional protocol used between two arbitrary resource managers is nested inside the communication protocol supported by the communication services they have chosen to use. Likewise, the communication services protocols are supported on the transport protocols that are supported by the network services resource managers. Figure 23 illustrates the protocol layering and the need for the protocols to match at each layer.

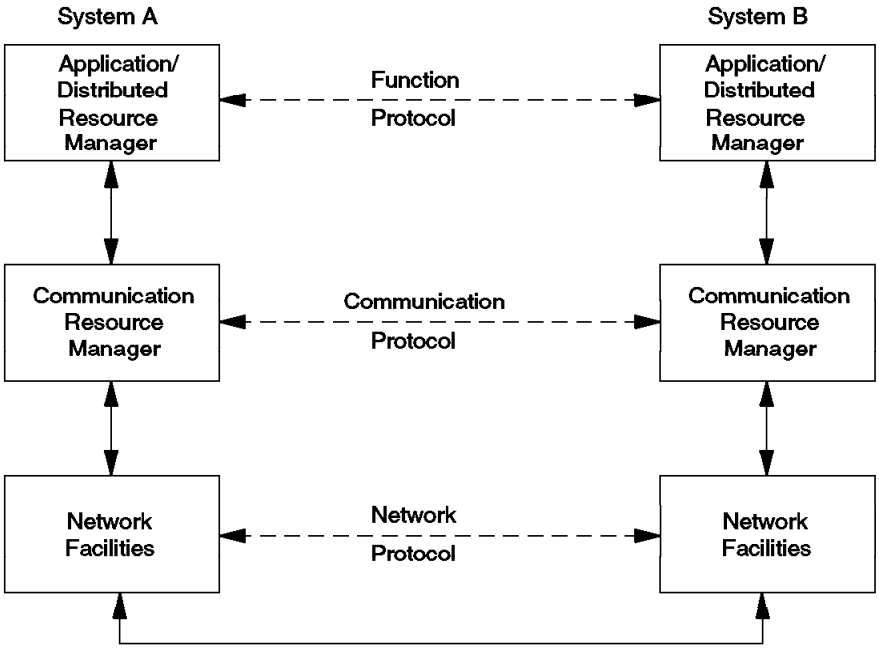


*Figure 23. Protocol Layers*

Normally, when two components communicate, they must use common protocol. Sometimes, a gateway is used to support existing programs that provide similar function but do not use a common protocol. Gateways support this interconnection by converting or translating the protocol of each program into the one expected by the other. While gateways can be implemented at any level, a common gateway usage is at the network services level to allow communication across heterogeneous transport stacks. This is shown in Figure 24 on page 57. Gateways are a good way to accommodate heterogeneity, but they are not simple and can be expensive. The system that provides the gateway must include both of the transport stacks and the code to do the conversions.

[2] The term *protocol* in this document refers to the common architecture usage of *formats and protocols (FAPs)* for the encoding (formats) and the way the encoding is used (protocols).

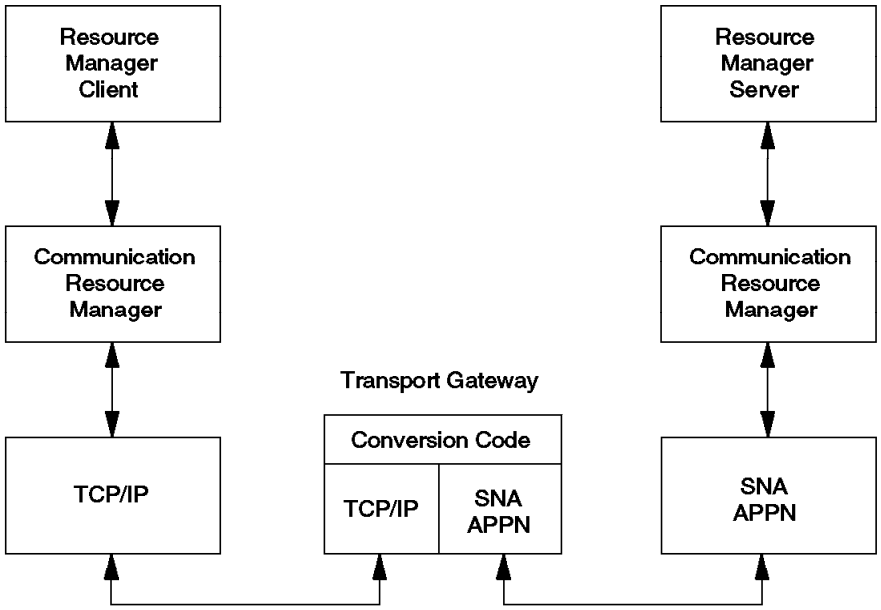


*Figure 24. Role of a Transport Gateway*

### 3.3.4.3 Platforms

An operating system and a set of resource managers constitute a platform that is part of the distributed system. Platforms can be configured in many different ways. The Open Blueprint does not demand that any particular set of resource managers be available, beyond that required by technical prerequisites. We will examine a number of IBM* platforms in section Chapter 6, “IBM Software Platforms” on page 135.

Packaging and licensing requirements may limit the configuration flexibility, but the following items influence the definition of platforms:

***Prerequisite Relationships:*** Every system in the distributed system must have network services with at least one network driver. For every system it wishes to communicate with, the system must share a common driver type, or share a common driver type with another system that contains a transport gateway to the desired end point.

The communications resource managers have a prerequisite of a transport network resource manager. They use the directory client and the security client, which depend on the time client. If each of these uses RPC to communicate with their respective servers, directory, security, time, and RPC would be a corequisite set.

The database, file, print, and transaction resource managers have prerequisites of either the conversation resource manager or RPC. Object-oriented resource manager implementations require the object management resource manager.

***Client Platforms:*** Depending upon application support requirements, systems can be configured to contain primarily the client parts of distributed resource managers. This would be a typical configuration for an end-user workstation. The framework for resource managers used by applications must be on the same platform as the applications. Smaller systems, like mobile computers or personal digital assistants (PDAs), could be configured to contain only a few resource manager frameworks and requisite communications support.

***Server Platforms:*** Some platforms in the distributed system are likely to be configured as specialized servers. Only the server part of a specific resource manager, such as the file resource manager, together with the client parts of the resource managers it depends upon, would be present. For an Open Blueprint distributed system to be functional, the server parts of certain critical resource managers (like directory, security, and time) must be accessible somewhere in the network.

***Platform Integrity:*** The distributed system platforms provide integrity when content control is maintained, that is, as long as content control of the systems is exercised by some authority to preclude unauthorized tampering. Users will be precluded from gaining a level of privilege beyond that specifically granted through security administration. This is not necessarily an easy job. Although it is desirable to offer autonomy to workstations that have hardware sufficient to guarantee integrity, it is impractical to require all workstations in the distributed system to have such hardware, particularly given the installed base of personal computers, most of which lack sufficient hardware. Therefore, a system installed on such hardware is content-controlled only through customer procedures.

Existence of viruses on personal computers is ample evidence of the inadequacy of such procedures. This leads to the requirement to support untrustworthy workstations, which does not reduce the exposure on the workstation, but reduces the exposure of the rest of the distributed system to the risks on the workstation. It is an example of the requirement that resources on secure systems must not be exposed by exposures in other systems. The IBM* distributed system platforms support IBM*′s system integrity guidelines.

***Structure for Heterogeneity:*** The Open Blueprint supports a heterogenous environment. It is a structure that divides system functions into distinct components. Each component is required to handle any diversity applicable to its function. Component interfaces are frameworks that are opaque to the variability inside the component. Each resource manager is defined so that it can support code implementing several protocols that achieve essentially the same function.

The interfaces are also a binding point. A binding mechanism is needed to select the appropriate implementation, depending on the actual circumstance of the interoperation.The directory plays a key role in determining the correct implementations to be bound together. The process for selecting implementations is open, in that the additional implementations to support new protocols are possible by third parties.

These interface frameworks are the key elements of the structure′s heterogeneity support. Figure 25 on page 59 emphasizes the support for multiple protocols.

Using such techniques, selection of a single specific protocol for a resource manager, such as the selection of OSF** protocols, becomes only a decision on

staging and investment priority. It does not become a commitment to interoperation solely over that single protocol. It does not require universal acceptance of that standard for the system to be successful in the open, heterogeneous, distributed environment.

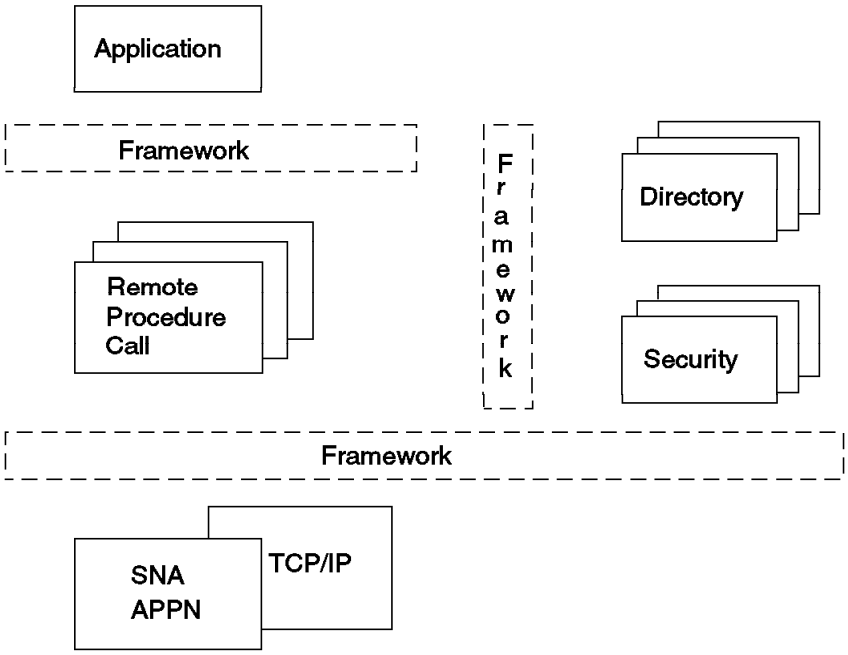


*Figure 25. Structure to Support Multiple Protocols*

## 3.4 Resource Manager Elements

The following sections describe the Open Blueprint resource managers as shown in Figure 20 on page 50. References are also made to technologies, models, and standards described in the Open Blueprint.

### 3.4.1 Network Services

Communications and networking are at the heart of the infrastructure for a distributed system. In the more homogeneous world of the 1970s and early 1980s, the communication requirements drove the structure of the application-enabling services and subsystems. This typically resulted in tying the enabling services and subsystems to particular communication structures.

In today's world of heterogeneous, distributed computing, the higher-level services and resource managers of the distributed system must support multiple operating system platforms and a variety of networking environments. At the same time, the higher level resource managers need program-to-program communication services that are suitable for their particular distribution model. However, they cannot afford to be tied to specific networking protocols or data link protocols. This led to the structural separation of communication services and other transport users from network services, as shown in Figure 20 on page 50.

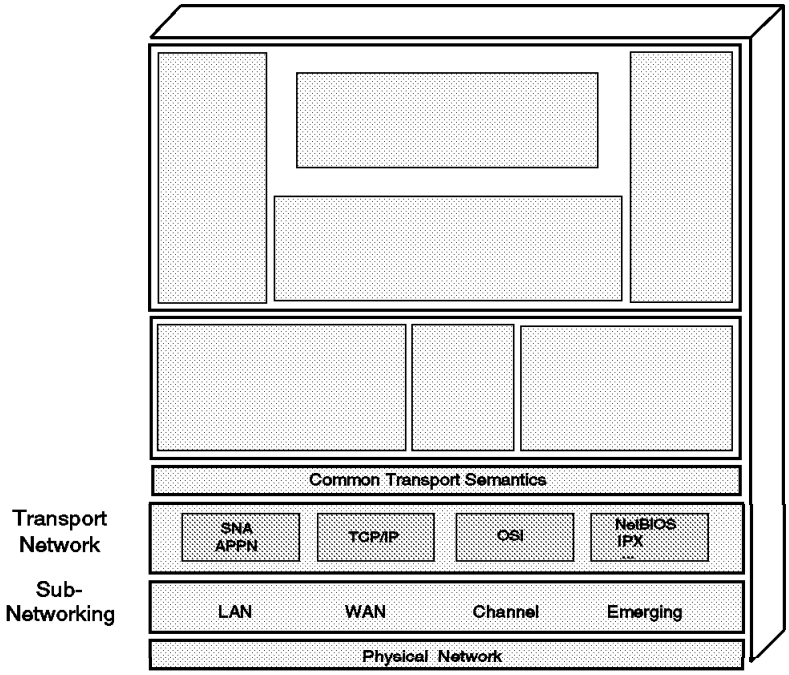


*Figure 26. Network Services in the Open Blueprint*

In 1992, IBM introduced the Networking Blueprint, its road map for networking in open, distributed systems, which focuses on the networking services, distributed services, and systems management portions of the Open Blueprint. The Networking Blueprint recognizes the need for structurally unifying network services by providing a common view of transport semantics, to make higher-level distributed systems services and application enabling services independent of the underlying transport network. This leads to the structure shown in Figure 26, in which network services consists of common transport semantics, transport network services, and subnetworking services.

#### 3.4.1.1 Common Transport Semantics

Common transport semantics (CTS) insulates the higher-level services from the underlying transport network (TN) by providing a common view of transport protocols. This common view, coupled with a standard set of compensation mechanisms, enables all higher-level services to be transport-independent, so that different transport network drivers can be plugged in under a common implementation of those services.

Using common transport semantics also enables the integration of networks with different protocols through transport gateways (see section 3.3.4.2, “Protocol Layering and Gateways” on page 56). As shown in Figure 27 on page 61, gateways provide compensation logic, where needed, to account for differences in the capabilities of the underlying transport network. For example, this enables the interoperation of client workstations without regard to which LAN media protocol (such as Token Ring or Ethernet**) or which LAN transport protocol (such as IPX, NetBIOS, SNA, or TCP/IP) is being used on the particular workstation.

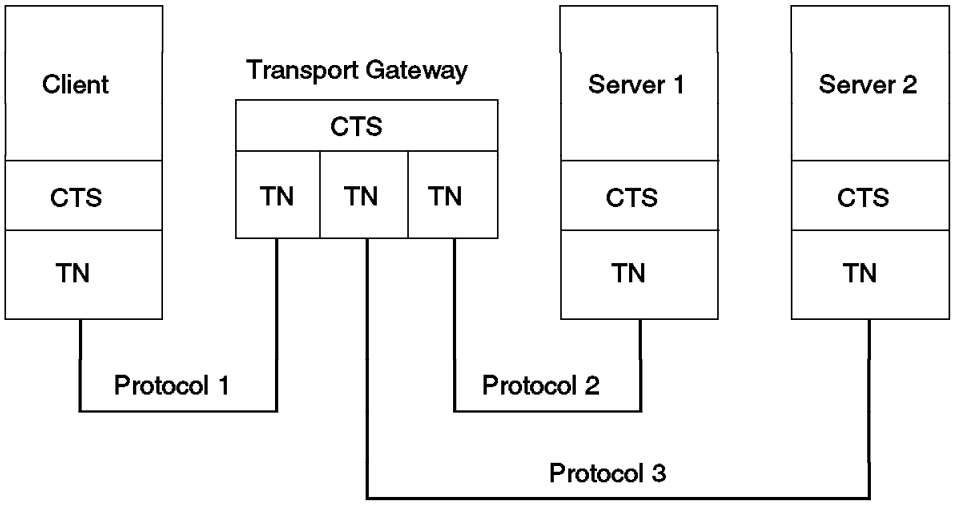


*Figure 27. Multiprotocol Transport Gateway*

The transport layer protocol boundary (TLPB) defined by the Multiprotocol Transport Networking (MPTN) architecture and implemented in AnyNet products, is at the top edge of the common transport semantics. Sockets, CPIC, and RPC interfaces can access the multiprotocol transport network through the TLPB.

The TLPB protocol boundary enables higher-level communication services (such as RPC or Conversational) to support multiple transport protocols transparently[3]. It also gives applications and higher-level resource managers the choice of achieving transport independence by using the TLPB directly, where appropriate, or by using the higher-level communication services.

The X/Open Transport Interface** (XTI**) provides access to SNA/APPC, NetBIOS, OSI, and TCP/IP, but does not shield applications or services from network differences. IBM has developed AnyNet products (based on MPTN architecture) that shield requesters from network differences. IBM has submitted the MPTN architecture to X/OPEN.

MPTN architecture includes two transport models:

**Connection-oriented**

The transport system is aware of a series of interchanges between the transport end-points. It detects and (if possible) corrects lost, out of sequence, or duplicated packets of data.

The connection-oriented model enables the transport system to optimize certain elements, and frees the using program

[3] This means that the using system may continue to use its native form of network addressing (such as SNA, NetBIOS, or INET) though the TLPB.

from having to deal with certain exceptions (such as detecting out of sequence packets and reordering them).

**Connectionless**[4] The transport system regards each packet of data as independent, leaving it to the program using the transport system to detect and correct lost, out-of-sequence, or duplicated packets.

The connectionless model enables “one-shot” communication without the overhead of connection setup. It may enable the using program to combine the detection of packet sequence errors with more efficient correction semantics than those which could be applied within the transport system. It also permits other delivery semantics, such as multicast (sending a packet to many receivers).

The difference between these models is *not* transparent to the program using the interface. The program explicitly uses one model or the other. However, these differences can be masked from applications by their using higher-level application services or communication services.

### 3.4.1.2 Transport Network

The Open Blueprint supports a variety of network protocols for transporting information over both wide area and local area networks. These include:

- Systems Network Architecture/Advanced Peer-to-Peer Networking (SNA/APPN)
- Transmission Control Protocol/Internet Protocol (TCP/IP)
- Open Systems Interconnection (OSI)
- NetBIOS
- Internet Packet Exchange (IPX).

Each protocol supports interfaces used to access its services. Also included are various end-to-end network monitoring functions that safeguard data integrity and help to avoid congestion.

### 3.4.1.3 Subnetworking

Subnetworking provides a structure to let networks evolve to accommodate and exploit new high-speed, highly-reliable transmission technologies without sacrificing business application and network investments.

Subnetworking includes three major types of network connectivity:

- Local Area Network, for example, Ethernet, Token Ring, and FDDI
- Wide Area Network, for example, HDLC, SDLC, X.25, and ISDN
- Channel.

Each type offers a unique set of configurability, connectivity, and performance options at varying cost levels. This is a rapidly changing field, in which major extensions to existing technologies are frequent.

In addition, there are a number of rapidly-emerging technologies, such as wireless communication facilities (which are key to mobile computing), asynchronous transfer mode (ATM) facilities based on high-speed frame-relay or

---

[4] Another widely-used term is **datagram**. For example, the *Internet datagram* is the basic unit of data transfer under the *Internet Protocol*, and the fundamental Internet service is the *connectionless, packet delivery system*.

cell-relay technologies, and very high-speed Synchronous Optical Network (SONET) technologies.

This structural separation of subnetworking services and transport services allows customers to separate the choice of transmission technologies from the choice of networking technologies and protocols, and to optimize each decision on its own merits.

## 3.4.2 Distributed System Services

Distributed systems services provide the communication services, object management services, and distribution services needed by higher-level resource managers to enable the commonly-used models of client-to-server and server-to-server distribution.

### 3.4.2.1 Communication Services

The communication services support three common distribution models. Each model describes how distributed parts of applications or resource managers communicate with one another. They are:

- Conversational
- Remote Procedure Call
- Messaging and Queuing.

***Conversational Model:*** In the conversation model, the distributed parts "converse" with one another and are synchronized in a manner similar to the speakers in a telephone conversation. This model is based on the program-to-program communication model defined by SNA APPC, where one part of a distributed application or resource manager initiates a conversation[5] with another, and they then exchange synchronous messages until the user′s requests are satisfied, at which time the conversation is terminated. Each part of the distributed application or resource manager is responsible for maintaining the state of the conversation and abiding by the rules of the conversation protocol.

The conversational model provides a synchronous service. Applications that use the conversational model include distributed transaction processing, distributed relational database access, and bulk data transfer operations involving multiple transmissions.

ISO has chosen the conversational model as the basis for the OSI Transaction Processing protocol specification, which is based on the SNA APPC architecture. X/Open′s Common Programming Interface for Communications (CPI-C) provides a common interface for the implementation of the conversational model on all major platforms for access to both LU6.2/APPC conversations and OSI dialogs.

***Remote Procedure Call Model:*** In the RPC model, one part requests a service from the other part, and awaits a reply. It provides programmers with a familiar model. Most programming languages support a CALL, and most programmers are familiar with obtaining services by calling routines from a subroutine library. The familiar concept of structuring an application into sets of services and users of the services is extended to a distributed environment.

---

[5] If a session had not been established previously, this must occur first. This involves establishing a logical connection with the distributed application or resource manager at the target server.

With RPC, a client program includes a call stub that packages the arguments of the call, sends them to the server program, and waits for a reply. A companion server stub unpacks the arguments, invokes the called procedure, packages the results, and sends a reply back to the client.

RPC has a mechanism for placing (exporting) the definitions of service interfaces into the directory. It includes a mechanism for operation across machines with different architectures, which is supported by the stubs. The stubs themselves are generated by an Interface Definition Language Compiler during the application development process.

The Open Software Foundation (OSF) chose RPC as the fundamental communication model for the Distributed Computing Environment (DCE). The DCE technology for RPC supports connection-oriented and connectionless transports. Because of the richness of the DCE technology, it was selected as the basis for RPC services.

The RPC model is synchronous from the point of view of the calling program, because it must wait until the requested procedure finishes its execution and returns the results. Applications include engineering and scientific applications that use RPC to invoke remote, high-performance computing systems, and applications based on the OSF Distributed File System (DFS). Transactional processing applications are also being developed based on transactional extensions to the RPC protocols.

***Messaging and Queuing Model:*** The messaging and queuing model is characterized by independent execution of the partner programs. The partners communicate indirectly through message queues, which are analogous to mail boxes, at the sending and receiving locations. The communicating programs do not have to be active simultaneously. Messaging and queuing support provides time-independent communication, with the sender and receiver executing at their own pace.

The sending application uses the message queue interface (MQI) to place a message on a queue at the sending system. In the receiving system, the message is placed on a queue for the receiving application. There can be a single queue, or separate queues for different types of messages. Multiple messages (including multiple instances of the same message) can be processed. After processing, the receiving application can generate a message to be returned to the sending application or to be forwarded to another application. Distributed applications may be created by arranging the flow of messages between serially-reusable message processing programs.

Messaging and queuing provides assured, once-only delivery of messages to the queues at the receiving system, and can optionally start the target process or transaction to handle the message. The assured delivery allows work to be committed by the sender, knowing that the message will be delivered despite system or network failures during the delivery process. This model is particularly well-suited to high-volume, networked transactional applications, in which numerous connections cannot be tolerated.

***Selecting a Communication Resource Manager:*** Although it is likely that almost any distributed function could be implemented using any of these three communication models, some applications or resource manager requirements may fit one style better than the others.

Developers of each application or resource manager must choose among the models in the context of their own requirements. Some of the criteria for choosing among the three models are:

- The need for real-time synchronization between the partner programs
- The need to communicate between programs that may not be active simultaneously (for example, due to different operating schedules)
- The need to control the flow of communication and resource synchronization
- The need to keep the communication flow and resource synchronization hidden
- The need to communicate, where the calling program is not blocked once the communication is initiated
- The need for request/reply-based processing, where the calling program is blocked until it receives a response/reply from the partner program.

Some resource managers may determine that one model is well-suited for certain functions, but not others, leading to the use of multiple models

***Communication Resource Managers Coexistence:*** Because it is sometimes necessary for an application or resource manager to use multiple communication resource managers, they must coexist. Examples are:

- A client that issues requests for different services, one accessed through RPC and another through conversations
- A server that supports both clients using RPC and clients using conversations
- A server that is invoked using messaging and queuing, and uses RPC during performance of the service.

#### 3.4.2.2 Object Management Services

Objects provide a way to create parts of applications by associating data with the programs required to access and maintain the data. These self-contained units (objects) afford unique opportunities for development efficiency through improved flexibility and reuse of the implementation. In a distributed context, objects are useful units for portability and transparent distribution.

In object technology, basic mechanisms support the actions of a using program invoking an operation on a target object. These mechanisms support the operation call in such a manner that a number of technical characteristics of object technology, such as encapsulation, inheritance, and polymorphism, are supported.

The object manager incorporates IBM′s System Object Model (SOM) and related distributed SOM technologies[6] that address the inherent interoperability problems of objects. Since each object-oriented language has a unique set of object characteristics or a unique object model, it has not been possible to write parts of an application in one language (C++) and use these parts from another language (Smalltalk**). Different compiler implementers for the same language (C++) have implemented different foundation mechanisms. The objects

[6] The initial delivered support for this function is named DSOM.

contained in a binary module compiled by one compiler cannot be used by a using program compiled by another compiler.

SOM provides a rich, language-neutral object model that supports binary interfaces defined at the system level and which are independent of a particular compiler implementation. The SOM object model maps easily into the popular object-oriented programming languages and the 3GL language base used in existing systems.

Distributed object support associated with SOM provides a complete implementation of the industry standard Object Management Group′s Common Object Request Broker Architecture (CORBA). Because of the seamless integration, CORBA′s standards apply equally to both local and distributed objects. The object manager provides CORBA-compliant support for distribution of objects.

A key advantage of SOM and distributed SOM is location transparency. The same mechanism that provides access to local objects is extended to access remote objects. As illustrated in Figure 40 on page 100, the object manager operates above the common transport semantics in the distributed system so that this technology is available to configurations involving a variety of transports. The distribution services are used so that this object service operates as an integrated part of the overall network. This includes using the distributed directory to access information about target objects, the security services to authenticate users, and the transaction manager to synchronize object operations with other resource requests.

A variety of system and language vendors are working with IBM to provide implementations across a range of operating system platforms.

#### 3.4.2.3 Distribution Services

***Naming:*** A major goal of the distributed system structure is to achieve a seamless, single-system image across a heterogeneous collection of systems. To accomplish this, a consistent approach to naming must be established across all resources of the distributed system.

Today′s operating systems frequently define one or more namespaces unique to the system. These namespaces may be defined by operating system convention or shaped by specific resource managers. In an open, heterogeneous, distributed environment, the potentially large numbers of resource types and implementations can create a complex array of naming conventions, with unique syntax and approaches to context.

***Universal Naming:*** To simplify the tasks of using, administering, and writing applications in an open, heterogeneous, distributed environment, the Open Blueprint includes a universal namespace based on the structure implemented in OSF′s DCE technology. With universal naming, resources of any class, such as programs, hardware, data, and users, in any location can be referred to by a name that follows a single set of naming rules.

The universal namespace includes two distinct subnamespaces: the global namespace and the local, or cell, namespace. Global names use the ISO X.500

or the Internet[7] naming standard. Cell names follow the Cell Directory Service (CDS) naming conventions (untyped) as defined by DCE. Resource names can be added in either namespace. Cell names refer only to resources within a single cell. To access resources in a “foreign” cell, the global name of the cell is combined with the cell-relative name. The global name of the cell can be either an X.500 or Internet name. Figure 28 shows an example of each of these universal naming formats.

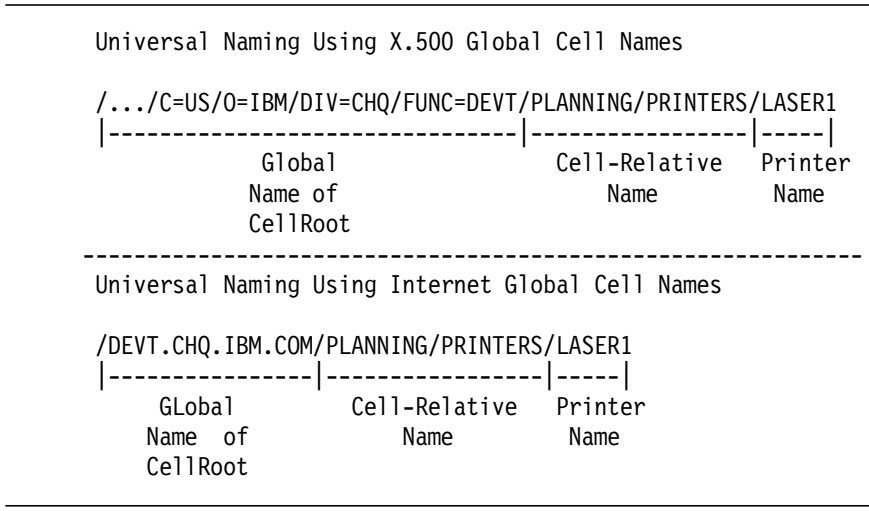


*Figure 28. Universal Naming Examples*

Over time, new resource managers, APIs, user interfaces, tools, and application programs will support the universal namespace. Existing system and resource manager-defined namespaces must, of course, continue to be supported. Resources may be referenced either by their universal name or through a resource manager-defined name.

***Concatenated Naming:*** In addition to the disparate syntax and structure of today’s resource manager namespaces, each is defined within a name (and usually administrative) scope established by that particular resource manager. For example, the scope of resource manager names may be limited to a single instance of the resource manager implementation, a single-system image, a single user, or a single collection of instances of the resource manager implementation.

Open Blueprint concatenated naming is an approach to achieving greater uniformity of these diverse namespaces by “plugging” existing resource manager unique namespaces into the universal namespace (using a technique called “junctions”). With concatenated naming, a resource name is formed by appending a resource manager-specific name to a universal name. Figure 29 on page 68 shows two examples of concatenated naming.

[7] Internet names are handled by the Domain Name Service (DNS), which is a standard TCP/IP application.

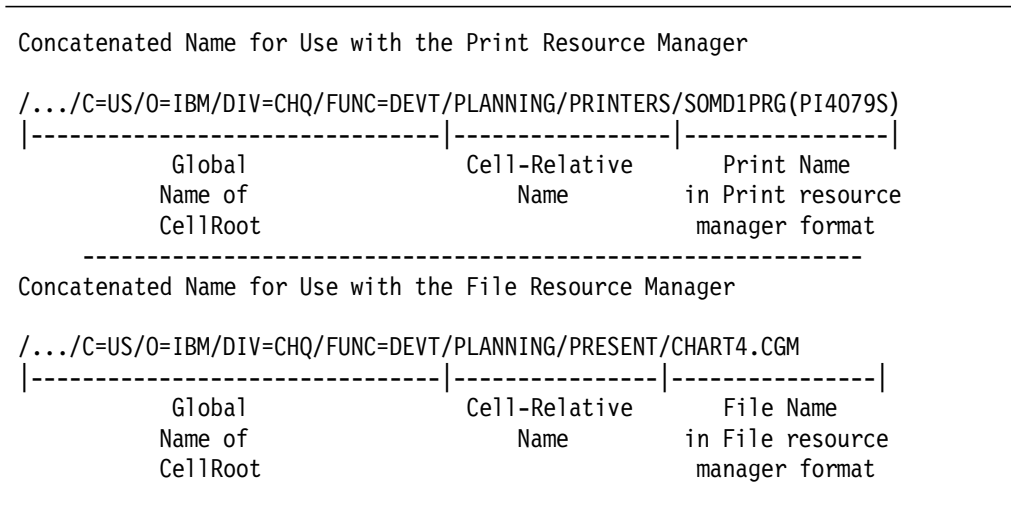


*Figure 29. Concatenated Naming Examples*

Concatenated naming, though not as robust a solution as universal naming, can be accomplished with existing resources at lower cost and less disruption to current components, while still providing some distinct benefits:

- It provides a common “root” for all resource identification. Without this concept, specific knowledge of how to find the root of each resource-manager namespace is needed to locate all resources. With a common root, it is possible to build universal resource browsers that can “discover” a resource, even though the browser had no prior knowledge of its type or existence.
- It establishes a single concept of administrative domain scope and ensures that all resources have a consistent approach to referring to instances in “foreign” administrative domains.

***Contextual Names:*** It is not always practical to use fully-qualified universal (or concatenated) names in every command, interface, or function. From a user perspective, such references would be time-consuming and error prone. From a programming perspective, such references require major changes in existing programming interfaces, and user interfaces. (A universal name can be up to 2048 characters long.)

Hence, it is common practice to use and support contextual names. A contextual name is a name that represents some portion of a fully-qualified name. Contextual names are valid only in a context in which the remainder of the fully-qualified name has been provided.

How context is established is currently an implementation choice of each resource manager. For existing resource managers, contextual names are often used to “map” their current programming interfaces to the universal namespace. For example, some portion of a resource manager-specific name (such as a device name, disk name, or group name) can be treated as a symbolic reference to an environment variable that contains the remainder of the fully-qualified name.

Contextual names also provide a reference scope. For example, a contextual name may refer to the name of a cell, defaulting to the current cell if not otherwise specified. Setting this value may implicitly allow all subsequent operations to be performed within the cell of choice.

***Directory:*** The directory service provides a database of information about resources in the distributed system. Since resources in the distributed system follow standard naming conventions, passing these names to the directory services allow resource manager client functions to learn about the location of a resource and all information needed to interact with the responsible resource manager server.

The directory service reduces the need for “side files” and other statically-defined configuration records. The directory enables a resource manager server, at the time a resource is created, to export information about that resource. Hence, client functions can dynamically learn the location of and how to access a resource, without it having been previously defined as part of the configuration.

The directory service is based on the DCE directory technology from the Open Software Foundation. This technology provides truly distributed directory services. The contents of the directory can be spread across many systems. As a result, looking up the information associated with a name can involve interactions with multiple directory servers.

A name is passed to the appropriate directory server to begin the lookup process. This server inspects consecutive segments of the name hierarchy until the name has been fully parsed, in which case the directory returns the associated attribute values. If the directory finds a reference to another directory server, it passes this referral back to the requesting system, which then forwards the remainder of the name to the next specified directory server.

With this technique, although there is a single global namespace, no single directory server needs to contain all of the entries. Furthermore, because the name is parsed a segment at a time, successive segments can employ unique syntax, as defined for the server that will handle that name segment. This feature enables name formation by combining global names with cell names and even with resource manager-specific names.

The directory service provides two classes of general directory servers, as defined by DCE: the global directory service (GDS), an X.500-compliant directory service, and the cell directory service (CDS). Both directory implementations support distributed naming and server replication for backup or performance optimization.

CDS uses DCE RPC for all client-to-server protocols (all name resolution services) and server-to-server protocols (such as replication). GDS uses communication protocols defined by OSI standards for both client-to-server and server-to-server protocols. Both CDS and GDS protocols flow across multiple underlying transports using Common Transport Semantics.

GDS provides a more robust set of name services, including the ability to look up an entry whose name is not known using attribute values. Both services allow public reads; however, specific authorization is required to update the directory.

A single programming interface, X/OPEN Directory Services/X/OPEN Object Manager (XDS/XOM) allows access to CDS and GDS. Using this API, an application is not concerned that a name may contain GDS and CDS parts.

***Directory User Interface:*** The directory service includes a user interface for storing, retrieving, and searching information in the namespace. This graphical user interface provides capabilities for "browsing" information stored in GDS, CDS, or the security registry. In addition, it is an open structure that can be extended to allow resource manager-specific information to be interpreted and presented. The resource managers provide the extensions necessary to enable their unique directory information to be viewed and manipulated by users.

***Security:*** Security is concerned[8] with identification and authentication of users, access control to resources, integrity of information, confidentiality of information, audit facilities, and management of security information. Security services are provided by many distributed resource managers.

The identification and authentication resource manager provides identification and authentication services. The access control resource manager provides access authorization services. The audit manager allows the specification of audit activities and collection of audit data. Other services such as information integrity, information confidentiality (or privacy), and non-repudiation are provided by other security resource managers.

In security terminology, an entity that requests a service is a *principal*. There are various kinds of principals in a system, such as users, servers, and machines. Each principal is assigned an identity that is managed in the distributed system such that it is unique across both space and time. Such identifiers are called universal unique identifiers (UUIDs).[9] These unforgeable identities are not user-friendly and, therefore, principals will also have user-friendly names such as JOE or WILSON associated with them. In the distributed system, security decisions are made using UUIDs because user-friendly names may not be unique. The binding between UUIDs and the user-friendly names is maintained in the security database, or security registry.

**Note:** This document uses the term "user" instead of "principal," except when the context requires "principal."

***Identification and Authentication:*** Users can potentially access the distributed system through many client systems. Authentication can be done through any of those systems. This process must be done before the use of any system service.

Authentication of a user proceeds in two steps. Local authentication involves validation of the user identity by a local identification and authentication resource manager (where one exists) using traditional approaches. For network authentication, identity validation with a network identification and authentication resource manager is also required.

---

[8] The basis of computer security is some level of physical security and software system integrity within any particular computer. The structure does not address the provision of this security but assumes that it can be provided. The structure addresses the distributed system issues, including the interoperation of systems having high levels of security with those having none.

[9] The UUID is a computer-oriented identification scheme that provides unique identifiers. UUIDs can be generated efficiently in many different places in the distributed system without ever having the same value generated more than once. This assumes that the UUID generator is not tampered with, works correctly, and that the "seed" provided for the generation is properly defined. The UUID is a 16-byte value that is opaque to all users. The only valid operation on UUIDs is a test for equality. The 16-byte UUID permits every person in the world to have a computer that generates a UUID every microsecond for three million years assuming an allocation mechanism that does not waste values.

- Network authentication begins by first locating (using the distributed directory) the appropriate identification and authentication resource manager instance that can validate the identity of this user.
- Sufficient identity-based information is then sent to the identification and authentication resource manager so that it can determine the user′s credentials. Credentials are the set of information, in addition to the user′s identity, that can be used by the access control resource manager to make access decisions. It is expressed as membership of specified groups, such as administrator, IBM employee, or dept 605[10].

Both activities are performed jointly using the same set of identity information provided by the user. Once local and network authentication are performed satisfactorily, the user will not have to be authenticated again for the duration of the session, no matter how many local or remote resource manager services he or she chooses to access.

Successful completion of local and network authentication is usually referred to as the equivalent of the successful completion of logon in standalone systems. Because the distributed system will not require any further identification from the user, it also possesses the property of single signon, a key user requirement. Server/machine authentication follows the same pattern, except that server passwords are stored suitably protected on the system—in a file for example—so that only the server logon code has access to the data.

When credentials are received from the identification and authentication resource manager, they are placed in the local security context of the user. They are available, through the security context management services, to any resource manager client acting on behalf of the user. When this information is transmitted to any resource manager, the resource manager can be assured that it is authentic[11].

Secure transmission of the user credentials between nodes is done through one of the communication resource managers. The integrity of these credentials is also protected by the communication resource managers during transmission. Integrity protection enables the recipient resource manager to detect any possible malicious or unintentional modification of the credentials during transmission. This is done by embedding credentials in a data structure called a certificate. Certificates cannot be fabricated by any untrusted party[12].

When authorized and requested, the identification and authentication resource manager can construct certificates that permit a user to pass on his privileges to a receiving server so that it can gain access to another server. For example, a user who wants to print a file may pass his or her right to read that file to the print server, so that the print server can read the file in order to print it. When a user permits another user to function on its behalf, possibly with some restrictions, to access a specific service, it is called delegation. When a user

---

[10] The credentials actually contain the UUID of each group, not a string name.

[11] If the system is not content-controlled, this information as well as all other information on the system could be compromised. However, the security system has specific safeguards to preclude user passwords or other critical security information from being captured and reused.

[12] If there are no untrusted parties in a position to fabricate the certificate, the certificate can be the user′s identity. When the path is not completely trusted, cryptographic certificates are required.

assumes the credentials of another user entirely with no changes, it is called impersonation, or forwarding.

Information regarding logon and logoff activity of users will be stored by the identification and authorization resource managers and may be distributed to appropriate resource managers. This knowledge is essential for products, such as office systems, that must be able to deliver mail to an individual.

Another example is the workflow resource manager that may need location information to distribute pieces of work. This location information cannot be relied upon because there is no guarantee that the user is still there by the time the information is used. Resource managers using the information must use the authentication mechanisms to ensure that the user with whom communication is established is the one expected.

***Access Control:*** The access control resource manager determines if a user is allowed to do what he or she is asking to do. It processes the authenticated identity and credentials against the access control list (ACL) that is maintained for the resource. The resource manager through which the access attempt is being made is responsible for initiating an access control check. The access control function of maintaining access control lists of users or groups of users with rights to named resources and services is generic.

A syntactically and semantically common form of access control list is defined so that resources can be moved between systems, along with their access control information, without changing the security control over them. The common form of the access control list is the OSF/DCE ACL with the direction being to migrate to the POSIX ACL when it is defined[13]. Resource managers use the common access control resource manager where it is appropriate.

***Information Integrity and Confidentiality:*** Information integrity and information confidentiality (when information is in storage and during transit) is achieved by invoking cryptographic services. Information integrity means using matching encrypted algorithmic results to ensure that the information has not changed. Information confidentiality means the transformation of the information itself through encryption so that only those principals that can provide specific keys can transform the information into a usable form. Non-repudiation guarantees the identity of the sender of information by using an encrypted “signature.” Various kinds of cryptographic services are supported. Client code or users are able to specify the options they want to use, as well as provide appropriate keys for decryption. Communication resource managers can provide these services transparently.

***Audit Services:*** Audit services monitor the activities of any principal. These services are available as APIs and commands. They are available through a special Graphical User Interface (GUI) from an administrator console as part of security administration facilities. A set of auditable events is provided from which the administrator can select appropriate activities to be audited. Audit reduction tools that integrate audit records from multiple nodes and resource managers are provided. The audit API is based on extensions to the POSIX audit APIs.

[13] This is the OSF direction also.

***Security Administration:*** A key attribute of security administration is that it can be performed within administrative scopes that are chosen by the customer. The distributed system can contain multiple administrative domains with controlled degrees of trust and delegation among them. Security administration provides centralized administration by which principals can be registered or unregistered simultaneously in a number of domains, and/or with a number of local security resource managers. Security administration includes functions for authorizing many principals to many resources following well-understood models of centralized administration. In addition, scaling of the registration and authorization functions to a large number of domains, as well as resources, is accomplished through role-based access models. A large number of roles may be created where each role has access to a large number of resources. Adding new principals is now achieved by linking principals to specific predefined roles. Examples of roles are bank tellers, financial credit managers, and loan managers in a banking system. Enterprise roles may be hierarchically related. For example, in a large department store, roles may be sales clerks (by department), department managers, floor managers, store managers, and regional managers. Security of the security management information is also critical and is achieved by using the previously discussed mechanisms.

***Time:*** The time resource manager maintains knowledge of the time of day and synchronizes the system clocks in the distributed system to a limited, but known, degree of accuracy. Whenever the accuracy of a local clock's time is beyond acceptable tolerance, time clients solicit the time from several time servers within the network. The local clock is then adjusted based on the intersection of the answers received from the time servers, allowing for processing and network transmission delays. Time servers synchronize with each other so that arbitrarily large networks can be synchronized.

Some time servers have access to authoritative quality time providers (such as a radio signal), so that even networks that are not interconnected are mutually synchronized[14]. All quality time providers in the world hold consistent time values. Therefore, the time resource manager would never attempt to adjust the clock of a quality time provider, such as the ES/9000* External Time Reference (ETR).

Adjusting a clock always takes the form of slightly changing its apparent tick rate, so that the clock gradually comes into synchronization, avoiding local requesters observing a discontinuity, and, especially, avoiding the appearance of the clock running backwards. During the period of adjustment, intervals measured by the local clock would be in error by about one percent.

Time synchronization and adjustment is performed in terms of a time-zone independent time standard, Universal Coordinated Time, (UTC[15]). The time service also maintains knowledge of "human" time, which is adjusted for time zones and daylight savings. "Human" time exhibits discontinuities and runs backwards (at changes between daylight-savings and standard times). UTC has discontinuities only when leap seconds are adjusted. Time values furnished by humans may be used to determine the offset from UTC to local time, but are never sufficiently reliable to determine UTC.

---

[14] If they are not synchronized, considerable problems arise if the networks become interconnected.

[15] *UTC* is the accepted (French) acronym replacing the term *GMT*.

***Transaction Manager:*** The transaction manager provides synchronization services so that multiple resource managers can act together to ensure that resources retain their integrity. The resources managed by each of them separately remain consistent according to relationships imposed externally, typically by application programs. The current use of the term transaction manager differs from earlier usage. This new terminology has been adopted to accurately reflect technical goals, to accurately reflect the functional parts of the distributed system structure, and to correlate with standard industry terminology. Major products, such as Customer Information Control System (CICS*), Encina**, and Information Management System (IMS), are combinations of the transaction manager, the transaction monitor, and other functions.

A distinguishing feature of transaction processing is that all the resource changes associated with a transaction must be committed before the transaction is complete. If there is a failure during execution of the transaction, all of the resource changes must be removed. Resources managed in this manner are called recoverable.

A typical example is a two-part financial application that credits one account and debits another. If a failure occurs after the credit, but before the debit, the application would want to back out the credit. The transaction manager interacts with the resource managers involved in the credit and debit, and ensures that either both actions complete or that the accounts remain unaltered. If the two accounts are located in different systems, the transaction managers in each system cooperate to eliminate any effects of a failed transaction.

Resource managers handle a sequence of operations against their resources such that the sequence of operations associated with a single transaction either wholly succeed or wholly fail, and in either case, the state of the resource is well-defined before and after the transaction. It is said that the resource operation sequence is atomic and that the integrity of the resource is maintained. Thus, a particular resource manager is responsible for the integrity of its resource, which includes recovering from physical or logical damage, backing out incomplete changes, and retrying operations.

Updates to multiple resources need to appear as a single atomic update called a logical unit of work. It is this logical unit of work that is the resource managed by the transaction manager. The transaction manager and resource managers exchange information about a logical unit of work, using an identifier called the logical unit of work identifier. A logical unit of work is required either to succeed wholly (all updates to all resources were applied successfully) or to fail wholly (none of the updates were applied). Inconsistent states, where some updates have been applied and some not applied, must not persist.

A single logical unit of work consists of operations on resources managed by many resource managers, possibly at several different locations. Several communication models may be used. Services of the transaction manager are used to coordinate the atomic completion of the distributed logical unit of work. Resource managers affected by a logical unit of work register their interest with their local transaction manager and are driven by it through a two-phase commit protocol. The two-phase commit protocol is used between a transaction manager and resource managers or other transaction managers to request that the involved resource managers:

1. Prepare to commit the changes (this is the first phase, which tests each resource manager to make sure that a commit can be performed.

2. Complete the commitment of the changes (this is the second phase).

An instance of the transaction manager operates in each system. When activity takes place outside a system, the communication resource manager used is responsible for mediating between the transaction managers in each system. The transaction manager is not aware of distribution or the type of communication.

The transaction manager supports three interfaces. These interfaces are illustrated in Figure 30.

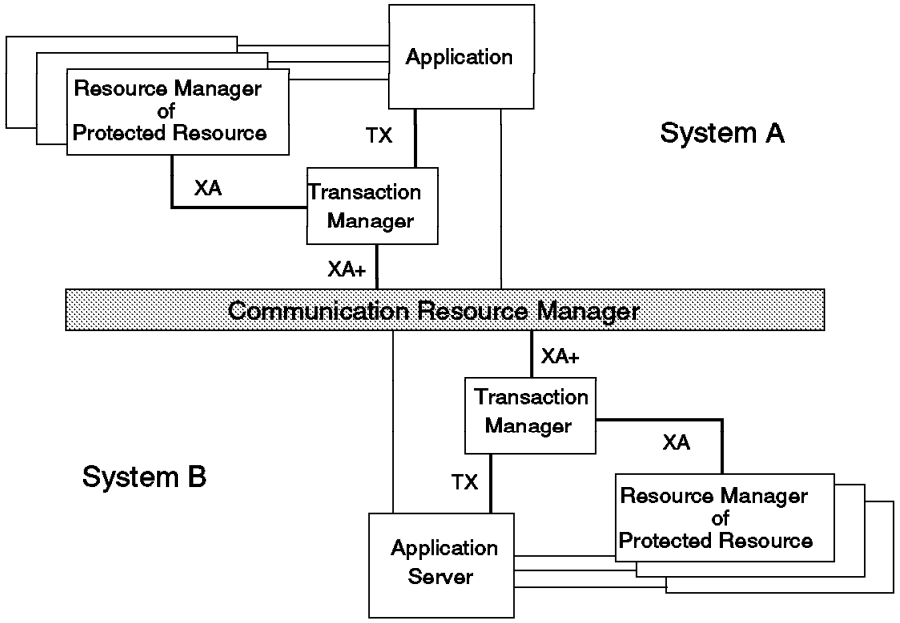


*Figure 30. Transaction Manager/Resource Manager Relationships*

**TX Interface** Used between the application program or its transaction monitor and the transaction manager to indicate when a logical unit of work begins and ends. In addition, if the logical unit of work is ending, the application can indicate whether the completion is correct and the resources can be committed, or the completion is in error and the resource changes are to be backed-out.

**XA Interface** Used between the transaction manager and the resource managers for the interaction needed to allow the transaction manager to synchronize all of the resource changes.

**XA+ Interface** Used between the transaction manager and the communication resource manager to inform a local transaction manager of the status of a distributed logical unit of work.

The X/Open standard definition of these interfaces is the basis for future technical work in transaction management, including evolving object-oriented transaction support. Current transaction management products will continue to support their existing interfaces. The functions of these current interfaces have served as a technology base for the X/Open transaction manager support.

### 3.4.2.4 Distributed Systems Services Integration

The distributed systems services resource managers work together and with local operating system services to provide an integrated, single-system image to using applications and higher-level resource managers. Some aspects of achieving this integration are as follows:

**Directory** At the highest level, information stored in the distributed directory is used by a resource manager to locate the target server (for the requested resource), select the appropriate protocol for communication with the target, and bind the appropriate method or driver code to the logical connection. The process of resolving the universal name of the resource returns the network location (network name) and communication/transport protocol (binding information) necessary for communication with the target server. This information (such as LU name, mode name, and TP name) is then used as input to network services to establish the appropriate transport connection and binding, and to complete the binding of the required communication method or driver code for the logical connection.

**Directory User Interface**
Used by resource managers to store resource manager specific information in the directory and enable it to be viewed and manipulated through the graphical user interface (see “Directory” on page 69). Alternatively, the resource managers may use the X/Open Directory Services/X/Open Object Manager (XDS/XOM) interface to store resource manager specific information in the directory.

**Security** Services used to obtain the credentials of the originating user[16], and augment those with credentials for the target server. The full credentials are then passed to the target server with the connection request for the target resource and with subsequent access requests to the target resource, as appropriate, based on the particular communication resource manager protocol.

**Transaction Manager**
Interfaces with a communication resource manager whenever communication with another transaction manager in a remote system is required to pass synchronization requests for a logical unit of work. The receiving communication resource manager in turn interfaces with the transaction manager in the remote system to pass it the synchronization request[17].

**Local Operating System Services**
Resource managers are dependent on a number of local operating system services for support of distribution, including the following:

- Maintaining the identity of the user (referred to as user context or environment state information) on whose behalf

[16] This refers to the user on whose behalf the resource is being accessed. Users’ credentials are established when they signed on to their workstations and the client security services signed on to the network on their behalf.

[17] If the communication resource manager supports a synchronization request protocol, such as the two-phase commit protocol of LU6.2, a mapping between the transaction manager protocol and the communication resource manager protocol is required.

the work is being done and providing interfaces for associating the user context with another process or thread on behalf of the resource manager, as required.

- Providing efficient task scheduling facilities for handling inbound communication from many clients and servers in support of resource manager servers of various types.

Applications and resource managers may choose to bypass the communication services specified by the Open Blueprint and implement their own private communication model. If this is done, the application or resource manager takes on the responsibilities described previously to provide an integrated, single-system image across their using applications and resource managers.

## 3.4.3 Application Enabling Services

### 3.4.3.1 Presentation Services

Presentation services provide several types of end-user interface services, all of which enable distributed processing.

***User Interface:*** The graphical user interface (GUI) has become the focus for the presentation of information between the network and the user. The goal is to provide the information in a form that most closely matches objects that users deal with in the real world. This “look and feel” is accomplished by an object-oriented user interface in which the user manipulates objects with “point-and-click” and “drag-and-drop” actions. These objects can represent any kind of data, including multimedia information.

This user interface is exemplified by IBM′s Common User Access* (CUA*) and OSF′s Motif**, which are similar in look and feel. IBM has implemented CUA in the OS/2* Workplace Shell*. The Common Open Software Environment (COSE) participants model their Common Desktop Environment (CDE) on Motif and CUA. The CDE is being submitted to X/Open for adoption in the UNIX** environment in 1994.

Presentation Services can be provided either locally or using distributed techniques. Microsoft Windows and OS/2 Presentation Manager are dominant in the PC environment and provide local presentation services. At present in the UNIX** and other environments, the X-windows** system provide presentation services that can be either local or distributed. Local presentation services can provide high performance because the video adapter can be connected over a high-speed local bus directly to the computer processor.

While today there are no common presentation services across desktop environments, IBM expects Taligent** object frameworks [18] to provide a rich set of presentation support controls (such as common desktop, window management, input event management, and multimedia) on all the major desktop platforms. Because these are object frameworks, they provide applications with capabilities that can be tailored to meet specific, individual needs. These object frameworks are the direct users of presentation management code. Moreover, since these frameworks are implemented using

[18] *Framework*, in this sense, refers to a collection of object classes designed to provide a specific set of functions.

the Object Manager, the objects that collaborate together on a desktop can be distributed transparently to other desktop and server systems.

An important aspect of the user interface support is the ability to integrate application work on the desktop. For example, an action in one application results in a function occurring in another application, or a change to data in one application is reflected in the data seen in another. This is accomplished through shared usage of the object frameworks provided. A specific example of the usage of an object framework is discussed in “Directory User Interface” on page 70.

A specific way to accomplish desktop application integration is through OpenDoc**. OpenDoc is an object framework offering derived from technologies available from the Component Integration Laboratories** (CIL). CIL is a consortium that provides a set of open technologies that support the integration of applications at the user interface. This integration is based on a model of a document whose parts are supplied by different applications.

OpenDoc describes what a part is, how it is structured, how its data is stored, how to visualize it, and how to communicate with it. Using OpenDoc, “containers” are constructed by embedding and linking parts from various applications into a single composition that has meaning to a user. Applications using OpenDoc collaborate to produce these compositions through the use of Object Management Services. OpenDoc technology enables the “inter-application collaboration” that is a key element of desktop integration.

OpenDoc uses SOM and related distributed technologies as its method for identifying and locating objects across systems. OpenDoc provides a complete methodology for development and execution. It offers existing applications a migration path into the object world without complete redesign.

***Print/View:*** The print and view services are made up of an open, distributed, print management facility and IBM′s Advanced Function Printing* (AFP*) and viewing facilities.

The print management services facilitate print submission, print resource management, and operational tasks in a heterogeneous network environment. Based on version 2 of the Palladium** print management technology, and developed jointly at MIT with IBM, Digital**, and Hewlett-Packard**, the technology conforms to the International Standards Organization (ISO) Document Printing Application (DPA) standard 10175, and tracks the emerging IEEE POSIX 1003.7.1 standard. Both standards have been endorsed by COSE. Designed specifically to operate in an open distributed environment, the technology provides a complete set of end-user functions to submit and control print jobs. It also provides extensive centralized systems management and operational functions to manage distributed print resources in this complex environment.

The print management services support industry-standard print data streams, such as PostScript** and HP** PCL**. When coupled with IBM′s Advanced Function Printing (AFP) architecture, the result is an industrial-strength, distributed print solution that includes:

- A high function, device-independent data stream architecture supporting both printing and viewing of documents without conversion
- A state-of-the-art, bidirectional print protocol (Intelligent Printer Data Stream* (IPDS*))

- Management of external objects associated with any page or document
- Architected indexing and segmentation of presentation data for viewing and archiving.

***Print:*** The print resource manager provides high function printing from anywhere to any printer in the distributed system. It is structured as a client and two forms of server: a print device supervisor and a print server. The client-to-server interface uses DCE RPC to deliver large quantities of data, such as a document to be printed.

Each instance of the print device supervisor manages a single printer and deals with printing one document at a time.

A print server manages a queue of print jobs that it receives and drives one or more print device supervisors to print the jobs. The print server exports an interface for the client to interrogate and manipulate the print queue.

Normally, clients work with a print server, synchronously delivering the document or the name of the document to be printed. The print server then takes care of scheduling the actual printing while the client machine proceeds with other work. The client can use a print device supervisor directly by simply naming a different logical printer.

The client/server protocol is ISO DPA. Line printer controller and spooler/line printer daemon (lpr/lpd) input is supported by a function protocol gateway for inbound print requests. SMB and NCP protocols are supported to access existing (legacy) print drivers.

***View:*** The View resource manager provides the ability to view, on the workstation, any printable document based on standard industry formats. Initially, this support is limited to documents conforming to IBM′s Advanced Function Presentation* Architecture, MO:DCA. The MO:DCA architecture was extended to enable the indexing, viewing, and archiving of documents, without conversion.

***Multimedia:*** Multimedia support allows computer systems to display, manipulate, control, and retrieve distributed graphics, audio, video, and animation data. Because multimedia data is digitized, it can also be searched and manipulated. Multimedia combines the interactivity of a computer with human modes of communication that makes the computer easy to use. The multimedia facilities provide an environment in which a heterogeneous set of multimedia supporting platforms and special-purpose equipment cooperate to support distributed interactive multimedia applications that process synchronized, time-based media.

Multimedia tools and applications can include capture and creation, composition and editing, and the playing of live and stored multimedia data. A key multimedia application is the support of real-time audio-video conferencing. Multimedia services can be used in extensions to business applications or specialized new applications [19].

[19] The multimedia resource manager is oriented to support multimedia data in the realm of business applications. The described support does not support what is typically called video on demand, especially for the home entertainment environment, including home television set-top boxes and large-scale video serving. However, the distinctions are not always clear. The two types of solutions will share technology (for example, ATM exploitation) and may share some implementation support.

System-level multimedia function is fundamentally oriented to the support of the transmission of time-dependent data, such as audio, video and animation. Time-dependent (isochronous) data must be transmitted and presented ("played") to the end user at a pace sufficient to maintain the usefulness of the data. For example, if a video was captured at 30 frames per second, it should be played at the same rate or very close to it. The challenge is to provide the requisite "real-time" support within the network operating system.

The multimedia resource manager uses presentation management services to support the playback of multimedia data. Device drivers are used to manage the specific hardware on which the data is presented, for example, speakers, displays, or microphones.

Multimedia playback can be standalone (that is, within one system), from a server based on redirected I/O, or distributed across a network, through the multimedia distributed resource manager. Systems that provide client multimedia support to end users typically provide standalone file system-supported playback, and use file system-supported redirection to support playback from a file server. The file system may be optimized to provide the support. The client system will also contain either hardware- or software-based decompression capabilities.

The multimedia support is oriented to support distributed multimedia (that is, playback from anywhere in the network), including multiparty audio-video conferencing. The need to handle very large data streams in real-time with satisfactory playback performance and synchronization of audio and video data streams can create stress on many areas in the systems. Thus, multimedia requires some specific support by the multimedia resource manager:

- Separation of control flow from data flow.

  In response to an API request to play multimedia data, the Multimedia resource manager client uses the directory to find the data. It then communicates through control flow with its server instance at that location to set up a data flow path from the source network driver or device driver to the target device driver at the lowest possible levels of the distributed system. Once the data flow path is set up, the multimedia data is streamed directly between the devices involved. In a play situation, the data never enters the requesting application's buffers.

- Quality of Service (QoS).

  As part of the control flow setup, the multimedia resource manager manages a commitment process whereby each component along the data flow path is asked to reserve its applicable resource (such as bandwidth, buffers, or cycles) to ensure that the multimedia data can be streamed as desired. Only when all participating components on all the involved nodes can provide the required levels of service will the data streaming be started.

The multimedia resource manager is object-oriented. Object classes are used to model multimedia devices and connections. Methods defined for the classes are used to control the device and network drivers that will do the data streaming. All communication supporting the set up of data streaming is through the object manager resource manager.

Multimedia support and its required QoS management depend on file systems, multiple networking resource managers, and local system services to support

both the resource reservation protocols that will provide end-to-end QoS and the data streaming of multimedia data.

### 3.4.3.2 Application Services

***Transaction Monitor:*** Transaction monitor is an industry term for functions that traditionally were included in IBM′s transaction processing systems, along with transaction management functions.

The transaction monitor provides an environment for the development and execution of applications, the transaction programs. The monitor typically provides an application programming interface and support for efficient transaction execution. The transaction monitor supports a large number of users concurrently sharing access to the transaction programs and the resources they access. Transaction monitors preallocate system and application resources, such as address spaces, connections to data, and other facilities. The preallocation allows transaction programs to be scheduled efficiently.

The transaction monitor uses the transaction manager directly, in many cases, to simplify the application programming implementation of the transaction. The application can identify or bracket all or portions of its processing to take place as a logical unit of work (see “Transaction Manager” on page 74). A transaction monitor also typically provides some application development and system management support for transaction programs.

The transaction monitor supports distributed transaction application execution. It uses the directory to find instances of other transaction monitors or specific resources supported by the transaction monitor. In addition, the transaction monitor enforces access control over the execution of transaction programs it manages. Before a user request is scheduled for execution, the monitor checks the authorization of the user to execute the transaction.

There are presently no formal standards for the transaction monitor application programming interfaces. The CICS transaction monitor API has been implemented on all major IBM platforms as well as on several non-IBM platforms. The IMS transaction monitor API has been implemented on a variety of platforms supporting applications associated with mainframe systems.

***Workflow Manager:*** Workflow management consists of a set of functions that help to define, execute, manage, and reengineer business processes across a heterogeneous system environment. Business processes can be quite diverse, such as contracting for a purchase or processing a mortgage loan application. They can also include support activities, such as application development or installing software updates. The workflow manager is a driver of complex applications. It is a coordinating agent initiating the execution of work by people, and the execution of programs in multiple, distributed workflow-managed processes.

In many companies, valuable process definitions can be buried deep within the logic of application programs. Changing a process means changing application programs, which can be time consuming and expensive. Workflow management makes application programs serve processes by separating the process definition from the applications performing the process. It helps organizations design processes, modify them more easily, and execute them more efficiently. An organization′s processes are an important asset, and companies in control of their processes have a definite competitive advantage.

The workflow resource manager contains components to support the reengineering of business processes through process model definition and documentation tools, as well as to control and facilitate the execution of those processes in open and distributed environments.

The workflow manager client function includes support for:

- Process setup and administration:
  - Process definition to capture and document work flows and test and analyze their execution before they are used
  - Administrative operations to manage workflow execution
  - Registration and configuration services
  - Post-execution analysis of logged measurement data.
- Process execution:
  - Work list handling, including support for end-user applications
  - User-specified workflow client applications.

The workflow manager server function includes support for:

- Management of all process models and instance state data
- Invocation and monitoring of function that is not driven by the end user
- Coordination of the state of multiple instances of multiple processes.

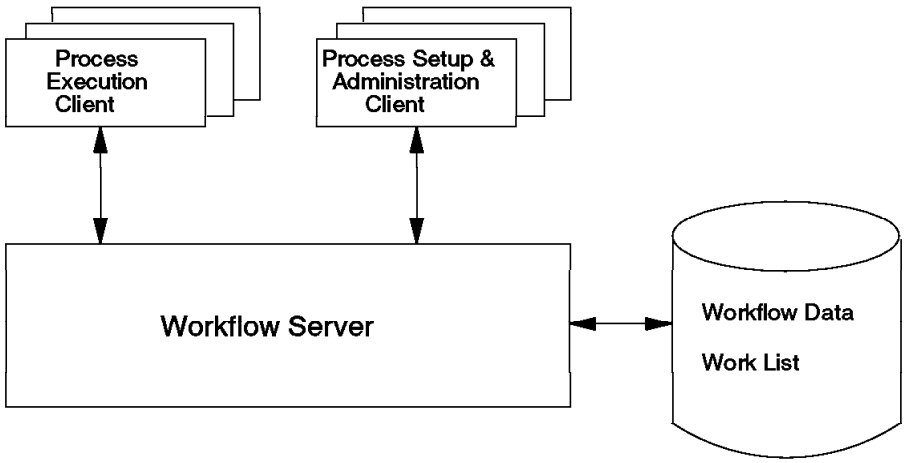


*Figure 31. Workflow Manager Structure*

The workflow resource manager exploits the services of the other resource managers:

- Database and File, for access to databases, files, forms, and images
- Directory, to locate people, applications and configurations

- Security, for identification and authentication, authorization, and access control
- Communication resource managers for all client-to-server and server-to-server communications.

IBM is a member of the newly formed Workflow Management Coalition whose goal is to foster interoperability among workflow products. The coalition will address application programming interfaces for access to workflow management services as well as formats and protocols for communication among workflow services.

***Mail:*** Electronic mail is viewed not only as a mechanism for users to exchange text messages but as a high-level service that other applications can use to send all types of materials to end users or other application programs. Applications in the distributed system can use electronic mail to transmit messages in many different forms. In addition to the interchange of unformatted text messages, the electronic mail service supports transmission of:

- Formatted documents
- Unstructured data and files
- Programs
- Graphics
- Images
- Video, audio, and other multimedia data.

Mail-enabled applications will use electronic mail service for many purposes. Some examples include:

- Electronic bulletin boards, conferences, and forums
- Interorganization business communication and transactions, including EDI
- Group scheduling and calendar systems
- Electronic publishing and distribution
- Electronic forms routing and workflow applications.

Electronic mail services have the following characteristics:

- Electronic mail services are completely asynchronous and are driven entirely by the senders of messages. As long as an application can connect to the electronic mail service, it can enter a message into the system regardless of whether the receiving application is presently active in the distributed system and even if the electronic mail service used by the receiving application is presently unavailable.
- The electronic mail service is a many-to-many, or multipoint, system in which any enabled application can direct messages to any other addressable recipient or recipients without any prior explicit agreement or protocol. If the electronic mail address of a recipient (application or end user) is known, electronic mail messages can be sent to it.
- The electronic mail system is completely transparent to the data it sends. Once the electronic mail envelope is addressed correctly, the electronic mail system delivers the content of the message to the recipient without depending on or disturbing the message′s content. After the receiver retrieves the message, the ability to understand and process the message is a function of that application.

Figure 32 on page 84 depicts the distributed system model for electronic mail. Electronic mail-enabled applications use an electronic mail agent as their

interface to the electronic mail service. This electronic mail agent in turn communicates with one or more electronic mail servers to store, route, and send messages. The electronic mail servers in turn communicate to form a store-and-forward network within the distributed system. At the receiving end, the messages are retained by a message store and can be retrieved by applications through their local electronic mail agent.

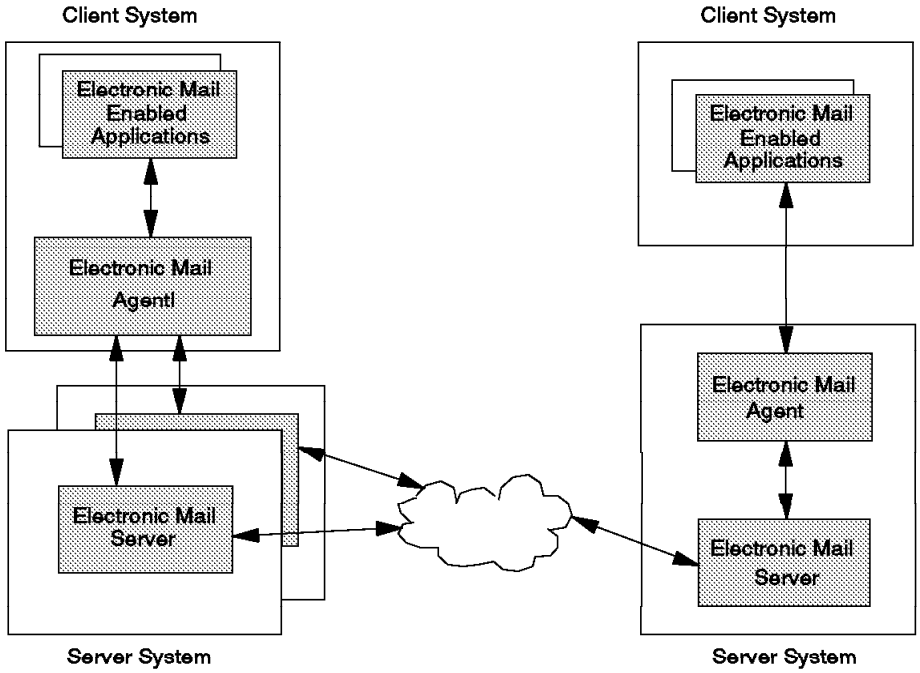


*Figure 32. Electronic Mail Model*

Presently, there are several open interface standard APIs that have become popular for application access to electronic mail services. Several companies, including IBM, have sponsored a group called the Vendor Independent Messaging consortium that has developed a standard application interface to electronic mail services called Vendor-Independent Messaging (VIM). VIM is intended to be a multiplatform standard that can be implemented by a variety of products running in different environments. Other standards also provide similar (basic) interfaces to electronic mail functions. Most of these standards are being extended to incorporate more sophisticated services. The Electronic Messaging Vendors Association (EMA) and the X/Open API Association (XAPIA) have defined a set of Common Messaging Calls (CMC). CMC is a commonly mappable subset of several of these definitions that electronic mail-enabled applications can use to access services that are provided in the implementations that support one of these standards.

Electronic mail agents should permit electronic mail-enabled applications to use any of the existing CMC interfaces. Additionally, IBM is working with its vendor partners and XAPIA to extend the open definition of CMC to encompass some of the more advanced electronic mail functions.

These electronic mail agents communicate with one or more electronic mail servers using the common transport semantics.

In addition to the interfaces that are used by applications to access electronic mail, there is a set of function protocols, known as electronic mail message transfer protocols, that is widely used in today′s networks. The Internet TCP/IP protocol supports a SENDMAIL service and Simple Mail Transfer Protocol (SMTP). Open Systems Interconnect (OSI) systems support the X.400 messaging protocol. Additionally, value added network providers, like AT&T**, MCI**, and Compu Serve** support their own protocols and supply gateways to support X.400 or SMTP. IBM products within the Systems Network Architecture support a store-and-forward distribution system called SNA Distribution Services (SNADS). Novell** NetWare** provides a Message Handling Service (MHS) that is widely used in DOS** systems for electronic mail. Most electronic mail vendors and network providers have recognized the need to connect to open systems protocols and at least provide gateway support from their own transfer protocols to X.400 and SMTP. In the Open Blueprint, electronic mail servers will support standard X.400, SMTP, and SNADS protocols directly and deliver access to other important proprietary message transfer standards through gateways.

Electronic mail agents and servers require the services of other resource managers. Communication services are used to connect agents to servers and to support server-to-server communication. Directory and security services are used to establish and authenticate client-to-agent, agent-to-server, and server-to-server interactions.

Mail-enabled applications can access an electronic mail address book through their electronic mail agents to determine where and how to send messages to particular system users or applications. Electronic mail-enabled applications will use the presentation and object management services to integrate electronic mail objects with other applications on the common desktop.

***Other:*** The structure we are discussing is still evolving; other application services may add additional functions in the future. In addition, there are other platform-specific application services that will be discussed in the Chapter 6, “IBM Software Platforms” on page 135.

### 3.4.3.3 Data Access Services

***File:*** The file resource manager provides access to byte-stream and record files throughout the distributed system, in the image of a single file system. Files are uniquely nameable through the hierarchic global naming conventions, and each local file system implementation will integrate access to its files through the distributed directory name resolution process and access control service.

File APIs can be classified as record, byte stream, or transfer.

**Record** Supports structured data (records, keys, and indexes).

**Byte stream** Data structures are left to application conventions.

**Transfer** Specifies how entire files are copied from one system to another.

File APIs are conveyed from file clients to file servers and among file servers through function protocols defined by specific file system implementations.

Two file protocols are considered primary:

- DDM[20] (Distributed Data Management).
- Distributed File System (DFS[21]) for file access in terms of byte streams. It supports a wide variety of semantics including byte stream requests, client caching, server-to-server caching, and file system administration.

DDM uses conversation protocols; DFS uses RPC.

As there are a large number of file system protocols[22] within the open, heterogeneous environment, the distributed file system is structured to support multiple protocols. Any file is accessible through any protocol, subject to the semantic limitations of the protocol.

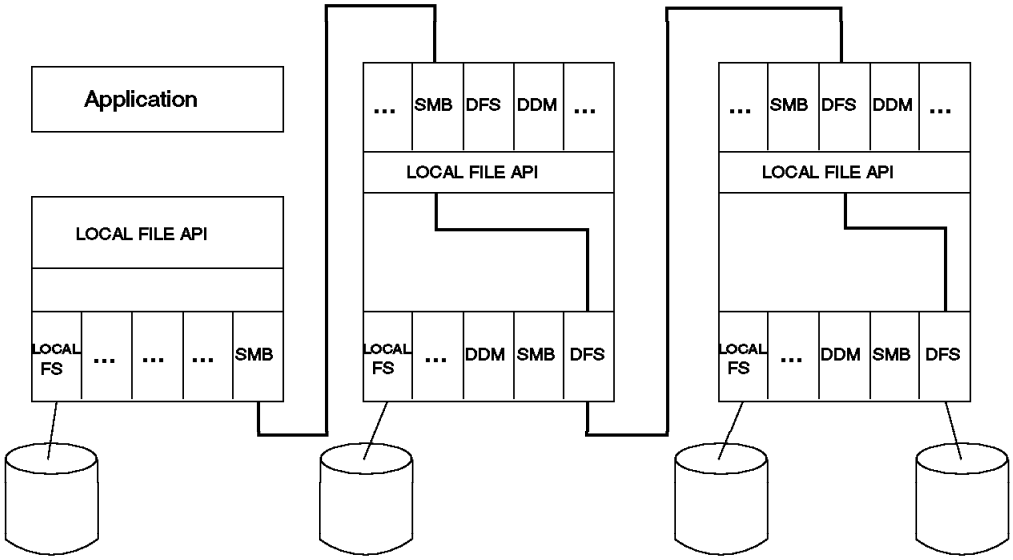


*Figure 33. Support for the Heterogeneous File Environment*

This is illustrated in Figure 33. The structure includes the possibility that a different file protocol is used between a client workstation and a file server, from that used between the file server and another file server. The first file server can, thus, shield the client from the multiplicity of file protocols, as well as possibly providing local caching of remote files, or exploiting a larger file server for archiving or back up of files.

File resource manager implementations integrate with the security services to exploit the user logon context established at the client (not requiring separate logon or user enrollment). File protocols support transmission over any standard networking transport by virtue of using transport-independant communication resource managers. When file gateways are involved, naming,

[20] The DDM architecture supports several classes of file semantics including record, byte stream, and transfer. It is also used to support Distributed Relational Database Architecture* DRDA*.

[21] DFS is the OSF/DCE file system protocol.

[22] Examples of these are System Message Block (SMB), Network File System (NFS**), and Netware Core Protocol** (NCP**).

location independance, and single signon objectives are maintained on an end-to-end basis (through the gateway).

***Database:*** Relational database facilities are made available through the Structured Query Language (SQL) by the relational database (RDB) resource manager.

The RDB is a distributed resource manager that supports clients on machines with no local database. RDB servers have a local database and also cooperate with other RDB servers to satisfy a client′s request. Thus, applications running on any system in a network have access to relational data residing on any other system in the network.

RDB resource manager clients use the server name specified by the application in the ISO SQL CONNECT statement to identify the universal name of the desired database. The universal name is then used to obtain information about the RDB resource manager database server from the directory. This information enables the client to establish communication with the server.

The connection to the RDB resource manager server is made using the user identity established at initial user logon. Authorization for access to the tables and views is handled by the RDB resource manager server, using the SQL GRANT and REVOKE concepts that are part of ISO SQL. The user or group identity to whom authorization is granted is derived from a universally-named principal or group. Finally, instances of the RDB resource managers cooperate with the transaction manager on their local system in order to coordinate database changes among the various RDB resource managers and with non-database resources.

Two standard RDB resource manager client-to-server protocols are defined: International Standards Organization′s Remote Database Access (RDA) protocol and the more advanced Distributed Relational Database Architecture (DRDA) protocol. Both are function protocols. RDA flows over OSI transport protocols. DRDA uses the conversational resource manager and can flow across any transport network supported by Common Transport Semantics.

While many commercial databases do or will support these protocols for interoperation with other vendor databases, nearly all use proprietary protocols between their client and server components.

Figure 34 on page 88 shows the use of a functional protocol gateway to support heterogeneous, distributed data over any installed, supported transport. Historically, application developers were forced to implement to a specific database client. To enable a single application to work with any installed client, X/Open has defined the SQL Call Level Interface (CLI).

The distributed RDB resource manager also provides support for data replication management for database information[23] to allow the performance optimizations needed for an application to achieve near local performance in the distributed system.

[23] As defined in IBM's Information Warehouse* strategy.

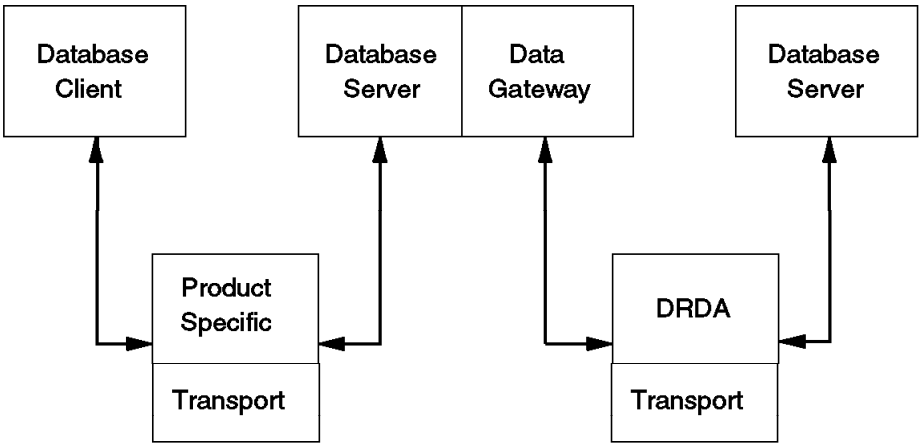


*Figure 34. Non-DRDA Access to DRDA Data*

***Hierarchical:*** In recognition of the need to access widely-used, existing data structures and formats, data access services provides interfaces to additional data types. Communication services may be used as the basis for the development of distributed capabilities for these interfaces. For example, Micro Focus** and IBM have developed a distributed capability for DL/1, which provides a consistent application program interface and remote access to DL/1 across several industry platforms, using the conversational model of communications.

***Object Data Base:*** Object-oriented database is an emerging technology with no well-accepted standards in place. The object-oriented database resource manager stores data that is not regular in structure as is, for example, record or relational table data. Computer aided design (CAD) data used in many engineering applications is typical of this type of data, but a growing number of other applications use non-regular data storage and access. IBM has negotiated a technology, development, and usage agreement with Object Design, Inc. to use its object-oriented database management system, ObjectStore**, in the development of software products.

### 3.4.4 Application Development Tools

From the perspective of the Open Blueprint structure, the application development (AD) tools are another class of applications. As such, both applications and development tools can use the structure′s resource managers and services. The unique role of the tools is to help developers create and maintain their own applications.

***Tool Diversity and Complexity:*** The information industry has inherited a great diversity of tools and skills, for example, COBOL on mainframes, C on UNIX systems, and some variant of BASIC on PCs. There are several client/server

design models that AD must support. And there are various ways of creating applications: writing them in third- or fourth-generation languages (either procedural or object-oriented), generating them from models of data and processes, and by visual construction or “assembly from parts.”

The facilities of the Open Blueprint structure offer major advantages in dealing with the complexity of distribution and heterogeneity. Rather than having to spend time and effort building functions that properly belong in the client/server infrastructure, applications and tools can use the existing resource managers and services. In addition to the benefits of enhanced developer productivity and reduced development time, the applications acquire functionality that is well-tested, fully-functional, and robust. Because the capabilities are based on established standards, the application components are highly portable and can easily interoperate with other application components developed separately.

***Tool Directions:*** Application development tool sets are organized around complete AD solutions, appropriate for the particular development community that they serve. These AD tool sets support workstations and servers as the primary environment for AD.

Some AD solutions, especially those targeted to professional developers of enterprise business critical applications, are language-centered. These include Smalltalk** and C++ development facilities, particularly for the workstation. In the near future, client/server solutions centered around COBOL should be available.

In other AD solutions, languages play a less central role, for example, those that offer visual construction from predefined parts or model-driven generation of distributed applications.

Both include a set of tools and one or more language processors. The distinction is in which development capability is emphasized to the user.

Indeed, there is a range of AD solutions, with tradeoffs among a number of factors, such as: productivity, flexibility, performance, and maintainability. The high-level language solutions offer the most flexibility, given that they can exploit services at any level in the structure. Generators and visual builders use a selected set of services that can provide a balance between insulating the application from changing technology, and optimizing the application for particular client and server platforms.

***Object-Oriented Technology:*** Use of this technology for application development provides benefits for developer productivity and quality. In addition, use of the object-oriented paradigm can result in encapsulated data structures and associated functions with well-defined interfaces between the objects. These characteristics make objects particularly well-suited for distributed computing.

IBM is a major participant in this area, as evidenced by the inclusion in the Distributed Systems Services of the SOM and distributed SOM implementations of CORBA-compliant object management services. SOM and distributed SOM are supported in C++ language solutions as well as object-oriented visual programming and construction-from-components tools.

In addition to the *Application Development Reference* which is part of the Library for Systems Solutions, the International Technical Support Organizations have written a number of books on designing and developing client/server

applications on various system platforms. Some recent examples are: *Client/Server Computing: The Design and Coding of a Business Application*, and *Client/Server Computing Application Design Guidelines: A Distributed Relational Data Perspective* .

### 3.4.5 Systems Management Services

Manageability of software and hardware resources is critical to users in the open, distributed environment. Many enterprises have sophisticated distributed systems interconnected over complex local and wide area networks. For example:

- Software must be able to be distributed electronically and installed either locally or remotely.
- Configuration information for resources should be self-defining and minimize user involvement.
- Basic error logging and fault isolation functions are required to support improved and automated problem determination.
- Standard management definitions and protocols should provide for the collection of error and performance data and the application of changes and fixes.
- Management facilities for systems administrators are required for both local and remote resources. Installations should be able to choose either a centralized or distributed management scheme.
- Standard APIs for management are needed to facilitate interoperability and application portability.

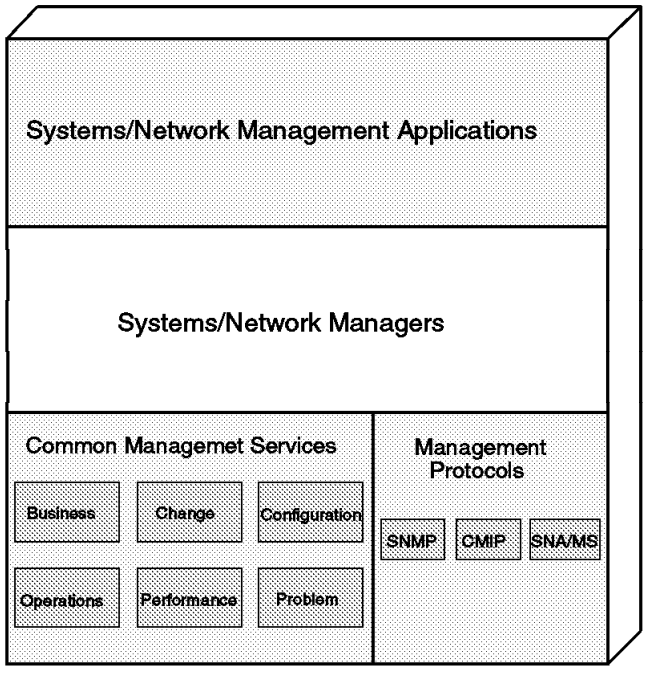


*Figure 35. Systems Management Services Structure*

The Open Blueprint includes a structure that allows for resources to be managed across open, heterogeneous, enterprise-wide, distributed systems. It contains a systems management component that is based on the SystemView* structure. The SystemView initiative defines the required management applications, management services, and structure. The SystemView structure, Open Blueprint, and Networking Blueprint describe complementary aspects of capabilities needed to address a heterogeneous enterprise environment: systems management, systems structure, and networking.

***Systems Management Structure:*** The resource managers use and benefit from a common set of management services. These services are described as part of the SystemView disciplines. They are: business management (for example, asset, license, and security), change management, configuration management, operations management, performance management, and problem management. In the Open Blueprint, systems management is shown as an attached component to emphasize its importance across all resource managers and to indicate that additional systems management services are needed.

Figure 35 on page 90 illustrates the systems management structure, which is described in the following sections:

***Common Management Services:*** The common management services are part of the SystemView* disciplines. They exploit the distribution services (such as directory and security) and enable all resource managers to use their external interfaces. The services are related to, and used by, the management applications of a given discipline. The services perform essential functions, like data collection. The following common management services provide enhanced manageability of the distributed enterprise:

- Business management: includes identification and monitoring of system components through standardized, vital product data structures (such as hardware serial numbers and manufacturer), as well as license management of software.
- Change management: is involved in any planned alteration to an information systems environment, including installation, distribution, planning, and scheduling functions.
- Configuration management: focuses on making software and hardware operational, including functions for the application of rules involved with prerequisites, corequisites, and policy constraints.
- Operations management: is involved with the monitoring of registered events by “producers” as well as the dispatching of events to “consumers” of events.
- Performance management: is involved with statistical counters and tracepoints, including functions for performance probing and performance tuning.
- Problem management: enables serviceability, including functions for problem determination and error logging.

***Management Protocols:*** The management protocols supported encompass a broad range of *de jure* and *de facto* standards. These protocols support the Systems/Network Managers interaction with either the common management services or other entities managing system resources. The key protocols are Simple Network Management Protocol (SNMP), Common Management Information Protocol (CMIP), and Systems Network Architecture/Management

Services (SNA/MS). Over time, the management protocol supported will also be provided by the object manager.

***Systems/Network Managers:*** The Systems/Network Managers exploit the data collected by the common management services to perform analysis and correlation. They provide systems and network management-specific APIs, services, and protocols. They allow for management of multiprotocol, multivendor networks, and enable delivery of integrated management applications through industry-defined APIs (XMP, SNMP).

The organizing concept in systems/network management is the identification of managed objects[24] which includes the specification of their attributes and behavior. This provides a concrete description of *what* is manageable. The *how* of management is defined by managing applications that support the management user and manage the resources of the system.

All systems are managed; that is, they have managed objects. When a system runs managing applications, it acts in a manager role. The Systems/Network Managers in the system are sufficient to run managing applications.

Resource managers provide methods that define, create and maintain their managed objects. They should enable generic managing applications, and resource manager unique managing applications, where necessary. Requests from the managing object to the managed object flow through the Systems/Network Managers.

Functionally, the Systems/Network Managers provide a rich set of services to manage the data received from a diverse multivendor network of resources, including devices, LANs, and systems.

To provide for distributed management and increase control of critical resources, Systems/Network Managers can include system monitoring functions that are placed appropriately in the network. These monitoring functions collect management data and allow for customization of information dispatched to the "manager." Trap filtering, threshold processing, and resource utilization are some of the functions supported.

The Systems/Network Manager functions provide for dynamic network discovery and visualization of network topology, using presentation services. The graphical display of multiple protocol topology supports user-definable views (such as protocol groupings, device groupings, and geography) and layouts. Continuous monitoring of network status, alerts, and performance thresholds is supported.

***Systems/Network Management Applications:*** The Systems/Network management applications support the SystemView* disciplines. They provide integration, coordination, and distribution of systems management functions. The management applications exploit the management interfaces (XMP and SNMP APIs) or other services to obtain the data they need.

Application integration is achieved by using common and integrated management end-user interfaces and collected management data. Management

[24] The OSI X.700 concept of managed objects, representing instances of resources, applies equally to existing resource manager implementations, as well as new ones designed with object-oriented concepts.

applications use the same end-user interfaces as do other applications. The presentation services are used to support a common and integrated end-user interface. The data access services are used to support common and integrated management data. Separating application logic from end-user interface and data access ensures that any application provider, not just IBM, can provide new applications that integrate well into the existing environment.

Coordination of systems and network management for the entire environment is achieved in two ways:

- Management applications operate on data collected by agents (that is, resource managers or the services of the Systems/Network Managers) to manage the resources.
- Management applications can also interact with certain agents using the management APIs and protocols provided by the Systems/Network Managers.

Distribution is enabled through the use of distributed systems services, such as directory and security support. Managing applications and agents can be located on physically distinct systems. Likewise, managing applications, management data, and even the management end-user interface can be on different systems. In each case, factors such as performance and usability need to be considered to determine placement.

***Management Standards:*** The following industry standards are supported by Systems/Network Managers:

- The OSI X.700 Management Model
- The SNMP, CMIP, and SNA/MS protocols
- The industry-standard management APIs XMP and SNMP, which enable portability for management applications.

There are several emerging standards for the common management services:

- POSIX System Administration services (1003.7 group)
- The Desktop Management Task Force (DMTF) Desktop Management Interface (DMI) APIs for Vital Product Data
- OSF/DME NetLS API for License Management
- X/Open System Management Object Services APIs.

The systems management structure will continue to evolve to support additional management protocols and APIs as they emerge.

## 3.5 Other Efforts

While the Open Blueprint is the vehicle that the authors have selected as a descriptive model to further develop this discussion of open, distributed systems and the issues of application distribution, other organizations have also developed related concepts. To assist the reader in awareness of other concepts that might well arise in discussion, this section briefly explores some of them.

The previous detailed discussion of the Open Blueprint will provide the basis for further development, as explained in the summary following

### 3.5.1 MUSIC

MUSIC was conceived by the Government Centre for Information Systems in the U.K., and is presented in Figure 36. The MUSIC framework allows analysis of applications and systems by categorizing the various functions that are provided.

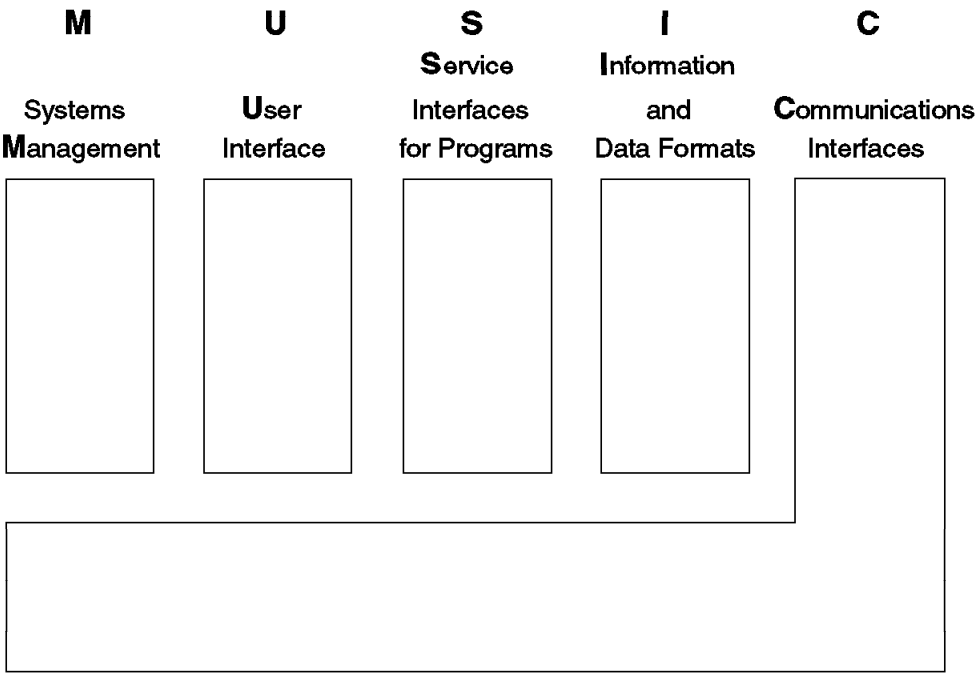


*Figure 36. MUSIC Framework*

Although MUSIC can simplify comparison of unlike applications or systems, it is too general to sustain the level of discussion we intend for this book, and we will not discuss MUSIC further. Nevertheless, you may find MUSIC to be useful when examining specific application or system alternatives.

### 3.5.2 X/OPEN** Distributed Computing Services

The X/OPEN** Distributed Computing Services framework is shown in Figure 37. The major difference in this figure is the explicit listing of component attributes in the areas of:

- Availability
- Internationalization
- Manageability
- Security.

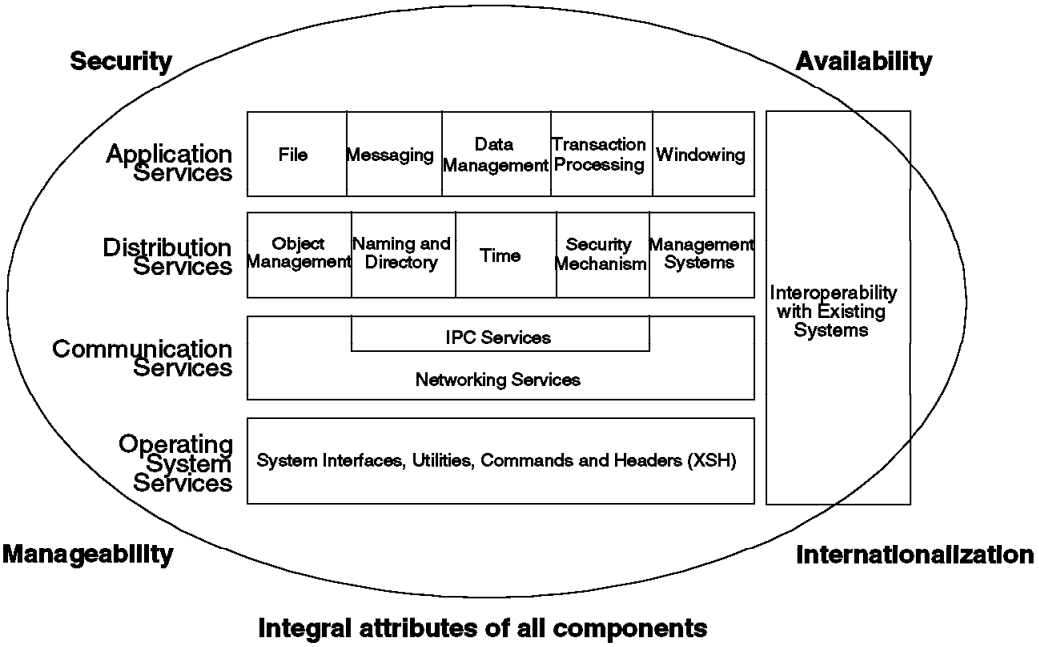


*Figure 37. X/OPEN** Distributed Computing Services*

These attributes apply to any effective distributed system structure. There is clearly no conflict between the X/OPEN** DCS framework and the figure we are using. We have elected to organize our discussion around the concept in Figure 40 on page 100, because we believe that concept is easier to understand.

### 3.5.3 X/OPEN** Portability Guide (XPG)

The X/OPEN** Portability Guide has had four versions from 1985 to 1992, adding more function in each version. They are a recommended set of standards (including de facto standards) and profiles (options in the standards) which will assist in portability, as shown in Figure 38.

Since the XPG versions are not intended to describe a system structure, they will not be discussed further. However, some of the standards they include will be discussed.

X/OPEN** has also defined a Common Application Environment (CAE). It describes the application environment which is obtained by following the X/OPEN** Portability Guides, and is similar to the SAA* picture in Figure 16 on page 43 (but offers a greater level of detail). We have elected to use the picture in Figure 40 on page 100 because it makes it easier for us to discuss the aspects of application and system design. However, there is no conflict with X/OPEN** CAE.

| | | |
|---|---|---|
| **XPG4**** 1992 | LMX Server | X.400 Message Access |
| | (PC) NFS** Server | X.400 Gateway |
| | CPI-C | Byte Stream File |
| | Network File System (NFS)** | X-windows** System |
| | Directory Access | |
| **XPG3**** 1989 | Transport Interface (XTI) | Ada Language |
| | X-window** System | |
| **XPG2** 1987 | Pascal Language | Relational Database |
| | Commands and Utilities | Terminal Interface |
| **XPG1** 1985 | Magnetic Media | COBOL Language |
| | ISAM | C Language |
| | FORTRAN Language | System Calls and Libraries |

*Figure 38. X/OPEN** Portability Guides (XPG)*

### 3.5.4 Open Software Foundation Distributed Computing Environment (OSF/DCE**)

OSF** has developed a set of technologies for distributed computing, which it licenses to interested companies. The key elements of this technology are shown in Figure 39. DCE** provides distributed application services, such as Distributed File System; and distributed system services, such as RPC. Although the DCE** picture does not look exactly like the structure we will use for discussion, it is clear that DCE** provides key functions of the open distributed system structure. IBM* and many other system vendors have expressed their intent to provide DCE** support.

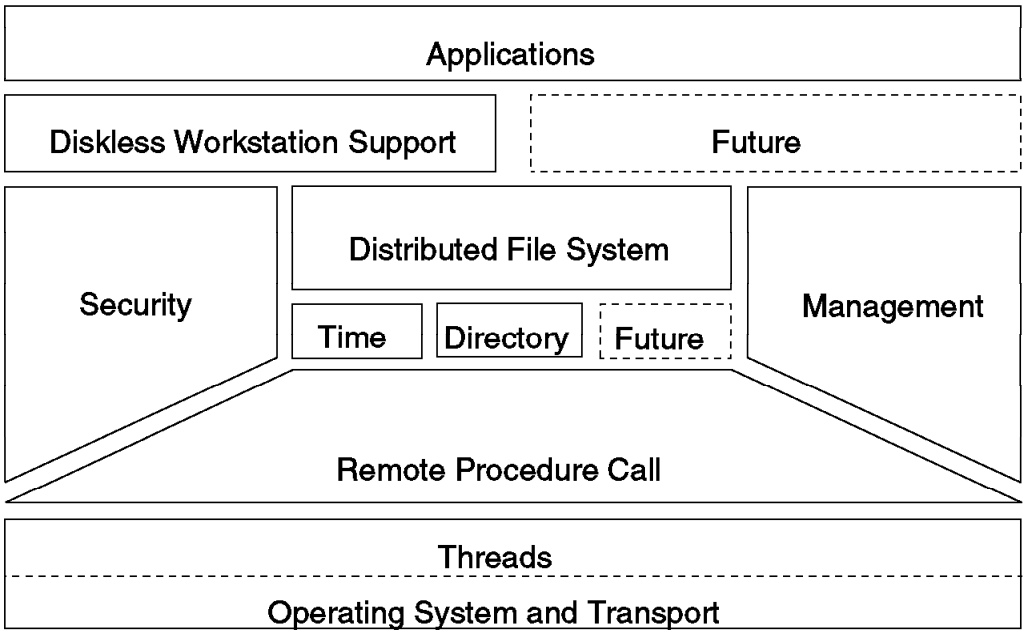


*Figure 39. Open Software Foundation (OSF**) Distributed Computing Environment (DCE**)*

Because DCE** addresses only a subset of the system structure, we cannot use it to organize our entire discussion, but you will see other references to DCE** in this book.

## 3.6 Summary

Having now examined the IBM* Open Blueprint in detail, through the vehicle of the Open Blueprint Technical Overview (GC23-3808), and reviewed other structures, it is now necessary to show how the IBM* Open Blueprint structure will be used as a descriptive framework in this document. A slight adaptation of the IBM* Open Blueprint figure will be used to assist in the process.

The recent material that has been explored has served as background to the fundamental objective of this document. This objective, as restated from the abstract, is to support technical professionals involved in defining solutions to data processing problems, in multiple configuration environments and multiple software platforms, including heterogeneous distributed environments.

Before the framework is described we should review some key distribution concepts which will further assist in acceptance of the framework.

### 3.6.1 Distribution Concepts Revisited

A distributed system is inherently more complex than a single, stand-alone system. Making a distributed system practical is a matter of providing a level of conceptual simplicity to hide physical complexity. At some level, conceptual simplicity usually means making a physical distinction transparent or invisible to someone.

There are approaches to distributed systems that provide transparency to the distinctions between different systems within the distributed system at a very low level. The single computer image creates the image of a single computer at, or almost at, the hardware. Programmers can ignore the several real processors, perhaps to the extent that storage can appear to be shared among processors, and certainly to the degree that the system can move programs between processors transparently.

This approach has been possible in environments with considerable homogeneity, with relatively close coupling, and relatively small numbers of processors. The approach, however, does not appear to be practical for more heterogeneous hardware and software and for less closely coupled systems. In these cases, it is necessary, for practical and efficient operation of the distributed system, for higher layers of the system to be aware of the distinction between systems, for example to modify their behavior depending on whether a resource is local or remote, or to use mechanisms that are native to one system and different on other systems.

Even so, it is not desirable to show the distinctions to the end user, but rather to hide the distinction at some level below the end user. Nor is it desirable to expose all the intricacies of the physical system to a level of software that has to be aware of only some of them.

Therefore, an appropriate structure provides layers of abstraction, progressively hiding the complexities of the system, while permitting each element to efficiently handle those complexities that it must.

For example, an application program may handle distribution explicitly, while hiding the distribution from its end user. Alternatively, an application program may execute in a single node, but use programming interfaces to resource managers that provide transparent access to distributed resources. The

resource manager may provide the transparent access by explicit programming, or might, itself, take advantage of transparent distributed services.

### 3.6.2 Local Operating System Services

Local Operating System Services are also key to the descriptive framework. They are local resource managers and services that support the IBM* Open Blueprint distributed resource managers. Local resource managers manage elements such as memory, CPU′s, or devices.

Local Services can include:

- Work management
- Environment state support
- Memory management
- Event handling
- Security context management
- Locking Service
- Accounting
- Tracing
- Journaling
- Language environment

In our discussions, we need to expand on this area because the subject range of this publication is not only distributed environments but also traditional environments. Local operating system services will continue to be a key factor in the development of distributed applications and in all these environments for several years. This demands a closer examination of local operating system services than the IBM* Open Blueprint figure represented in Figure 20 on page 50 permits.

Therefore, in our discussions, we will use a variation of the figure that explicitly extends the view of local operating system services, as shown in Figure 40 on page 100. This also makes it easier to discuss existing system components that were not really designed with distributed processing in mind.

### 3.6.3 Descriptive Framework

Before we develop our discussion, note that:

- This is not an operating system or processor architecture definition. It is a framework for discussion.
- The layering is for convenience in understanding.
- Products will evolve to enable the capabilities and options outlined in this structure.
- A product may provide functions, or an application may request services, which reside in different layers or in different elements of the same layer.
- We will focus much of our attention on distributed systems, because that is the direction in which the industry is evolving. We do not mean to imply, however, that all applications or all systems should be distributed.
- This structure for discussion should be applicable to any system, from IBM* or from other vendors.

- The structure organizes a software platform into applications, sets of related services and resource managers.

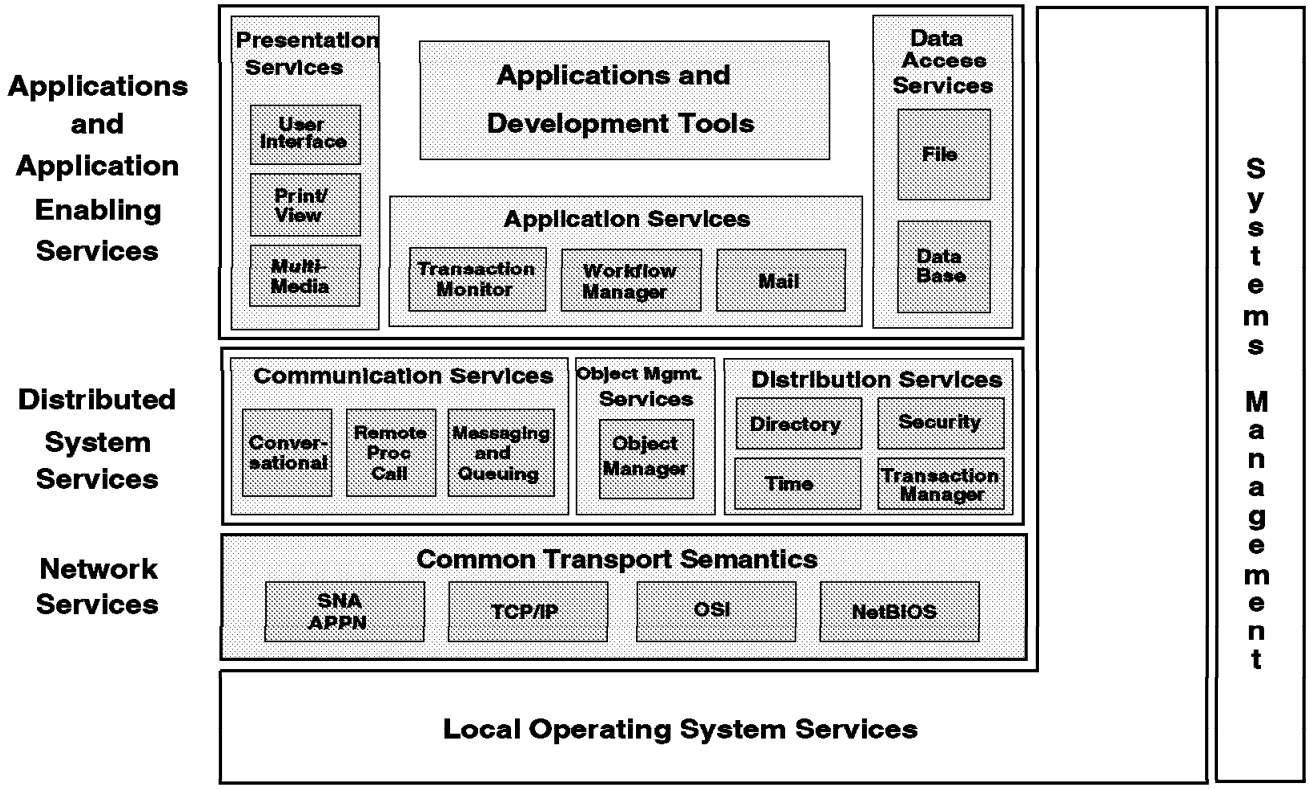


*Figure 40. Elements of a Software Platform*

### 3.6.3.1 Resource Managers and Services

The resource managers (see section 3.3.4.1, “Resource Manager Concepts” on page 52 for more on resource managers) are shown as the interfaces they provide through services as follows:

- **Application enabling services**
  - *Presentation services* control the end-user interaction with applications. This includes display, screen management, printing and viewing of information, and multimedia interaction. All of these modes of presentation allow for distributing portions of the end-user interface processing.
  - *Application services*
    - Transaction monitors provide an application development and execution management environment. They typically include a scheduling function; they interface with data resource managers; and they provide communication functions to introduce new work into the system. They manage distributed transactions transparently to the applications and the end users.

- Workflow manager provides for execution of business processes. It allows a business process to use application portions that are distributed, shielding end users, as well as application developers, from the specifics of the distribution being used.
- Electronic mail is a requirement of nearly all businesses. It can also serve as a base for higher-level applications.
- Other platform dependent services are possible for future technology evolution.

– *Data access services* control access to data in files, data bases (relational or other), and object oriented formats. They shield applications and other distributed resource managers from explicit awareness of the location of data.

Application enabling services make use of communications services to achieve distribution in a number of ways, called distribution models, which vary according to which element is aware of the distribution and which communications model is used.

- **Distributed system services**

  Distributed system services are an additional set of distributed resource managers that provide communication services and models for the relationship between communicating programs.

  Each program of a pair of programs that work together must have some understanding of the operation of the other. This understanding, and its encoding, is called the protocol. Thus, two programs must have a common protocol to be able to work together.

  The following distributed system services are defined:

  – *Communication services* allow distributed application portions to interact in three styles, or models. Conversational is similar to a human telephone conversation; RPC (Remote Procedure Call) is similar to calling an application sub-routine; and Messaging Queuing is similar to mailing a letter.

  – *Object management services* provide transparent access of local and remote objects. Data attributes and methods of any specific object are found at a single location.

  – *Distribution services* include:

    - Directory functions resolve actual location information, which simplifies application development, and allow changes without impact to programmers or end users.
    - Security functions provide for identifying end users and portions of applications, ensuring only appropriate access to resources, such as data, is allowed.
    - Time services maintain consistent time information across distributed systems, allowing for such things as time-based data recovery.
    - Transaction managers protect the integrity of data. They provide for backup and recovery, and offer services that synchronize updates to multiple resources as a single unit of work. Because there may be several transaction managers in a distributed environment, a 2-phase commit process synchronizes all the distributed pieces of a specific unit of work.

- **Network services**

  Network services are an additional set of distributed resource managers that transport data from an end-point in one system to an end-point in another. All the transport mechanisms supported by the network services are accessible through a transport independent interface. A system can have more than one transport independent interface, but all transport networks are accessible through each of the interfaces. The choice of the transport independent interface used by a requester is not detectable outside the system and, therefore, has no effect on interoperation.

  Network services support a wide range of transport protocols, such as SNA, TCP/IP, OSI, and NetBIOS. They provide *common transportation semantics*, shielding higher level services and applications from the specific protocol being used to connect two parts of a distributed system.

- **Local operating system services**

  Local operating system services control resources, such as processor time and storage; data storage on disk, diskette, tape, and so on; and displays (screens) that are attached to that specific system. Many operating system services fall into this category. They interact with distributed resource managers at all levels of the structure.

  Local operating system services include an environment state resource manager that maintains information about the environment in which a program is executing, such as the name of the user on whose behalf the work is being done.

- **Systems management**

  Systems management is a crucial element of any distributed system and interacts with all the elements of the structure.

### 3.6.4 Conclusion

In the remainder of this document, this framework, as shown in Figure 40 on page 100, will be used to organize the discussion. In Chapter 6, “IBM Software Platforms” on page 135, the same structure will be used to discuss a number of IBM* operating system platforms. These discussions will work from the bottom up. As we have seen, there are multiple pictures and organizational techniques that can be used to discuss distributed systems. Many of these are remarkably similar.

The framework and its relationship to the Open Blueprint provide a critical link with current structural thinking.

# Chapter 4. Configuration Environments

The total computing environment in any large enterprise has become quite complex during the last decade, and this complexity continues to increase. Defining functional subsets, providing they correspond to realistic modes of usage, allows us to simplify planning and design work. System solutions, and especially distributed system solutions, tend to be associated with particular configuration environments, some of which are shown in Figure 41.

This section describes one categorization method we have found useful.

The method is built around the way terminals, workstations, and systems are used by a particular application. The term *system* is used here to indicate any hardware equipment capable of executing applications and system software.

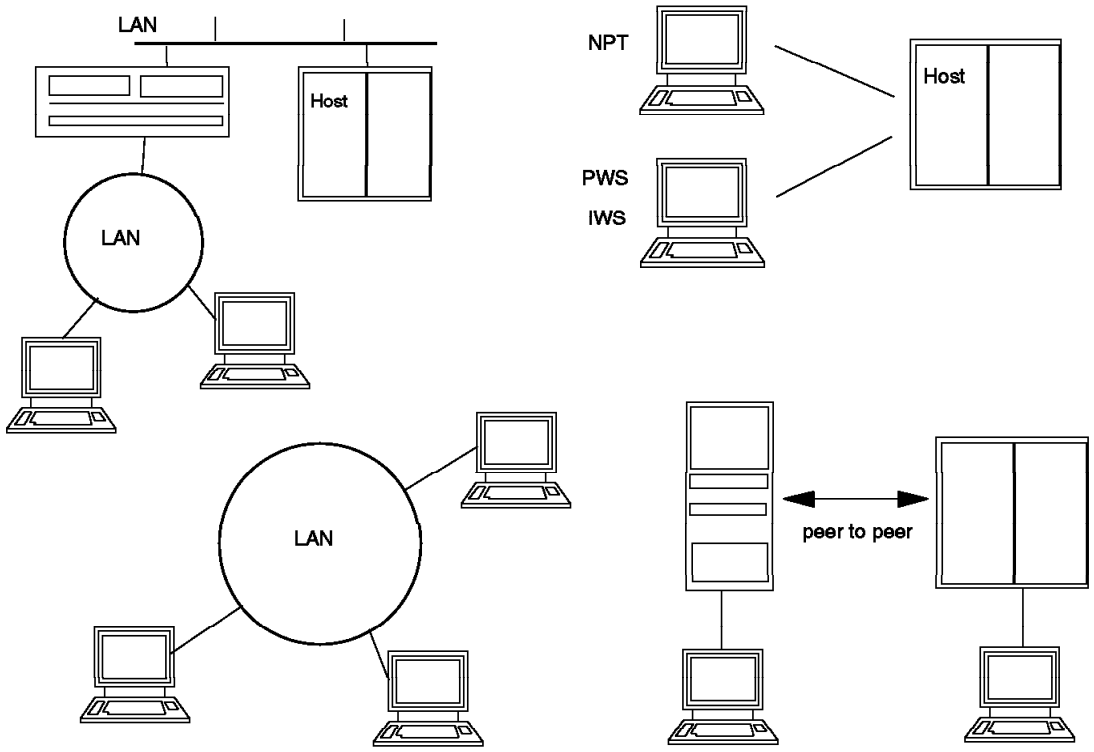


*Figure 41. Configuration Environments*

The categories represent modes of operation rather than physical configurations. The same hardware might function as a non-programmable terminal environment for one application and as a LAN environment for another application. We find that many applications, or major subsets of more complex applications, are often designed for one of these modes.

Different applications, with a given set of hardware, and with a given degree of functional distribution, will very likely fall into at least one of the different categories.

The categories are:

1. Non-programmable terminal (NPT) environments.
2. Wide-area network (WAN) environments.
3. Local-area network (LAN) environments.
4. Multi-level server environments.
5. Other interconnected systems and peer-to-peer environments.

The books in *The Library for Systems Solutions* will generally refer to these environments when describing their systems solutions.

Except for the NPT environment, where the processing power is available in only one place, all the other environments have distributed processing power, allowing distributed systems solutions.

#### 4.1.1.1 Other Approaches

Other ways exist to categorize distributed systems solutions.

As an example, take into consideration the way the functional components of an application are distributed. If an application is seen as a mixture of presentation logic, data logic, or business logic, distributed applications might be categorized as distributed presentation, distributed function, and distributed data applications. Each category also has the possibility of sub-categories.

This approach to distributed processing is the basis of a model for distribution used by the Gartner Group.

Distributed applications might also be categorized based on the relationships between the distributed application components or the style of communications. With this approach, there are client/server solutions, peer to peer solutions, and cooperative solutions, just to refer to the most commonly used industry terms.

When multiple levels of distribution are involved, such as in the multi-level server environment, multiple styles of communication and multiple levels of functional distribution might coexist in the same application solution.

#### 4.1.1.2 Library for Systems Solutions

The environments used here to categorize system solutions are expected to be more practical for the purpose of the *The Library for Systems Solutions*, that is, to identify the components that provide solutions to specific data processing problems in specific environments.

Any other categorization method should be easily mappable into the appropriate environment, if the necessity arises.

## 4.2 Non-programmable Terminal (NPT) Environments

Non-programmable terminal environments include solutions where the application uses local or remote terminals as if they were non-programmable terminals.

With these solutions the processing power is located in only one location or node, and, as a consequence, all the controls, the management functions, and

the applications processes are also localized in only one place, or to use a common term, centralized.

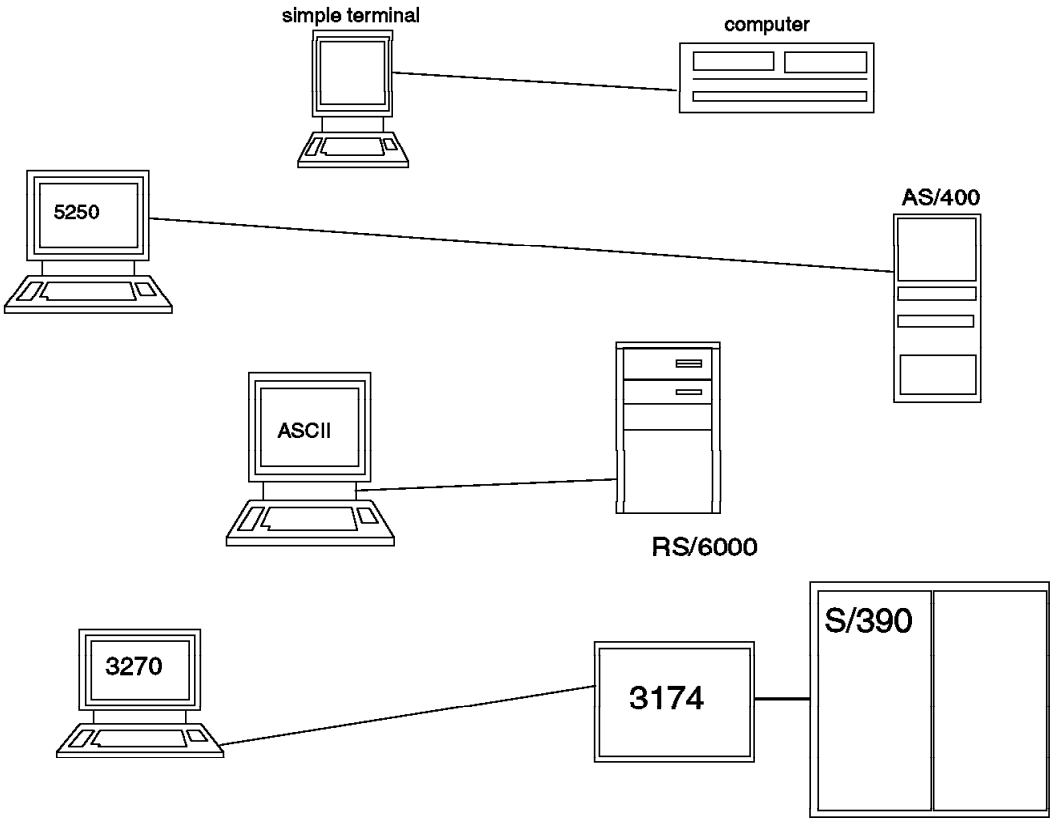


*Figure 42. Basic Non-programmable Terminals. Simple, or dumb, terminals are the traditional instance of a non-programmable terminal. Minor terminal intelligence, such as setup or customization ability, does not alter the general classification.*

Non-programmable terminals are the most basic forms of computer terminals. The standard examples are IBM 3270 terminals attached to a mainframe, or simple ASCII terminals attached to a UNIX** system. Figure 42 illustrates the simplest forms of this mode.

A non-programmable terminal application is a style of usage and is not restricted to *dumb* terminals. Figure 43 illustrates more non-programmable styles of usage. A personal computer executing a 3270 emulator program is, in effect, being used in a non-programmable mode. The personal computer might be coax-connected to its host system, might be on a LAN with a gateway to a 3270-oriented host, or an AS/400* host, or be one of many alternate configurations. Likewise, a telnet session, in a TCP/IP environment, represents a more complex instance of a nonprogrammable terminal mode of use.

In each of these cases, the particular application involved is using non-programmable terminals. The mechanics of connecting the terminals to the application may be complex -- LAN gateways, telnet, terminal emulators, and so forth -- but may not be relevant to the application solution.

This same physical environment, as shown in Figure 43, has other modes of use. The personal computers have their own local applications. There may be a file

server on the LAN representing a LAN mode of use, and so forth. The users switch between these modes of use as they switch applications throughout the day.

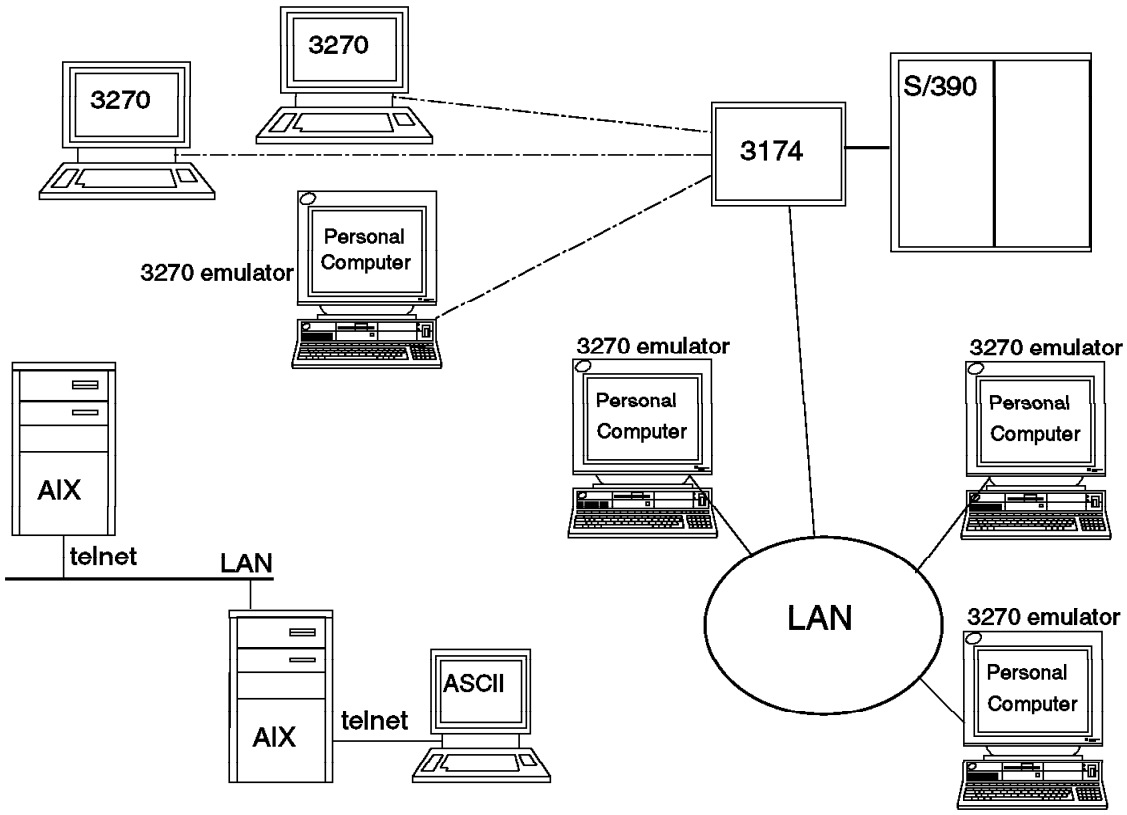


*Figure 43. Non-Programmable Terminal Equivalents. Intelligent systems can be programmed to function as non-programmable terminals. Such programs are usually called emulators. There are many personal computer programs, for example, to emulate the IBM 3270 family of terminals. From a design point of view, such systems, when used with an emulator program, are equivalent to non-programmable terminals.*

Planning a distributed solution for an application requires a clear distinction between the logical user environment for that particular application, and the physical environment being used.

Categorization is not always easy. Consider the environment shown in Figure 44. Special-purpose terminals, such as banking terminals, are typically connected to a local controller, and this controller is connected to an application host system. The special-purpose terminals may be programmable, but not by the immediate user, that is, not by the bank tellers, for example. They are, effectively, fixed-function terminals connected to an intelligent controller. The connections might be by point-to-point wiring or over a fixed-purpose LAN. For application planning, we would regard these terminals as non-programmable terminals. The controller, if it contains any significant amount of variable logic, is a WAN element, which is discussed later.

Another example of a fixed function *terminal* is the 3270 gateway, shown in the same figure. The gateway program is a complex product, but it is probably transparent to the design of the host application being used from the emulator sessions in the personal computers.

One common characteristic of non-programmable terminal applications is a large amount of time, effort, and code devoted to screen formatting and management. The basic mapping support (BMS) of CICS* is an example of this. A more striking example is the large amount of code in Microsoft Windows or IBM OS/2* devoted to the graphics user interface. From the viewpoint of a personal computer's processor, the computer's display adapter and display constitute a non-programmable terminal.

#### 4.2.1.1 Other Considerations

Security concerns in this environment tend to focus on good authentication of users; that is, the focus is on passwords and password control. Communication security is often ignored. This may be reasonable for leased line, or for direct or coax connections, but it may not be reasonable for LAN communications. Current technology is moving many physical terminals, in the guise of terminal emulators on various workstations, to LANs, but without considering the security implications of this change. LANs have a unique security exposure in that every workstation on the LAN can potentially monitor all the data traffic on the LAN.

Administration is completely centralized for this environment. User administration is usually separate from communications and terminal administration.

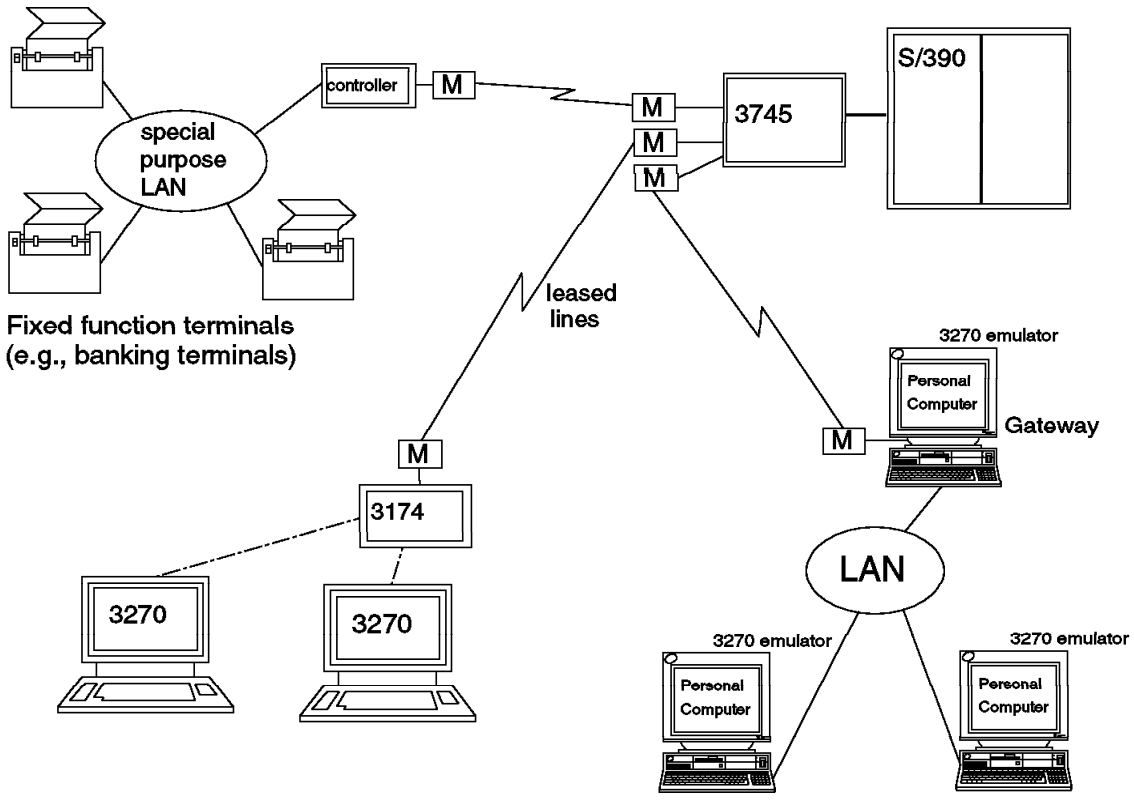


*Figure 44. Non-Programmable and Fixed Function Terminals. A complex connection environment, possibly involving LANs, WANs, gateways, modems, and so forth, does not change the design point of non-programmable terminals. Fixed-function terminals, not changeable or programmable by their users, can fall into the non-programmable category, even though the terminals were programmed to produce the fixed functions.*

#### 4.2.1.2 Summary

Non-programmable terminals, excluding emulators on workstations, have one particularly strong advantage; they represent the simplest user environment. With appropriate programming, and design, the user might be able to switch on a terminal and have the application directly available. The user is not required to own and maintain a local system (a PC DOS system with a 3270 emulator, for example).

## 4.3 Wide Area Network (WAN) Environments

This category includes solutions where the application makes use of remote terminals that have some level of data processing capability.

Any network application involving some form of leased lines can be considered a wide area network (WAN) environment. This could include, for example, a bridge linking two LANs, or an IBM 3174 Control Unit connecting IBM 3270-type terminals to a remote host. This definition of a wide area network is so general that it is not very useful. It tends to reflect the physical environment, rather than a mode of design and use.

We will adopt a narrower sense of a *wide area network,* using one or more of the following guidelines:

- It is a network application composed of a host and one or more remote, limited-function controllers connected to the host by leased lines. The

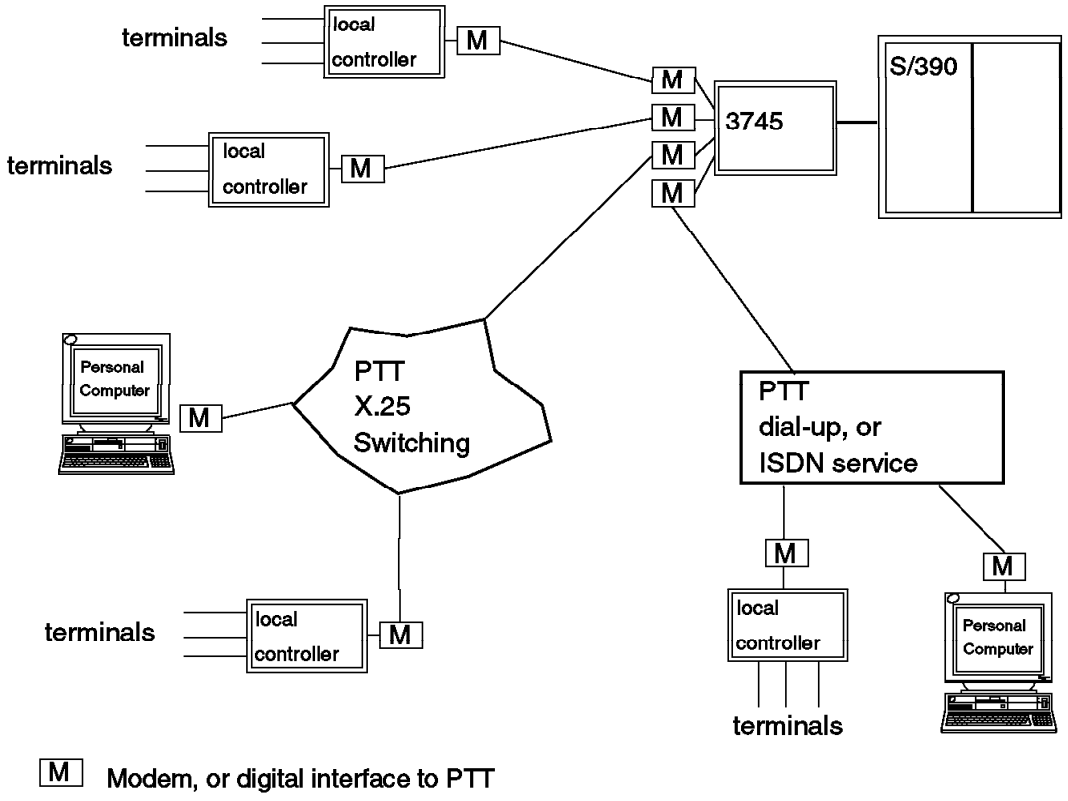


*Figure 45. Typical Wide Area Network Environment*

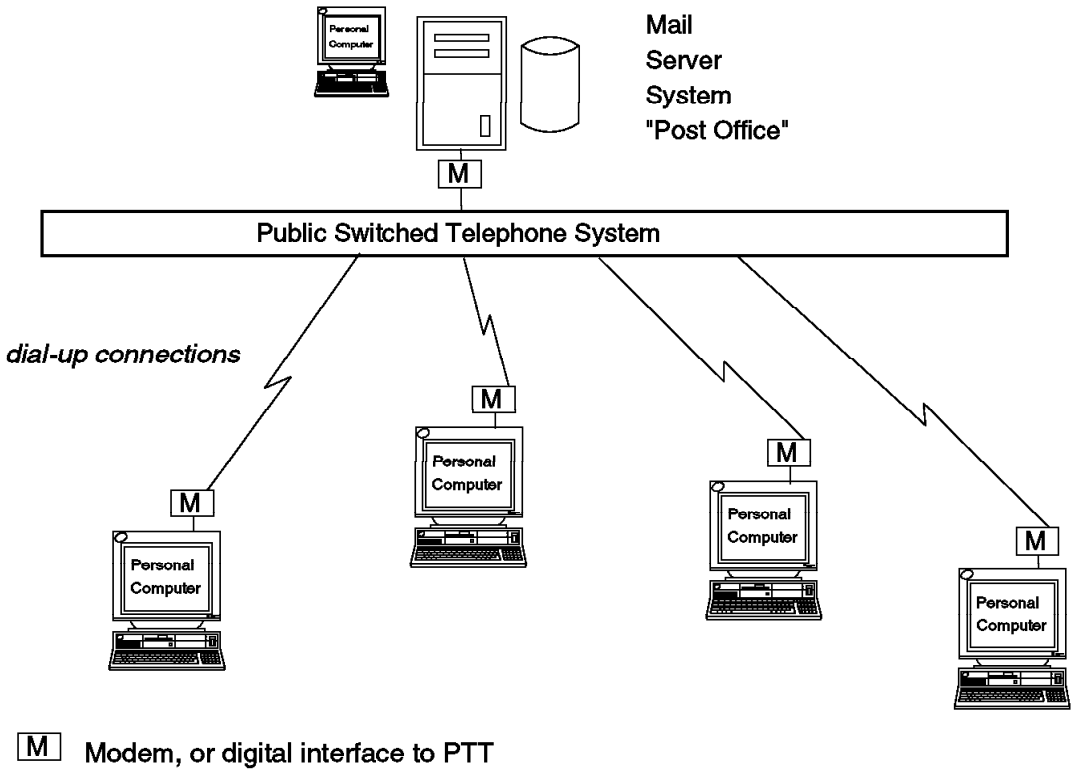


*Figure 46. Wide Area Network Mail System. Dial-up bulletin boards are a more general case of a WAN system. A mail system is a specific instance of the more general case, optimized for the single purpose of mail handling.*

limited-function controller implies the existence of a master/slave relationship. Good examples are bank controllers, IBM 3174 Controllers, and so forth.

- It is a network application requiring explicit use of dial-up functions. An example would be dial-up electronic mail systems.
- It is a network application depending on switching functions, as opposed to mere leased channels by a public carrier.

A dividing line between *wide area network* environments and *interconnected (peer-to-peer)* environments can be rather arbitrary, and no real purpose is served by attempting to create too many definitions. Likewise, the separation line between fixed-function terminals (usually implemented with a local controller), and a WAN local controller is arbitrary, and is usually decided by the nature of the application under consideration.

Figure 45 illustrates typical WAN application environments. A WAN, as the term is used in this document, contains one or more of the elements shown in the figure. The *local controllers* imply a lower-level functionality that is used in a master-slave relationship with the central system. The public switched system, whether it is X.25, ISDN, dial-up, or other, may be a central point in a WAN design. Depending on the application design, either the central system or the remote systems may initiate the switched connection. A System/390 is shown here, but the central system could be almost anything -- AS/400*, a UNIX** system, a large personal computer -- in which the application is designed for a

WAN environment. Figure 46 illustrates a completely different type of WAN application, based on dial-up connections to a simple server. This could be generalized to dial-up access to any *bulletin board* type of server, or dial-up access to systems such as Prodigy or Compuserve. If one considers all the existing personal computers with dial-up modems, this type of WAN may become important for many applications.

#### 4.3.1.1 Other Considerations

Security concerns are usually limited to authentication controls. Communication, with encryption, for example, is usually ignored except for special situations, such as bank automatic teller machines. In general, both leased lines and the public switched line network are perceived as being inherently secure enough for most applications.

There is no dominant programming characteristic for WAN applications. Our WAN definition is based on a style of communication, not on a style of programming or terminal management. Our definition assumes some system intelligence at both ends of a WAN connection, as opposed to a dial-up non-programmable terminal connection, and is usually a more complex user environment than that of a simple non-programmable terminal.

Administration of a WAN application or environment is centralized, from the viewpoint of the WAN application. The dial-up systems in Figure 46, for example, will each have their own local administration, but the central *post office* application will be administered centrally.

## 4.4 Local Area Network (LAN) Environments.

This category includes solutions where the systems, the workstations and the personal computers used by the application are connected to form a local area network.

Compared to WAN environments, and ignoring other substantial elements of difference, the LAN features shorter distances and higher communication bandwidth among its elements. Every station in a LAN has its own data processing capability, and therefore, the application processes in a LAN environment have the potential to be distributed.

Most of the solutions that exist for the LAN environment assign to some stations the role of servers of requests for services coming from other stations, or the clients. Because of its characteristics, a LAN environment is also sometimes referred to as a client/server environment or a workgroup environment.

The starting point for almost any LAN environment is the simple file server, as shown in Figure 47. This is offered by products such as the IBM LAN Server, the Microsoft LAN Manager, the more basic elements of Novell′s Netware**, and so forth. Although known as file servers, these systems are also print servers and can usually share other serial devices.

A basic LAN file server is the key element for many PC-based installations. It provides:

- Common databases for applications.
- The base for specialized shared data applications, such as many electronic mail *post offices*.

- Some degree of data security, since logon and authentication is an option of all file servers.
- Shared application code, greatly simplifying software distribution and maintenance.
- Centralized point for operational functions, such as regular backup of data.

LAN servers are not limited to simple file server functions. Figure 48 illustrates some extensions. Novell′s Netware**, for example, allows additions to the server code to perform other application functions. LAN server products that extend significantly beyond the simple file/printer server area are often called *network operating systems* (NOS). This term, *network operating system,* can be ambiguous. It can mean:

- A software system, residing on a base operating system, that provides operating system-like functions at the network level: IBM′s LAN Server is an example of this type.
- A self-contained operating system that only provides network functions to client systems running other operating systems: Novell′s Netware** is an example of this type.
- A mixture of the previously mentioned systems, functioning both as a base operating system for a local user and as a provider of network services: Microsoft′s Windows NT is an example of this type.

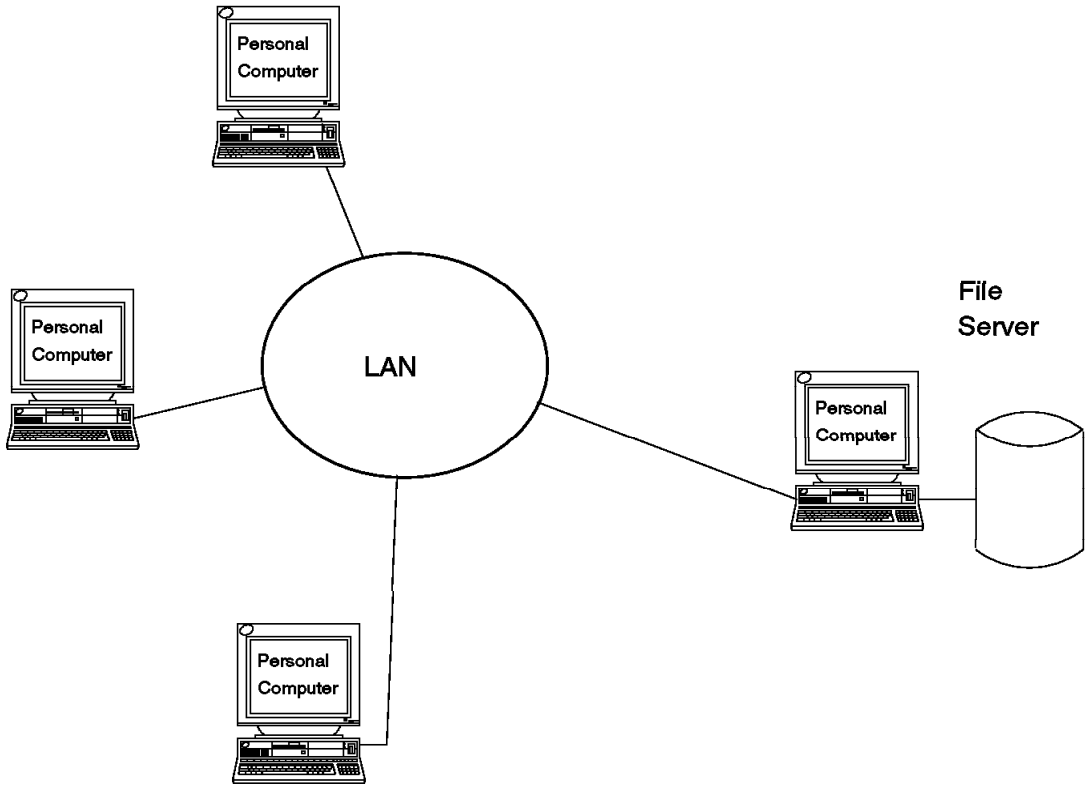


*Figure 47. Simple LAN System. The most common simple LAN environment consists of personal computers using DOS, Windows, or OS/2*, and using a LAN server as a file server. The LAN server is usually dedicated to this function.*

#### 4.4.1.1 Other Considerations

Security and user administration tend to be major problems for LAN based applications and systems. Each node on the LAN may have its own security system, although the most common node, DOS, normally has no security; and the servers have their own security, administered separately for each server. In addition, communication over the LAN has widely-known eavesdropping exposures. LAN authentication (password submission and verification) methods can be complex, because in-clear passwords sent over the LAN represent a significant security exposure. Larger environments often have multiple servers offering different functions. Different server applications may incorporate different styles of administration, making general administration more of a challenge.

There is a tendency to centralize authentication security in more complex LAN environments. DCE** is the primary instance of this, using Kerberos functions for a centralized security server. Novell's NetWare 4.0 addresses part of the problem in a different way, by automatically linking multiple server directories.

Local Area Network (LAN) wiring is rapidly replacing all other forms of computer connections within buildings. *It is important to distinguish the LAN as a communications carrier, from the LAN as an application element.* Non-programmable terminals, for example, may use a LAN for connectivity, but this is transparent to the application involved. Likewise, peer-to-peer protocols, such as TCP/IP, also use LANs for connectivity.

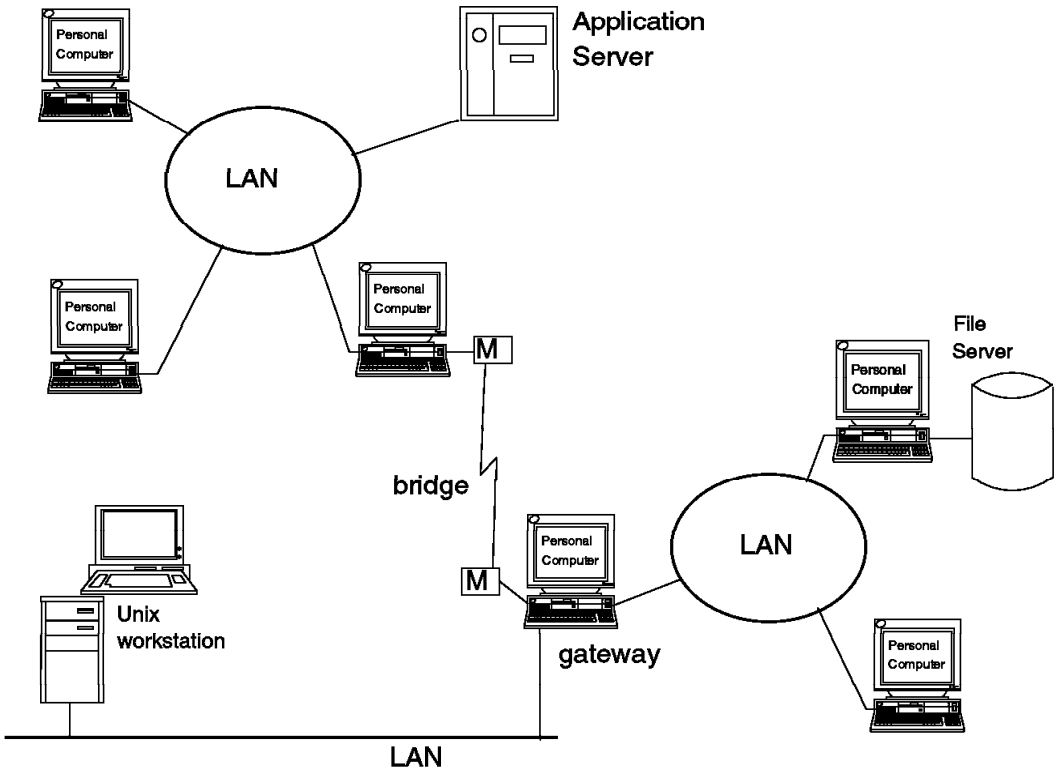


*Figure 48. More Complex LAN System. Two LANs can be bridged together, appearing as a single LAN for users and applications. Multiple servers, offering identical or differing services, can appear on the LAN. Some servers offer functions useful on several different operating systems.*

#### 4.4.1.2 Summary

Basic LAN-oriented applications are usually written in client-server mode, using netbios or TCP/IP protocols, including RPC. This gives the user a powerful, but more complex, environment. Each LAN node is a peer-level system, and users are expected to resolve minor difficulties with their local node and basic LAN functions.

LAN applications (client-server mode) tend to concentrate on passing arguments and data correctly. Presentation of data (screen formatting) is usually left to the client system and, in some cases, may not be considered part of the LAN application. LAN application design usually ignores LAN data transfer rates, as long as the amount of data being transferred is reasonable.

LANs may be interconnected through bridges, connecting portions of the same LAN; or through gateways, connecting separate LANs; or through combinations of these functions. These physical arrangements are usually transparent to LAN applications.

## 4.5 Multi-level Server Environments

The mutli-level server environment includes the solutions where the applications based on one or more LANs require the services of one or more higher level servers, hierarchically arranged, in addition to those defined on the LAN itself.

Several CICS* systems, as shown in Figure 49, are an excellent example. The CICS* intersystem communications (ISC) feature can be used to send various elements of a transaction to other CICS* systems. A transaction application might execute in CICS/6000*, except for database accesses which are *function shipped* through ISC to another system.

A multi-level server design does not presuppose a particular hardware LAN design. A multi-level server application might exist in a single, simple LAN or might be spread throughout many interconnected LANs. The hardware environment is independent of the software application design.

Multi-level servers can be important in producing a total solution for a complex application. They can be used in distributed applications in a variety of ways:

- A local database can act as a cache for a higher-level database, improving local query performance while retaining a single master database for the enterprise.
- Out-of-territory database requests can be managed without excessive special-case programming.
- The conflict between multiple, distributed databases (as part of right-sizing) and a single copy of information elements can be partly resolved. For example, a departmental database may contain inventory information, while certain applications must access a corporate database for the authoritative copy of a suppliers shipping address.

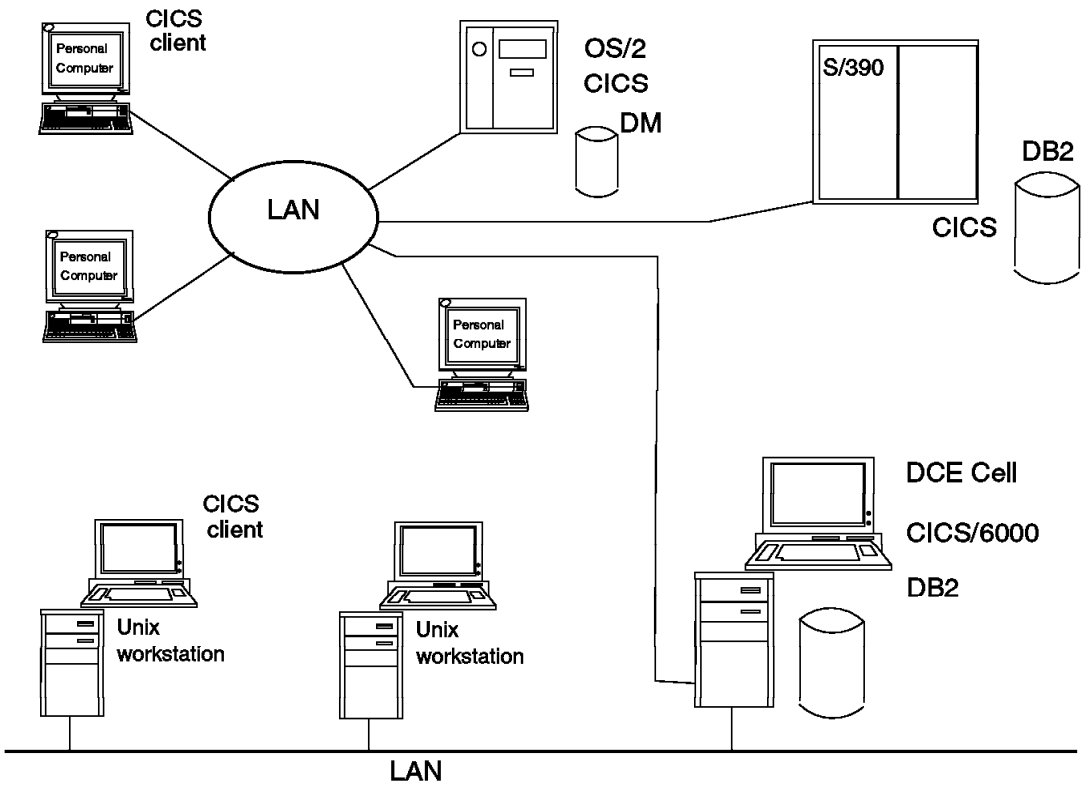


*Figure 49. Multi-level Servers*

Some uses of hierarchical servers are implied in other system functions, such as DCE** name servers. This is transparent to other applications and is not a reason to categorize an otherwise simple LAN application as a multi-level application.

Most security concerns for this environment are about the same as for a basic LAN environment. More attention may be needed to provide consistent user definitions on the servers used in the hierarchy, but this falls within normal administrative actions.

One security concern is unique to this environment. Suppose user JOE requests a service from SERVER1. JOE will be known and authenticated by SERVER1, perhaps using DCE** or a simple server logon. If SERVER1 must request help from SERVER2, to satisfy the request of JOE, what is the security environment of the request from SERVER1 to SERVER2? With DCE**, SERVER2 authenticates SERVER1 as its user; the original client, JOE, is not relevant to SERVER2 in this case. (Of course, specific application coding can change this, but security should be left to the *system* as much as possible.) Therefore, SERVER1 to SERVER2 authorization must be high enough for any potential user, and SERVER1, in its application code, must decide whether JOE is permitted to use SERVER2 indirectly. Other distributed designs might pass user JOE through to SERVER2, but this has another set of administrative and practical problems.

## 4.6 Other Interconnected Systems and Peer-to-Peer Environments

This category includes the solutions where multiple systems are interconnected to provide functions that are inherently peer-to-peer, such as NJE networking between MVS systems; or are used by end users in such a variety of ways that they can only be categorized as interconnected systems.

The standard set of TCP/IP applications is an example of the latter and is illustrated in Figure 50. Individual standard TCP/IP applications and functions can be categorized in more specialized ways. For example, telnet emulates non-programmed terminals; NFS is a file server; ftp is either a file server-type or peer-to-peer application, depending on how it is used; and so forth. More sophisticated users tend to use many of the standard applications and, as a group, the TCP/IP applications can only be categorized as *interconnected systems*.

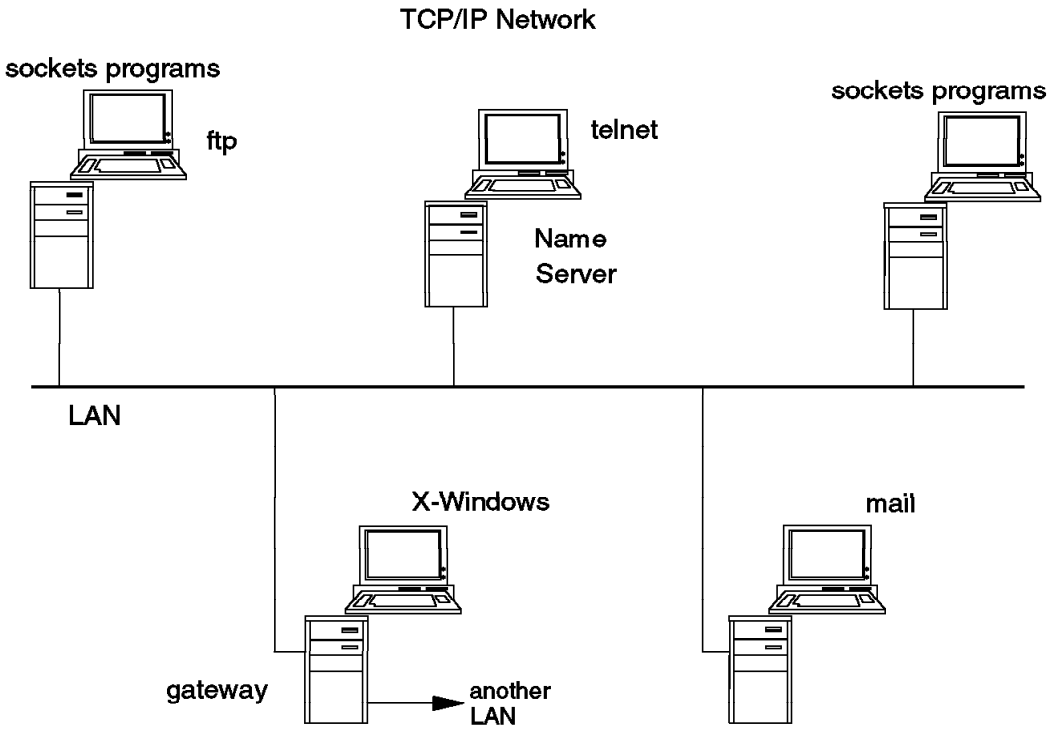


*Figure 50. TCP/IP. The standard TCP/IP applications, such as telnet, ftp, NFS, mail, rsh, tftp, DNS, X Windows (usually), and so forth, are often seen as a single package.*

Multiple CICS* systems are often connected in true peer-to-peer applications, in which each system is sometimes a client and sometimes a server. This can be generalized to LU6.2 (APPC and CPI-C) applications, although full two-way applications are rare.

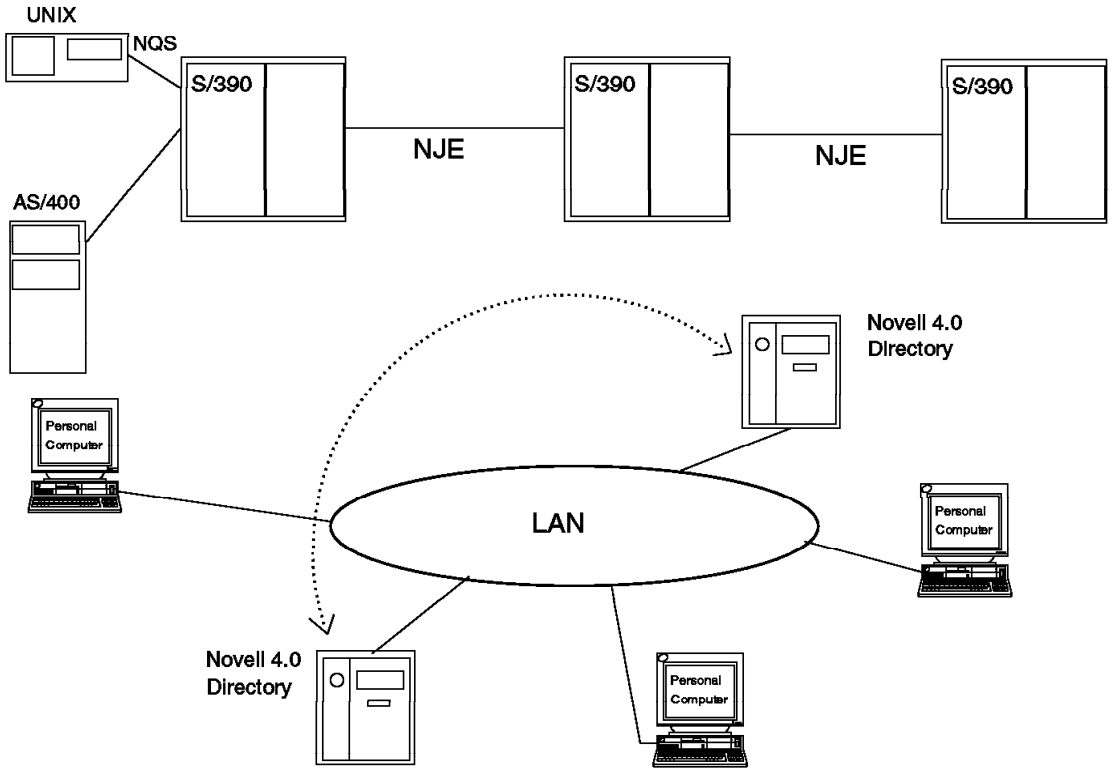


*Figure 51. Peer-to-Peer Applications. NJE (and RSCS) networks between hosts are pure examples of peer-to-peer links. There is no sense of a master-slave relationship, and many varieties of requests and data flow over waiting links. Peer-level directories (or other services) of network operating systems are another example of generalized interconnected systems.*

OS/400* offers excellent peer-to-peer functionality. Users, functions, and jobs can be distributed throughout a network of AS/400* systems. This built-in distributed processing capability is an important competitive advantage of the AS/400*.

Peer systems usually have their own administration, with a small add-on for administrating the peer-to-peer connection. There is some security administration for the connection itself. The peer systems usually authenticate the identity of the other system when the peer connection is established. User authentication may or may not be required; peer systems may trust other peers to authenticate users. The major application(s) using the connection may or may not apply further security controls to specific requests that arrive over the connection.

Peer-to-peer programming is concerned with conversation control, parameter sequences, and data, but seldom with presentation (screen design) of data. Programming is more complex than for client-server code, because the *direction* of the connection must be managed by the programs.

Figure 51 illustrates two examples of pure peer-to-peer systems: an NJE network, and peer-level directories of a network operating system. An application that is explicitly designed to use these properties has a peer-to-peer design.

The standard example of a peer-to-peer system, based on LU 6.2, is actually a description of the communication method, not the application design. If the application is designed for two-way, peer-level work exchange, then it is a peer-to-peer application. Many applications using LU 6.2 can be more accurately described as simple server (WAN or LAN) applications.

Peer systems are often administered separately; that is, there is no single, central administration database of users, passwords, security controls, and so forth. This is not a requirement, but is a typical characteristic. Design work often centers around *transaction codes* or *commands* that the peer systems will recognize when received from other peers. Data exchange is often in binary blocks, without regard for presentation formatting.

## 4.7 Other Application Environments

All real-world application solutions will not fit neatly into the categories described in this section. Nevertheless, these categories are useful for two reasons:

- Existing solutions, products, methods, tools, design techniques, and expertise are very often centered in one of the categories described here.
- Complex problems can often be subdivided along the lines of these same categories.

Understanding some of the application solutions available within each of the categories can be a tool for developing more complex solutions for specific situations. This approach has a fundamental requirement: categorize systems by their mode of usage, not by their physical components. In some cases, the mode of usage matches the underlying components, but in many cases, especially when more modern technology is involved, the two are almost unrelated.

If you attempt to categorize solutions, both known and required solutions, by a mixture of physical technology and modes of use, you will have a large number of permutations and combinations. The usefulness of the categories as a design and organization tool becomes questionable.

# Chapter 5. Software Environments

Distributed systems design should be a management concern, especially if business-critical applications are involved.

Information processing technology is changing very rapidly, and many of the new components are attractive to computer people simply because of the new and fascinating technology. This fascination, compounded by trade-press impressions that everyone else is already using the newest technology, creates strong pressures to use the latest technology, simply because it exists.

Distributed processing should be used to further the business needs of a company. It should not be used as the justification to simply chase the latest technology.

Distributed processing is a key component of rightsizing, and can be very successful when planned well. Planning it well involves much more than buying a few LAN servers.

Distributed processing is a general concept rather than a specific plan. The concept covers a large range of possible designs. A key design question is: "What is being distributed?" Choices are:

- **Users.** Users are the easiest element to distribute, and this distribution has been happening for many years. New technologies, especially LANs, have helped the process by providing higher bandwidth and more responsive channels to the distributed users.
- **Processing.** The potential for distributing processing has been present since the development of personal computers and other workstations. At one time, the concept was known as cooperative processing. The current term is client-server, although this term may include more than just distributed processing.
- **Data.** Distributed data is, at first appearance, an obvious goal of distributed processing. It involves complex considerations that are not immediately obvious, and it is probably the most difficult element in any design.
- **Management.** Distributed management includes many topics, such as user management, network management, security management, capacity management, and so forth. The need for distributed management exists partly because the other forms of distributed processing exist. This requirement is often stated in reverse terms, that is, a desire for centralized management of a distributed environment. However, this centralized point of management might be exercised from any point in a distributed network.

A well-planned distributed processing implementation will have a clear vision of what is being distributed and why it is being distributed.

Software, not hardware, is the key element in distributed design. Almost any combination of hardware can communicate, to some degree.

Software environments, or platforms, have at least two dimensions: a basic operating system, and higher-level interfaces added to the basic operating system. For example, UNIX** is a basic operating system, and DCE** is a higher-level set of functions that can be added.

Families of operating systems exist where there is considerable functional and file compatibility within a family. The higher-level interfaces provide certain degrees of compatibility across operating system families. This chapter briefly discusses several of the operating system families and some of the compatibility considerations involved when a distributed application crosses multiple families.

As discussed earlier in this document, a software environment is composed of many layers, as shown in Figure 52, for example. Any large application solution will use interfaces with several different layers of the software environment. For example, a distributed OLTP application might use functions of Encina**, DCE**, DB2, and AIX*.

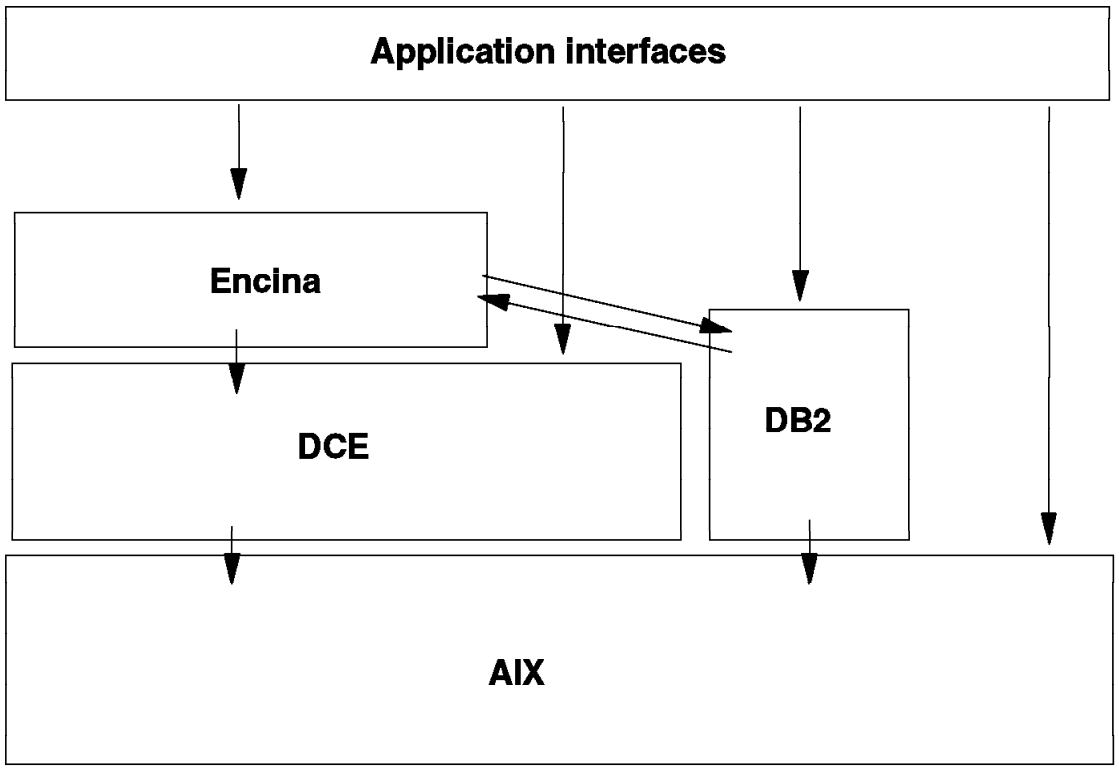


*Figure 52. Software Layers*

The operating system is often considered the lowest layer of the environment. Even lower levels, such as device drivers, are extensions of the operating system. Higher levels can provide common functions, or shield the application from some of the particulars of a given operating system, or both. However, it is unlikely that the higher layers will completely shield an application from all the characteristic features of the underlying operating system. The transcendent features include, among others:

- Symbolic file system, if one exists
- The nature of file names, such as absolute names, as with MVS; verses path-related or relative names, as with UNIX** or DOS.
- File organizations, such as record, byte, or keyed
- Security system structure

- Locking mechanisms, controlling multiple accesses to resources
- Apparent memory organization, for sharing data with multiple tasks
- Program loading functions, for using multiple program modules
- Operator interfaces and functions.

In a few cases, CICS* transaction programs are the best example, the characteristics of the underlying operating system are completely hidden from application programs. But, this is rare. Most realistic applications are influenced by the characteristics of the underlying operating system, especially when all the elements of the application solution are considered. Functions and characteristics of various operating systems must be considered when planning a distributed application.

The following categorization of operating systems is useful for planning distributed applications:

- Mainframe software environments, such as MVS and VSE
- Proprietary, multi-user systems, such as OS/400* and DEC′s VMS**
- UNIX** and UNIX**-derived systems
- Network operating systems, such as NetWare**
- DOS and DOS-related systems, such as Microsoft Windows, Windows NT**, and OS/2*
- Standardized higher-level platforms, such as CICS* and DCE**
- Other systems not related to these general families, such as VM/ESA*.

Distributing an application solution within one of these groups is generally easier than distributing across several groups. The problem is not just programming interfaces, but includes many working assumptions, training, intuitive expectations, conflicting meanings of terms, style of usage, and so forth.

## 5.1 Mainframe Software Environments

While there are a variety of mainframe platforms and operating systems available, the field is dominated by MVS, and only MVS is considered here. Key characteristics of the MVS software environment are:

- Symbolic file names, through JCL. This provides a standard approach for manipulating applications with many files.
- Record oriented file system, with various access methods, including VSAM.
- Absolute file names, with a strong volume orientation.
- An external security reference monitor, usually RACF. Security control is heavily oriented to generic names and group structures. Mainframes, with MVS and RACF in particular, offer complete security and integrity structures and controls.
- There is no single user interface to the system. CICS* and TSO are the most common interfaces. ISPF under TSO is the most common non-OLTP menu interface, but many other external faces for MVS can be found.
- COBOL programming, with smaller subsets of PL/1 and assembler language programming. C compilers exist, but are not widely used.

- Database usage is common, including DB2* and DL1. DL1 is unique to mainframes; therefore, applications built around DL1 may be difficult to distribute.
- Elaborate and exacting accounting mechanisms. Local programs, or a variety of program products, are required to make use of the data. Some system programming may be necessary to customize the data collected.
- Sophisticated systems management tools and procedures.
- Large body of (COBOL) CICS* transaction programs. CICS* processing, including back-end work and associated database functions, often is the dominant workload.
- Complex terminal networks, working through VTAM, almost completely oriented to 3270-type terminals.
- Very strong batch processing orientation, with well defined operator functions.
- Major printing subsystems, driving multiple high-speed page printers, with excellent spooling functions and control.
- Many tuning parameters are available at various system and application levels.
- Very detailed trace and debugging tools are available; although, to use these might require considerable expertise and, possibly, some specialized programming.
- MVS is a mature operating system, with basic compatibility extending over decades. System software is optimized for performance, availability, integrity, and maintainability.
- Many, if not most, applications used on a mainframe are developed in-house by a resident programming staff.
- The EBCDIC character set is used. The common code points, as provided by 3270 terminals, do not include some of the common C, DOS, or UNIX** characters.

## 5.2 Proprietary Midrange Software Environments

The primary example here is the AS/400* family. Key characteristics of the OS/400* software environment are:

- A generalized symbolic file system is not available, although limited substitutions for hard-coded file names can be made through the command system.
- Record oriented file system, including keyed access.
- AS/400* security is excellent, but the methods used are unique to the AS/400* architecture. Matching AS/400* security elements with those of other operating systems is difficult.
- User access is through a standard full-screen menu system. This menu system provides the external appearance of OS/400*.
- Single-level addressing, including all files.
- Absolute file names, with no volume orientation. File names can include or omit an associated library. If a library name is not specified, a controllable search path through multiple libraries can be used.

- RPG and command-level programming. COBOL is also used, but at a lower level. C compilers exist but are seldom used.
- The transaction processing terminology is relatively new for this system. The basic OS/400* system can be used as a low-volume transaction processor, by entering a program name on the command line. There is increasing interest in this area, and a new CICS/400* product offers main-stream transaction processing facilities.
- Database functions are built into OS/400*.
- Moderate accounting and auditing facilities are built into the operating system.
- System management menus and functions, sufficient for most installations, are a standard part of OS/400*.
- Terminal networks are usually limited to local terminals. OS/400* has very good peer networking with other OS/400* systems.
- An acceptable batch processing environment exists, with operator functions.
- High-volume printing is possible, but not usually done on these systems.
- Many applications are purchased, and in-house programming staffs tend to be minimal.
- The PC support function is widely used, providing a distributed environment for advanced users.
- Some tuning parameters are available at various system and application levels.
- Very detailed trace and debugging tools are not commonly used.
- OS/400* is a relatively mature operating system.
- The EBCDIC character set is used. The common code points, as provided by 3270 terminals, do not include some of the common C, DOS, or UNIX** characters.
- Terminals are often twinax types unique to AS/400* systems, although 3270-type terminals are also supported.

## 5.3 UNIX** Software Environments

UNIX**-based operating systems are available from a wide range of vendors, and with a wide range of features. There is no single definition of what constitutes a UNIX** operating system. Elements and features that tend to make the operating system more robust are being added. However, the core portion of all UNIX** systems includes these characteristics:

- UNIX** has no symbolic file system. File names are passed as calling parameters, derived from calling parameters, or hard coded. This works well when only one or two files are involved, but works poorly when a large number of files, possibly with unrelated names, are involved. Applications often assume their files are located in certain directories, or pass directory names through environmental variables.
- A byte-level file system is provided. Any record-level functions are up to application programs.

- File names include a location-dependent path name, or are relative to the current directory. Moving a file from one disk to another causes its full name to change. A file can have several names.
- UNIX** security is in line with various system functions, such as *login* and *open*. Specific security permissions are associated with every file. Some versions of UNIX** offer ACLs (Access Control Lists), in addition to the basic file access permissions, but these are also tied to individual files. Security through generic names is not commonly available. In principle, a secure UNIX** system can be installed and maintained; in practice, this is difficult for a variety of reasons.
- The UNIX** command line and, to a slightly lesser extent, a window-manager window, under X windows, are the standard *external faces* of the system.
- Most programming is in C, although FORTRAN, COBOL, and other compilers are widely available.
- Transaction systems (OLTP) are not commonly used, although this area is expected to expand rapidly.
- Complex terminal networks are almost unknown. Complex system networks are common, usually based on TCP/IP.
- Batch processing functions are primitive.
- No database functions are included in basic UNIX**, but many third-party products are available.
- UNIX** can, optionally, collect accounting information, but the accuracy and repeatability does not match that of mainframe systems.
- Native UNIX** system management often consists of editing stanza files. Many UNIX** vendors have added higher-level system management tools, such as IBM′s SMIT. These tools differ greatly among vendors. UNIX** system management tools are important factors for many commercial users, where skilled UNIX** programmers are not always available.
- Printing subsystems are primitive and usually do not provide full spooling.
- Relatively few tuning parameters are available at various system and application levels. Tuning tends to be a matter of application design.
- UNIX** is still undergoing changes at several levels and from several sources. It is not a mature operating system in the same sense as MVS or OS/400*.
- Good interactive application debugging tools are available, but system-level trace and debugging tools are sporadic and nonstandard.
- There is a large market in third-party application products, and major applications or subsystems are often purchased. Nevertheless, there is almost always some amount of programming activity in any installation.
- The ASCII character set is normally used, providing good exchange with other ASCII-based systems.
- Terminals are either workstation graphics terminals or ASCII terminals. 3270-type terminals are not supported.

## 5.4 Network Operating Systems Software Environments

A Network Operating System is not a fully defined concept. It is a moving definition as technology develops, but it is not, as yet, a full operating system in the traditional sense. The general premise is a high-bandwidth, shared connection with all the units in the network. A token-ring LAN or Ethernet LAN are the most common examples of network hardware. The basic element is always a file server. The user's local system, his personal computer using DOS, for example, automatically redirects certain disk requests to the file server instead of to the local disk. Novell's NetWare**, IBM's LAN Server, and Microsoft's LAN Manager are common examples of a network operating system.

The basic concept is simple. The key element is that the file server can permit users to share files among themselves. The file server function includes:

- Security controls, governing which users can access which files on the server.
- Mapping functions, to cause certain directories or disks on the server to appear as logical disks on the user's systems.
- Locking services, to allow controlled sharing of files.

At the first level, the programs that use the shared files are executed in each user's system. That is, these application programs are not executed in the server or under the network operating system. The only program executing under the network operating system, in the server, is the program to respond to file server, mapping, or locking requests from the client systems. A program executing in a client system might reside in a file that happens to be on the server, but the program is read and executed by the client system. More advanced server applications can execute in the server, but this is definitely an advanced level use of the server.

Thus, to a large extent, the use of a file server is transparent to many applications and users. The key characteristics of the client's operating system are not changed by the use of a server. For example, the DOS or UNIX** characteristics concerning file naming, user interfaces, record interfaces, programming languages, and so forth, are not changed by the presence of a network operating system as a server.

Products, such as electronic mail systems, that are built on shared files in file servers have intrinsic security and integrity exposures that must be understood. These applications usually require that all users be permitted to write to certain selected shared files. The application solutions offered by these products can be very attractive, and the security and integrity exposures can be an accepted risk. However, the risk should be understood before it is accepted.

In practice, servers offer more functions than the base file-server functions listed above. For example, almost all commercial products offer print server functions, allowing clients to redirect print requests to the server. A new term, *application server*, is being applied to some systems that might be regarded, at least partly, as network operating systems. Novell**, for example, uses this term with their UNIXWare product, and the trade press applies the description to Microsoft's Windows NT**. So far, the term application server appears to describe the use of the system, rather than the fundamental nature of the platform.

A network operating system environment implies the following characteristics:

- A requirement to log onto a server or onto more than one server.
- The availability of additional logical disk drives, provided by the server.
- The availability of some applications implemented through the shared files of the server. Applications often include electronic mail and relational databases.
- The availability of application programs that do not depend on shared data files on the file server disks. This removes the need for all users to have application copies on their local disk, saving disk space and making maintenance easier.

Networking products, whether or not part of a network operating system, share a common problem in managing software updates. For example, electronic mail systems usually have components that run in each client′s system, and other components that run in servers. Still other components are passive files, forming a post office in file servers. There is a potentially severe problem in installing software updates. If the updates are downward compatible, the problems are minor. Not all updates are downward compatible, however, and the network or application manager faces the task of installing new software files in all clients and servers at the same time.

User and security management, at the enterprise level, is a well-known problem with network operating systems. In the extreme case, each LAN manager or NOS must be administered separately and may require users to log onto each one separately. Newer systems, such as Novell′s Netware** 4.0, are solving some of these problems. Products that provide administration across systems, such as IBM′s LANRES, provide additional help in this area.

## 5.5 DOS and Related Software Environments

Due to the huge number of personal computers sold, DOS, from various vendors, but all based on Microsoft DOS, is the most widely used operating system. Microsoft Windows is an extension of DOS that adds significant functions. IBM′s OS/2* and Microsoft′s Windows NT** are advanced operating systems that are derived from DOS and share some common characteristics.

Key characteristics of a DOS software environment are:

- DOS has no symbolic file system. File names are passed as calling parameters, derived from calling parameters, or hard coded. This works well when only one or two files are involved, but works poorly when a large number of files, possibly with unrelated names, are involved. Applications often assume their files are located in certain directories, or pass directory names through environmental variables.
- A byte-level file system is provided. Any record-level functions are provided by application programs or system software extensions such as database managers.
- File names include a location-dependent path name, or are relative to the current directory. Moving a file from one disk to another causes its full name to change. A file cannot have multiple names.
- DOS has no native security protection. Several security products are available, including IBM′s Secure Workstation Manager, each with different features and techniques. These products are not widely used, however. OS/2* has no native security functions. IBM has a statement of direction

indicating that security functions are a future goal. Microsoft Windows has no native security functions, but third-party products are available. These products are not widely used, however. Windows NT** is designed with extensive native security functions and features.

- Graphical User Interfaces (GUIs), such as Microsoft Windows and the OS/2* Workplace Shell, are becoming the standard external faces of the system, replacing the previous command level interface of DOS.
- Most programming is currently done in C. Visual programming and object-oriented techniques are becoming increasingly important.
- Transaction systems (OLTP), resident on these systems, are not widely used, although OS/2* does have a full CICS* product. CICS* and DCE** client packages are expected to be important future items.
- Terminal networks are not used and are not needed, because each PC has enough local processing power. System networks are common, usually based on LAN file servers.
- DOS, Microsoft Windows, and Windows NT** have no native database functions. Database functions are provided by Independent Software Vendors (ISVs) and are widely used. IBM provides DB2/2, a database manager for OS/2*.
- DOS, Microsoft Windows, and OS/2* have no native accounting facilities. Some third-party security products offer very basic accounting.
- DOS has very limited native system management facilities. Configuration and Distribution and Installation (CID) of code and data is a key factor in OS/2* and similar environments. Windows, Windows NT**, and OS/2* have substantial systems management tools, all with graphic interfaces.
- Batch processing is rarely used in DOS family software environments. OS/2* provides the powerful procedures language, REXX.
- Printing subsystems are limited, but on Windows and OS/2* they are powerful and provide full spooling.
- Relatively few tuning parameters are available at various system and application levels. Tuning tends to be a matter of application design.
- Most applications are purchased as packages. Corporations are now developing many in-house applications as well.
- Excellent interactive application debugging tools are available for both Windows and OS/2*. System-level trace and debugging tools are more limited.
- The ASCII character set is normally used, providing good exchange with other ASCII-based systems.
- Today, nearly all PC video adapters have advanced graphics capabilities, and most desktop displays are in color. Most laptop displays are still monochrome. Terminals, such as 3270s, can be emulated.

## 5.6 Higher-level Software Environments

Some application bases, sometimes called subsystems or platforms, are extensive enough to create their own environment. CICS* is a good example. CICS* provides a complete environment for application programming. A well designed CICS* transaction program often can be moved across platforms, such as MVS, AS/400*, AIX*, OS/2*, HP-UX**, simply by recompiling it. This is very attractive for distributed processing because it allows for continuous redesign with minimal reprogramming.

Other higher-level environments are not as encompassing as CICS*, but are just as important for distributed processing. DCE**, from the Open System Foundation, is an important example. The DCE** environment addresses function calls, as from a client program to a server. The DCE** environment permits a server or a client to be moved among a variety of systems and platforms, without changing the client-server communication programming. Programming code within a client or server, for the most part, is not within the DCE** environment, and it will be sensitive to the underlying operating system platform. For example, if a server function is ported from AIX to MVS, a certain amount of reprogramming or redesign might be required. However, the clients calling the server will be unaware that the server is on a different system and platform.

TCP/IP sockets programming interfaces and CPI-C (or APPC, or LU6.2) programming interfaces have some of the elements of higher-level platforms. The communications interfaces and protocols are rigidly defined and are mostly independent of the underlying operating system. This eases the migration and porting of applications across systems and platforms.

Some data and file systems can be considered higher level platforms. Generic ISAM or VSAM products are available on many platforms, and these can ease the migration of applications. Likewise, database products, especially relational database managers using SQL-based access, can be regarded as higher-level software environments. These are key elements in migrating existing applications to a more distributed environment. Products, such as Oracle and DB2*, are available on many bases and help provide a platform-independent environment for database functions.

The POSIX** (IEEE 1003.1) definition of basic UNIX** and C functions can be regarded as an application platform, even if the programming interface is not as high-level as CICS* or SQL. Program code that conforms to this definition is more easily moved between various vendor's systems. X Windows, including Motif**, is widely regarded as a higher-level platform for graphics programming and is well known for interoperability across multiple platforms.

DCE** offers excellent authentication and communication security for distributed systems. DCE** has its own user registry that must be administered. For the near future, this is an additional task for system administrators. In the longer term, other platforms may link with the DCE** registry for common user management.

Good use of higher-level software environments is a key factor in successful migration to distributed processing. Hardware and software technology, especially for smaller systems, is changing very rapidly. Higher-level environments, such as CICS* and DCE**, help isolate applications from the more

complex areas of a specific hardware/software base, allowing for easier accommodation of new technology as it arrives.

## 5.7 Other Software Environments

Some software environments are unique enough that they cannot be included in other groups. IBM′s VM operating system is an example. Other operating systems can run under VM, each in its own environment. The VM environment usually implies the use of CMS, a single-user operating system, and mini-disks in a CMS format.

DEC′s VMS** is a widely used operating system that is not part of another family. VMS** is available on many different VAX-compatible platforms and on the newer *alpha* systems.

For the purpose of planning distributed systems, unique operating system environments might have to be considered as distinct platforms, subject to all special cross-platform limitations and considerations.

## 5.8 Planning Considerations

Extending the scope of users and applications from a traditional centralized system to a distributed system is a complex undertaking. There are compatibility and migration issues at many levels. The following list includes many elements of traditional data processing that are not directly related to programming. You can use the list to consider before and after conditions for a planned move to a distributed environment, with the focus on the whole enterprise, that is, programs, users, operational environment, management aspects, and so forth. The purpose is to ensure that everyone involved understands the implications of the planned changes. The elements include:

- **User terminals.** These are often non-programmable terminals (NPT) or personal computers emulating non-programmable terminals. Examples are IBM 3270 family of terminals, IBM AS/400* family terminals, and personal computers emulating these terminals. Too many different terminal types is disruptive to users (“Where is the CLEAR key?”), and unproductive for the enterprise. A move requiring the use of emulated terminals can be especially confusing to end users.
- **User shells.** A shell is the immediate program interface driving the user′s terminal. Examples are the TSO command line, ISPF, the AS/400* menu system, CICS* terminal control, a VTAM logon screen, and so forth. A programmer or sophisticated user will deal with many shells, while a clerical user might see only one. The shell terminology is not usually associated with traditional computing environments, but the functionality does exist and is important for planning. User shells can be written in an infinite number of variations and styles. There is a trade-off between powerful function and complexity of use. A person using only one shell will learn most of the options of that shell, while a person who must shift between different shells will often use only the lowest common denominator functions, and thereby lose productivity. A well-known shell, such as ISPF, provides an important comfort factor for many end users. Computer professionals, who routinely use many different shells, tend to ignore this factor when planning changes.
- **Local processing functions and applications.** Local processing is work that is done in the user′s terminal or workstation. Again, this is not part of

traditional terminology, but the effects are familiar to anyone using an emulator in a personal computer. The user switches between the emulator session and the local DOS or OS/2* session for different applications. Some applications, based on HLLAPI,[25] for example, may use both the emulation link and the local session. The move to rightsizing and distributed processing places growing importance on local processing. The right balance between local processing, departmental processing, and centralized processing is the general goal of rightsizing. Moving from a dumb-terminal environment to a mixed local/remote computing environment can be confusing for end users, especially if remote emulation, remote server use, and local applications are all involved.

- **Connection to server or host.** Traditional connections are usually point-to-point coax connected to a controller. The controller is attached to the main system directly through a channel, or through a leased line. In recent years, this has been expanded to include LAN connections to the controller, but the functional result is the same as a coax connection. Newer technologies offer many ways to connect terminals and systems, often in ways that are transparent to traditional systems. There is a strong tendency to confuse medium, LANs, for example, with services available over the medium. Concepts that are clear to computer professionals may be confusing to end users, unless they are explained very clearly.
- **Major applications.** Major applications usually justify the expense of the data processing operation. The operation of the enterprise may depend on these applications. Higher-level management is aware, to some degree, of these specific applications, and is concerned if they are not working correctly. It is important not to lose the sense of major applications if the processing is decomposed and distributed to several locations. The application function, if it continues to be important to the enterprise, should continue to exhibit functional unity. This requirement can be lost if a strong management presence is not part of any conversion plan.
- **Application development and maintenance.** Traditionally, applications have been developed in-house, involving large groups of analysts, designers, and programmers. This implies that application expertise is available from an organized, known source. Application development should be a responsibility, rather than a programming activity. Who is responsible if the general ledger process begins producing the wrong answers? Application design responsibility or ownership must not be lost when an application becomes distributed.
- **Batch functions.** Full batch operation implies a number of things. It assumes packaged jobs that can be run, without intervention, by someone who has little or no knowledge of the job. The batch operator is fully isolated from the batch application programs, the data, and control files, such as JCL. It often assumes a major queueing or spooling facility, such as JES2, to handle massive printing functions. It assumes a method of systems control to manage the workload. It assumes system facilities to manage file conflicts, so as to avoid simultaneous updates to a file. It assumes the ability of a job to request tape mounts as required. It assumes system integrity in the handling of the job, with automatic recovery and restarts where appropriate. Auditors prefer well defined batch environments; fully interactive applications are very difficult to audit. The concept of batch

[25] This is a High-Level Language Application Programming Interface that is available with many 3270 emulation programs.

operations, where the operators are several levels removed from the data and programs being used, is not well developed on some platforms. There is usually more focus on the interactive parts of an application. Many commercial applications have important elements that work best in a traditional batch environment. These must not be overlooked in conversion plans.

- **Files.** Files are sometimes called *data sets*, or *simple files*, or *flat files*. These are files that a program can open, read, and write directly, as contrasted with a database, where this is not done. File structure might be defined by the system in terms of records, keys, and so forth; or by programs, in which case the system sees only a string of bytes. In MVS, many file formats are available; VSAM is the most sophisticated of these. Files can be read by multiple users in a distributed manner, through file servers, NFS, DFS, and so forth; multiple users cannot easily write to these files, however. The greatest difference between traditional mainframe systems and newer UNIX**- and DOS-based systems is in their files. This area must receive considerable attention if a planned solution requires regular file exchanges across platforms.
- **Data sharing.** There is frequently a need to share data among processors. For example, an OLTP system often requires periodic, *heavy duty* batch work. This implies a need to share substantial amounts of data. Sharing large amounts of data across different platforms can be complicated for a variety of reasons.
- **Databases.** A database and its associated database manager are seen as a single entity by the rest of the system. A user requests the database manager to read or write elements, based on high-level specifications for the element. SQL is an example of a high-level application interface to a database. The actual data formats and accesses, at the physical disk level, are hidden inside the database manager and are not available to the user. Distributed databases have additional requirements, including use of *two-phase commit* functions and programming. The higher-level environment provided by database managers is a key element of distributed systems. However, mere existence of an SQL interface does not guarantee compatibility of different database managers on different platforms.
- **Personal computing services.** These services include such utility functions as electronic mail, text editors, text formatters, and so forth. These personal computing functions should not be confused with personal computers. Personal computing services, such as electronic-mail and text editing, can be provided in many ways, and the most effective solutions may involve a variety of platforms. Day-to-day company work may depend on services, such as internal electronic mail. Such simple functions may be somewhat overlooked when designing a complex distributed solution, leading to work disruptions later. For example, a conversion that involves moving electronic-mail from fixed function mainframe terminals to new personal computers for every user has an implicit assumption that all users know how to manage their new personal computers. This is not a safe assumption.
- **User and security administration.** It is difficult to mingle different styles of security systems. For example, RACF is an out-of-line reference monitor that works with generic name strings. UNIX** uses in-line code, when opening a file, for example, to test specific protection bits for a single file. Either can provide acceptable security, but planning and administration are different for the two models. Kerberos, one element of a three-party security protocol, is

not a complete security system, and is oriented to authentication rather than authorization.

Current distributed processing technologies have made security administration more complex. Lack of a detailed security plan is a characteristic of a hasty, poorly planned migration. *User administration* includes functions such as adding and deleting users, and controlling user access to files and other resources. Large installations usually have a separate group or department responsible for these functions. Security depends, to a large extent, on separation of duties. The person controlling access to the payroll file, the security administrator, for example, might not be the person who runs the payroll jobs, at least in larger organizations where company officers do not do the hands-on computer operations. This type of security is easy to establish on large, traditional systems, but requires considerable planning in a more distributed environment. Security aspects, in the largest sense of the word, should be considered from an auditor′s viewpoint when designing an important distributed application.

- **User training and skills.** This is often one of the largest investments required for information processing, but it seldom appears as a line item when listing assets. Widely used applications affect many users, in one way or another, and these users, who are usually not computer people, tend to resist change. They know how to use an existing system, to the extent necessary for their job, and are reluctant to accept a new technology unless the benefits are immediately obvious to them. Any change impacts their productivity, and continuous change, such as from chasing the newest, least stable technology, is very disruptive. This factor should be a key management concern, since it is almost always slighted in system conversion plans.
- **Transaction processing.** Transaction processing, largely defined by CICS*, is often the single most important function of a system. Transaction processing has many special requirements and is usually seen as a separate element of data processing.
- **Portable and storage media.** A tape is the most common example of this medium. Long-term storage and interchangeability among systems are factors to be considered. Storage technology for smaller systems is changing very rapidly, and long-term compatibility of media is a concern. Standard corporate functions, such as payroll and general ledger, usually require long-term retention of records. Consideration is required to select a medium type that will still be in use ten years later.
- **Operations services.** An operations group ensures that production jobs are run on schedule. It also ensures that backups are taken and managed properly, hardware is maintained in a timely manner, printer jams are cleared, printers have paper, and so forth. It is desirable to differentiate operators from users or administrators. For security reasons, operators might not have userids; that is, they might not be permitted to log onto the system as users. The daily, routine functions performed by operators can disappear in a poorly planned distributed environment, with potentially serious results.
- **Help desk services.** Any sizable installation will have a recognized source for user assistance. In practice, this service might consist of several separate components. Obvious hardware problems might be reported to one group, simple operational problems, such as “I cannot get the logon

display.," to another group, and so forth. The requirement for help desk functions usually increases in a distributed environment.

- **Software maintenance.** Software maintenance, for both locally written and purchased software, can present a substantial workload. Maintenance programming comes under the category of application development. Software maintenance is the process of receiving, testing, installing, and managing software updates. A well-known example is the handling of IBM PTF tapes. These tasks can be much more complex in a distributed environment, and solutions, at least for mission critical applications, must consider operations with mixed-level software, and must provide an appropriate solution for the software distribution problem.
- **Hardware maintenance.** This category includes such unrelated functions as tape drive cleaning, printer maintenance, cable installation and repair, and debugging of suspected hardware problems. In traditional installations, this area includes planned and preventive maintenance. Workstations and small servers tend to rely on *on-call maintenance* when a component fails. A move from traditional to distributed computing often neglects total system maintenance.
- **Responsible manager.** In traditional computing installations, this is the DP Manager or the Operations Manager. If the payroll job is not run on schedule, for example, the DP Manager is the obvious target for questions and complaints. Executive management expects that important, or mission-critical, applications will have an identified responsible manager. Applications that become extensively distributed may lose the sense of a responsible manager; this exposure must be considered when designing a distributed application.
- **Logs and audit trails.** While these are not important for personal computing, they can be critical for production work. Auditors are unlikely to accept the typical informal use of personal computers for the repository and processing of major financial and company status information. Current distributed technology is poor in this area, requiring considerable attention to audit trails when moving applications to a distributed environment. Logs and audit trails are more a management concern than a technology concern, and tend to be overlooked in some solutions.
- **Systems management, performance management, accounting, capacity planning, problem management**. These are often grouped under the general heading of systems management. Traditional systems have many functions and tools for this work. Newer systems tend to have very few functions and tools in this area. The nature of smaller, distributed systems might reduce the need for some of these functions, but will increase the need for others. A poorly planned shift to distributed processing could produce a system that cannot be managed nearly as well as the previous system.
- **Testing environments.** Systems with a symbolic file system, such as MVS, offer a convenient way to test large applications. Many mixtures of test data and live data can be used, controlled by relatively simple JCL changes. Other platforms might require source code changes or moderately complex, and non-standard, shell script changes to juggle test files. The ability to simulate terminal workloads or replay logs of transactions can be important test tools. Projection and design of routine, acceptable, production-test environments is important, and is sometimes overlooked when converting to new platforms.

- **Disaster recovery**. This is more a matter of planning and controls not directly related to specific operating system and hardware functions. Disaster recovery plans for a distributed environment will probably be quite different than plans for a centralized system. These plans are likely to require actions by many people, in the distributed environment, and might influence some design decisions. Disaster planning should be an early part of distributed system plans.

# Chapter 6. IBM Software Platforms

This chapter provides a description of the term software platform with particular attention to the concept of resource manager (see 3.3.4.1, “Resource Manager Concepts” on page 52). The most common IBM software platforms are described, in alphabetical order, referring to a common structure of functional layers and services. A distinction is also made between local and distributed resource managers, based on the functions and the services they provide.

The description of the IBM software platforms provided here is not intended to replace the existing technical documentation for the individual platforms, to which users are expected to refer for a complete technical description.

## 6.1.1 Software Platform Definition

An operating system basically manages the resources of a computing system, and therefore, determines the functions the system can provide. An operating system can be seen as an organized collection of resource managers and services.

The resource managers are the principal structural elements of an operating system. They maintain the state of a particular resource and provide services and programming interfaces to applications that wish to access and use a resource. If a resource manager operates within a single system, such as storage management, it is a local resource manager. If it operates across multiple systems, it is a distributed resource manager. Distributed resource managers include parts that provide the interface that requesters use, called the client parts, and parts that perform functions on resources, called the server parts. The client part provides interfaces to allow an application to manipulate a resource. It also hides the location of the resource, as well as the actual protocol used to access it. Multiple protocols are supported to deal with a heterogeneous environment.

An operating system with its set of local and distributed resource managers constitutes a software platform.

The early operating systems were mostly local computing systems because their resource managers operated within the boundaries of a single system. If all the resource managers are local, the system is only aware of its own resources and is a stand-alone applications system. All the program instructions required by an application are executed in a single computer.

Over time, operating system technologies have evolved due to the availability of multiple system architectures, different price/performance ratios, and, above all, by the evolution in computer communications that allow computers to be geographically dispersed but still interconnected. The ability to communicate has not only allowed electronic data transfer and remote operations, but has also prompted the design of distributed resource managers. Using application enabling services, an application can access and use resources located on different software platform and, furthermore, the application itself may be distributed across software platforms.

The interconnection of multiple software platforms that have distributed resource managers results in a distributed computing system or, simply, a distributed system.

The aggregation of heterogeneous software platforms (see section 3.2, “Historical Heterogenous System Structures” on page 42) results in heterogeneous distributed systems.

Today’s heterogeneous distributed systems are required or expected to be open as described in section 2.9, “Open Systems” on page 36 and section 2.3.1, “Programming Interfaces” on page 8.

Figure 53 shows the elements of a software platform. The resource managers are shown as the interface they provide through local operating system services and application enabling services. In this chapter, we will use this figure to address the requirements of heterogeneous distributed systems, and to describe how the current IBM software platforms (AIX/6000*, AIX/ESA*, IBM DOS, MVS/ESA*, OS/2*, OS/400*, VM/ESA*, and VSE/ESA*) qualify as members of a distributed system.

The distributed system structure is intended to provide a reference for discussion about distributed systems and does not intend to establish absolute prerequisites or a mandatory list. Therefore, the functions, facilities, and services do not necessarily have to be present in every software platform.

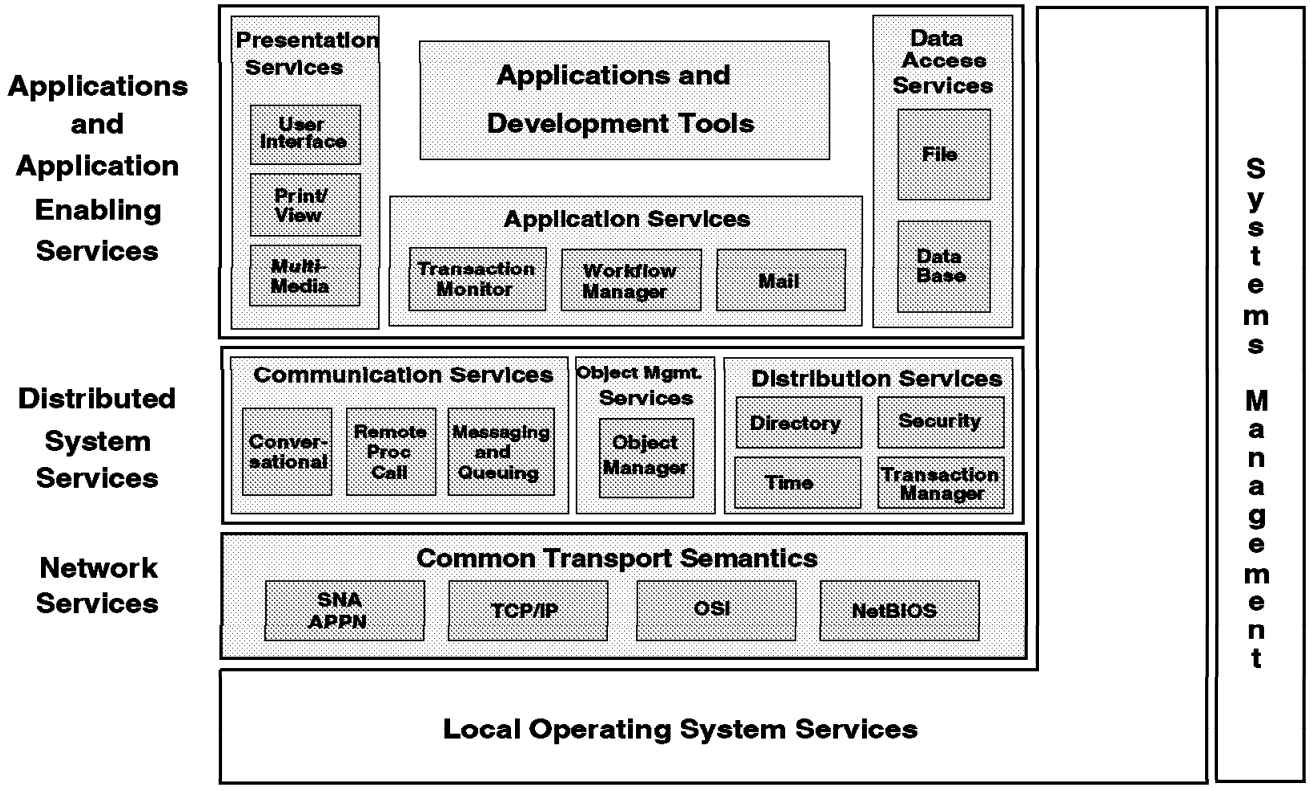


*Figure 53. Elements of a Software Platform*

Each platform is described by the following sequence of functional layers:

- Local operating system services, such as:
  - Virtual storage, multiprocessor and multiprocessing, I/O and other hardware resource managers
  - Workload schedulers, including interactive, online transactions, and batch, and workload management
  - Spooling
  - Local data access services and storage services for inactive data
  - Security services.
- Network services
- Distributed system services
  - Communication services
  - Object services
  - Distribution services.
- Application enabling services
  - Presentation services consisting of user interface, printing/viewing and multimedia services.
  - Application services consisting of transaction monitors with their application interfaces, mail services and workflow manager.
  - Distributed data access services including storage services for inactive data.

  In the application enabling services layer, there are platform-dependent services (for example, the Network Operating System (NOS) in the OS/2 platform described in section 6.6.4.4, “Network Operating System” on page 196), and therefore, other services are possible for future technology evolution.
- Application development tools.
- Selected system management services, including:
  - Performance management and accounting
  - Operations and automation
  - Availability and integrity.

Finally, for each platform, there is a table of selected system services and APIs, in common use in distributed systems, indicating the availability status for the platform. (Appendix A, “APIs, Protocols, and Facilities Description” on page 227 provides a short description of each API and service).

The products and components mentioned as service providers are only a selection of the total offering available for each platform. The intention is to select significant (rather than complete) functions, and to show how the various products and components can be mapped against a common structure of functional layers. Some non-IBM products are included when appropriate.

The other books of the Library for Systems Solutions provide a more complete and detailed description of the solutions available on each software platform for various system configurations.

## 6.2 AIX Version 3

Advanced Interactive Executive Version 3, or AIX* V3 (5756-030), is the native IBM UNIX** operating system for the RISC System/6000* and is, therefore, also known as AIX/6000*. The name AIX/6000* will be used in the rest of this section.

AIX/6000* is designed to be an operating system for an open distributed environment and integrates a wide range of industry standard technologies.

AIX/6000* can be used in a variety of environments, including numeric intensive scientific applications, graphics intensive engineering applications, and commercial applications. It can be used as a stand-alone environment with one or more users, it can be a server, or it can be combined with other systems in a network for distributed applications.

Figure 54 shows the elements of the AIX/6000* software platform.

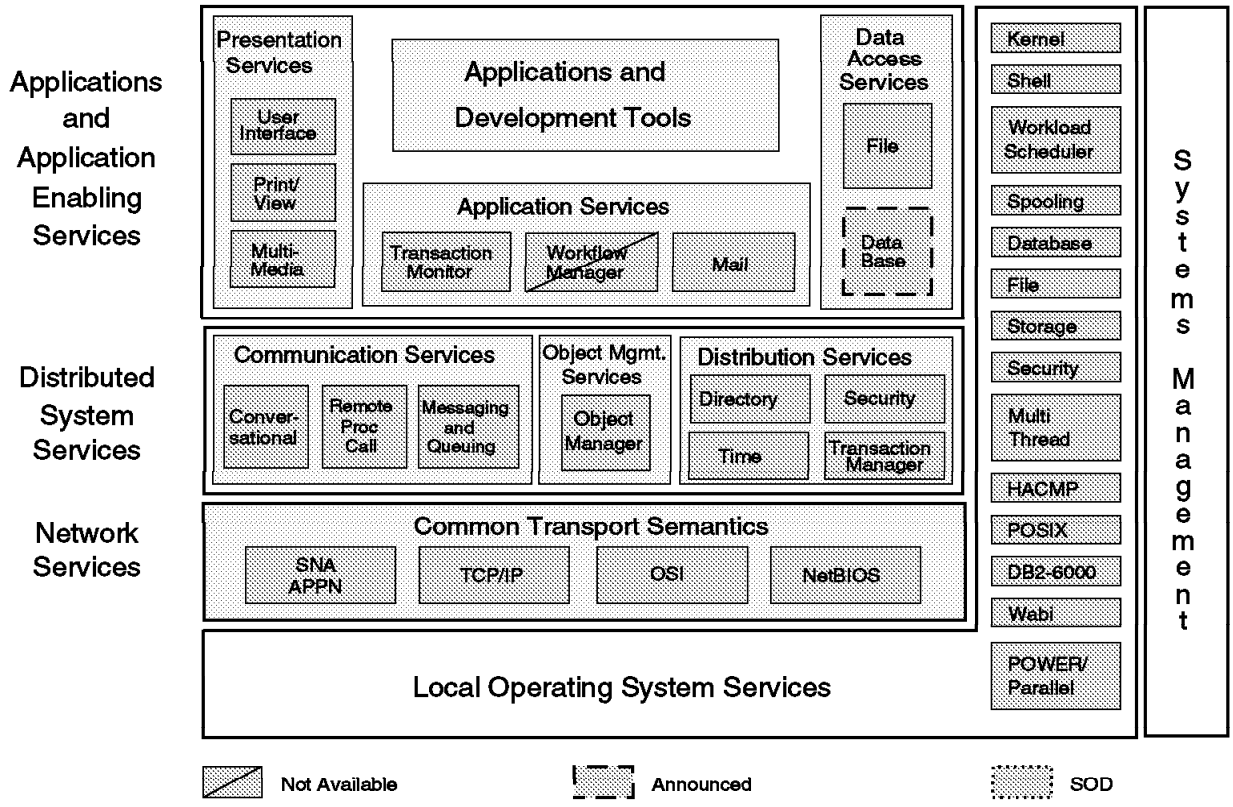


*Figure 54. Elements of the AIX/6000* Software Platform*

### 6.2.1 Local Operating System Services

The basic structure of a UNIX** operating system can be represented by two layers (see Figure 8 on page 20):

- The kernel that provides all the base services for the applications.
- A command processing component, called a shell, that provides the user interface, performs parameter substitution, and calls appropriate command programs. The shell also provides utilities for editing, program development, and text processing.

#### 6.2.1.1 Kernel

The system architecture and characteristics of AIX/6000* as follows:

- AIX/6000* is derived from UNIX** System V and is POSIX** IEEE standard 1003.1-1990 conformant and Berkeley Software Distribution 4.3 (BSD 4.3) compatible. It also meets the specifications of XPG3**. It incorporates X-windows** from MIT and Motif** from OSF**. Distributed systems support is provided by the OSF/DCE**.
- The low-level kernel contains the base services for the control of the RISC/6000* POWER* architecture, the virtual storage and the I/O subsystem. The low-level kernel shields the kernel from the details of the hardware architecture. Additional services include multiple preemptable processes (a process is one unit of work in the UNIX** environment), process creation, multi-preemptable interrupt levels, time control, and multi-thread.

  IBM has stated the intention to include OSF/1** functionality and to conform to the Common API Specification for UNIX**
- The virtual storage available with AIX/6000* is more than 1000 TB (2••) and is managed as 16 million segments of 256 MB each. The kernel occupies one (256 MB) segment, and each process has three segments allocated: one for program code, one for computational data, and one for the program′s nesting information save area or stack. Additional segments can be obtained by the processes for use with private or shared data, shared code, or mapped files. The RISC/6000* storage access instructions generate an address of 2•• (4GB); four bits select a segment, providing access to 16 segments, and 28 bits give the offset within the segment. The AIX* process space is a 2•• address space (4GB), and each process is able to address only a portion of the system-wide virtual storage space. If desired, segment registers can be changed, allowing a process to access many more than 16 segments.

  Figure 55 on page 141 shows the AIX/6000* single-level virtual storage and the virtual storage used by an application.
- The real storage that AIX/6000* can use is up to 4GB (2••).
- The file system provided by the kernel is a hierarchical tree-structured directory that provides access to unstructured files. It has the following extensions over the traditional UNIX** implementation:
  - The logical volume (LV) function allows user file systems to span across multiple storage devices (file-systems in traditional UNIX** are normally limited to the size of a single, physical storage device). Logical Volume services also allow for the addition and removal of physical disk units. Logical Volume also offers data mirroring without using hardware mirroring features to improve data availability.

- Mapped files is an extension of the file system that uses the single-level-store environment. A mapped file is accessed through the virtual storage mechanism simply by loading data from the appropriate address. A virtual storage segment can contain only one file. Multiple files can be mapped by a single process.
- The kernel file system services are extended to provide the necessary facilities to manage databases in virtual storage. With the mapped file facility, a database can be shared and accessed concurrently by many processes executing different transactions. The services provided include the ability to perform space management, file and record level locking, and buffer cache synchronization.

• AIX/6000* is designed to optimize the performance of the RISC/6000*. It takes advantage of features such as superscalar architecture, floating-point implementation, high resolution graphics, Micro-Channel Architecture, and high speed optical serial links for high I/O bandwidth.

  Eight I/O adapter slots for external device attachment are available on RISC/6000* and are connected to the main storage through one I/O bus.

AIXwindows* Environment/6000 (5601-257) provides Wabi (Windows application binary interface) as a separately ordered and chargeable feature and this allows support for 13 of the most popular Microsoft** Windows** applications on RISC/6000*.

High Availability Cluster Multi-Processing/6000 (HACMP, 5765-111) provides for coordination and operation of multiple AIX/6000* and associated applications (AIX/6000* clusters) in an open system client/server environment. The clustered systems are loosely coupled with software services. The HACMP/6000 software architecture provides reliable, recoverable shared disk resources for database or online transaction processing servers.

AIX* Parallel System Support Program (PSSP, 5765-296) is software to support the scalable Scalable POWERparallel System 9076 SP2.

The 9076 SP2 is the evolution of 9076 SP1 and can have up to 128 nodes. Each node runs a copy of AIX/6000* and PSSP. The nodes are connected through a High-Performance Switch (HPS) that allows any-to-any communication.

The 9076 SP2 provides a variety of device attachments across multiple interfaces which includes:

- Fiber Distributed Data Interface (FDDI)
- Ethernet**
- Token-Ring
- Small Computer System Interface (SCSI) adapter to attach disks, tapes, optical devices and storage subsystems
- High-Performance Parallel Interface (HPPI) for high speed data server and high speed connectivity to super computers
- ESCON and Block Multiplexor Channel Adapter (BMCA) for high speed interoperability with IBM S/390 systems.

The Scalable POWERparallel System enables high performance parallel processing for computational intensive and UNIX** business critical data query and transactional applications.

PSSP includes System Monitor, Resource Manager, System Data Repository (SDR), System Management (see section 6.2.6, “System Management” on page 149), Virtual Shared Disk (VSD) and Communication Subsystem Support (CSS).

The System Monitor enables the system administrator to monitor and control the 9672 Scalable POWERparallel System as a single system.

The Resource Manager allows space and time sharing of nodes by parallel programs.

The System Data Repository is a distributed data repository that provides distributed access to system data on the control workstation and nodes of the SP system.

The Virtual Shared Disk allows global shared disks to an application, even though the disks are not physically attached to all nodes.

The Communication Subsystem Support allows message passing through the HPS among 9076 SP2 nodes.

IBM Transaction Monitors and other Licensed Program Products for AIX/6000* are supported for the Scalable POWERparallel System in serial mode. IBM DB2 AIX/6000 will be supported for the Scalable POWERparallel System in parallel mode.

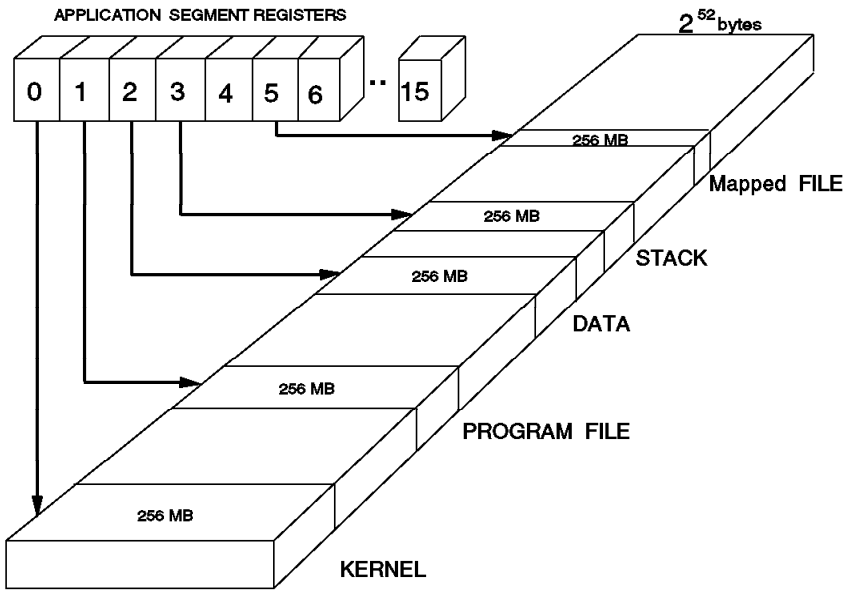


*Figure 55. AIX/6000* Virtual Storage*

#### 6.2.1.2 Shell

The command shell is the basic interface to a UNIX** system, interacting with the user through the keyboard and the terminal display. The shell is actually a procedures language, a command interpreter, and provides several commands and utilities. The AIX/6000* provides multiple command shells as follows:

- C Shell
- Bourne Shell**
- Korn Shell**

#### 6.2.1.3 Workload Scheduler

AIX/6000* provides an execution environment for interactive and real-time-processing applications.

The following features are available to schedule and control the workload:

- Priority control and scheduling of multiple processes.
- Preemptive priority dispatching of ready processes.
- Preemptable kernel.
- Facilities for direct control of virtual storage, enabling a program to lock itself and its data into real storage. Using these facilities allows the program to bypass the demand paging algorithms utilized by the kernel to manage the single-level virtual storage.

#### 6.2.1.4 Spooling

AIX/6000* provides a Simultaneous Peripheral Operations On-Line (Spool) facility for temporary storage of user related data, mainly output data for later printing.

#### 6.2.1.5 Database

IBM DB2 AIX/6000* (5765-172) is the member of the IBM relational database family for the AIX/6000* platform. It can be used in a single-user workstation. The database manager provides SQL capabilities and roll-forward recovery for database recovery from a system failure.

#### 6.2.1.6 File

AIX/6000* provides the standard UNIX** file system that allows a hierarchical access to unstructured byte-string files. Facilities allow mapping of an entire file system in virtual storage and manipulating it as shared storage.

Indexed files are not standard on UNIX** systems. Two products provide this function with AIX/6000*: AIX* VS COBOL Compiler/6000 (5601-258) for ISAM organization, and Encina** Structured File Service (5696-237) for indexed files.

#### 6.2.1.7 Storage Services

Unitree** for AIX/6000 (5696-398) provides continuous, multi-level, transparent file backup and restore, as well as file archival management. It is controlled by customer specified parameters.

#### 6.2.1.8 Security

AIX/6000* provides security features to meet the C2 level as defined by the U.S. Department of Defense (DOD). C2 level means that users must have a password to login and to control access permission for their files, and the system administrator may trace the individual user's activities.

AIX/6000* provides protection against unwanted program alterations (for example, computer viruses) by maintaining control information about programs defined by a specialized algorithm (program checksum). When a program is invoked, its current checksum is compared to a previous valid checksum.

To prevent the accidental disclosure of information, AIX/6000* provides features to clear the content of a real storage page and a disk block space after they have been released by the program.

### 6.2.2 Network Services

AIX/6000* supports a wide range of physical connectivity options using the three main network protocols: SNA, TCP/IP, and OSI.

AIX/6000* running on a RISC/6000* system can communicate as a workstation, as a peer system, or as a host system. The following connection types are supported:

- Ethernet** V 2
- IEEE 802.3
- Token-Ring
- X.25
- SDLC
- S/390* ESCON* or parallel channel - (host requires TCP/IP)
- Fiber Distributed Data Interface (FDDI)
- Serial optical channel
- Asynchronous
- Coaxial connection.

TCP/IP provides the following main services:

- File Transfer Protocol (FTP) server facility for file exchange. (see section 6.2.4.3, “Data Access Services” on page 147).
- Terminal passthrough (TELNET) to allow a user client to remotely logon to another computer with the TELNET server function.
- Simple Mail Transfer Program (SMTP) to allow an end-to-end electronic mail exchange.
- The agent function of Simple Network Management Protocol (SNMP) for relaying information to an SNMP manager.
- Remote procedure call (RPC) for distributed applications.
- The socket interface from Berkeley** Software Distribution (BSD) and the UNIX** sockets, both for writing network applications.

Network Computing System (NCS**) is a standard component of AIX/6000* and allows programmer to distribute processing power to other systems.

IBM LoadLeveler* (5765-145) is a distributed, network based, job scheduling program for RISC/6000* and Scalable POWERparallel System based on NQS** protocols.

AIX SNA Server/6000 and AIX SNA Gateway/6000 (5765-247, 261) provide application programming interfaces for SNA LU 0, 1, 2, 3, and 6.2 protocols and allow user applications to communicate with:

- Traditional 3270 terminals
- Remote job entry stations
- Peer applications.

AIX SNA Server/6000 implements the SNA Advanced Peer-to-Peer Networking (APPN) technology.

AIX* OSI Services/6000 (5696-385) provides:

- File Transfer Access and Management (FTAM) (see section 6.2.4.3, “Data Access Services” on page 147).
- Message handling, according to the X.400 standard.
- Virtual Terminal (VT).
- ASN.1 tool kit to develop OSI applications.
- OSI applications can operate over TCP/IP networks.
- OSI/6000 provides gateway to SMTP and FTP services of TCP/IP and Virtual Terminal to TELNET TCP/IP, thus providing access to applications residing on OSI networks.

AIX* LAN Distributed Platform/6000 (LANDP/6000 5765-076) provides client/server distributed capability for applications in a LAN environment connected to multiple operating systems, such as IBM DOS, OS/2*, AIX/6000*, and OS/400*. LANDP/6000 provides the user with a consistent programming interface to request services from the four software platforms and provides:

- SNA communication server facility for LU 0, 1, and 2
- Program-to-program communication with SNA LU 6.2 server
- LAN router supporting NetBIOS on Token-Ring and TCP/IP on token-ring and Ethernet**
- Store and forward server
- Database server to store, retrieve, and update data within the LAN
- Distributed Computing Environment (DCE**) support.

Multi-Protocol Transport Networking architecture (MPTN) supports CTS (Common Transport Semantics) in IBM′s Networking Blueprint. CTS provides a common view of networking protocols and makes applications independent of the underlying network transport. IBM intends to provide the following AnyNet functions for AIX/6000*.

- AnyNet APPC over TCP/IP
- AnyNet Sockets over SNA.

3270 Host Connection Program/6000 (5601-260) allows AIX/6000* users and applications to interact with an IBM S/390 through a 3270 display or printer emulation session.

IBM Connection Program/400 (5798-RZB) allows AIX/6000* users and applications to communicate with AS/400*. It allows users to access applications and data residing on AS/400*.

Table 1 summarizes the AIX/6000* networking capabilities.

*Table 1. AIX/6000* Networking Capabilities*

| | Ethernet** | TokenRing | FDDI | X.25 | SDLC |
|---|---|---|---|---|---|
| **SNA** | YES | YES | YES | YES | YES |
| **TCP/IP** | YES | YES | YES | YES | YES |
| **OSI** | YES | YES | YES | YES | - |
| **NetBIOS** | YES(1) | YES(1) | - | - | - |

**Note:** (1) Supported by LANDP/6000

## 6.2.3 Distributed System Services

### 6.2.3.1 Communication Services

The communication services capabilities within AIX/6000* are provided for:

- Conversational

  CPI-C API for Application Program to Program Communication (APPC) using LU 6.2 protocol provided by AIX SNA Server/6000.

- Remote Procedure Call

  AIX* Distributed Computing Environment (DCE**) (5765-117,118,119,121) provides RPC that allows programs to work across heterogeneous systems by masking the differences between data representations and the network details of different software platforms.

- Messaging and Queuing

  The product ezBRIDGE** Transact** (5787-ECY) provides Message Queue Interface (MQI) to distributed applications running on AIX/6000* platform.

### 6.2.3.2 Object Management Services

IBM SOMobjects Developer Toolkit and the associated runtime products Workstation Runtimes and Workgroup Runtimes (5604-480, 483, 485), provide System Object Module (SOM) technology and Distributed SOM (DSOM) technology on the AIX/6000* platform. SOMobjects comply with the Object Management Group's (OMG**) Common Object Request Broker Architecture (CORBA).

### 6.2.3.3 Distribution Services

- Directory

  Global directory services are provided by AIX* DCE Global Directory Server/6000 (5765-120) and AIX* DCE Global Directory Client/6000 (5765-259).

  Cell directory services are provided by AIX* DCE Cell Directory Server/6000 (5765-119) and AIX* DCE Base Services/6000 (5765-117).

- Security

  Security services are provided by AIX* DCE Security Server/6000 (5765-118) and AIX* DCE Base Services/6000 (5765-117). The DCE** security service

component is integrated within the main distributed services and data-sharing components. It provides the network with three conventional services: authentication, authorization, and user account management. DCE**'s distributed security service incorporates an authentication service based on the Kerberos** system from MIT.

- Time

  Time services are provided by AIX* DCE Base Services/6000 (5765-117).

- Transaction manager

  Encina** Server (5696-240) for AIX/6000*, employing the 2-phase commit protocol to insure transactions and data integrity across distributed systems, with features such as locking, logging, and recovery.

### 6.2.4 Application Enabling Services

The application enabling services available with AIX/6000* are described in this section.

#### 6.2.4.1 Presentation Services

- User Interface

  AIX/6000* provides character mode terminals or graphic displays. AIX/6000* incorporates the X-windows** system from MIT and the Motif** interface from OSF**. X-windows** provides a network transparent window system that allows an application program to access a local display terminal independent of where the application program is actually executing.

  X-windows** is based on a client/server model where a client application communicates with the server, which controls the shared resources of display, keyboard, and mouse. On a single display, a user can have multiple windows to execute multiple applications.

- Print/View

  AIX/6000* provides standard TCP/IP print services. The line printer client/line printer daemon (LPR/LPD) function of TCP/IP provides client/server support for remote printing. The LPR sends data to be printed to the LPD on a specified server workstation, which manages the specified printer. The LPD server on the workstation provides access to the attached printers.

  NetWare** for AIX/6000* (5696-236) allows RISC/6000* workstations to act as servers for NetWare** LAN. It brings the AIX/6000* resources and applications to PC LAN users. It also provides print sharing for NetWare** clients and AIX/6000* users.

  Print Services Facility/6000 (PSF/6000 5765-140) delivers the IBM Advanced Function Printing* (AFP*) capabilities to the RISC/6000* platform running the IBM AIX/6000* operating system. PSF/6000 provides printing solutions for stand-alone environments, local area network (LAN) environments, distributed print environments (via TCP/IP and Network File Server** (NFS**) protocols), and printer sharing between LAN systems and host systems.

- Multimedia

  AIX Ultimedia Services/6000 (5696-709) provides API, graphical interface and tools to support audio and video on RISC/6000*.

### 6.2.4.2 Application Services

- Transaction monitor

  AIX* CICS/6000* (5765-148), a member of the CICS* family, and AIX* Client for CICS/6000* (5765-152) provide compatibility with the other CICS* family members through the CICS* API, while interoperability is supported by the Inter System Communication (ISC) facility.

  This facility, which exists across all members of the CICS* family of products (AIX/6000*, MVS*, OS/2*, OS/400*, VSE), allows CICS* to route transactions to another CICS* for execution, allows transparent access to remote CICS* resources, and allows the invocation of remote CICS* applications.

  Encina** Monitor (5696-239) for AIX/6000* is a transaction monitor that uses OSF/DCE** for distributed processing and ensures transaction integrity. Encina** Server for AIX/6000* provides the capability to access resource managers that support X/Open interfaces (TX,XA). Encina** Structured File Services (see section 6.2.4.3, “Data Access Services”) provides transactions with data access services.

  UNIX System Laboratories has expressed the intention to port and support on the Scalable POWERparallel System 9076 SP2 its transaction monitor TUXEDO**.

- Mail

  Electronic mail support in AIX/6000* is supported through the implementation of the 4.3 Berkeley Software Distribution (BSD**) Reno level of TCP/IP (providing mail routing facilities), and the message handling (MH) application, providing additional facilities for mail applications.

### 6.2.4.3 Data Access Services

- Relational

  IBM DB2 AIX/6000 (5765-172) is the member of the IBM relational database family DB2 for the AIX/6000* software platform. It can be used in a single-user workstation and in a client/server environment. The database manager provides SQL capabilities and roll-forward recovery for a database recovery from a system failure. IBM DB2 AIX/6000 will be supported on the Scalable POWERparallel System 9076 SP2 in parallel mode.

  Distributed Database Connection Services for AIX/6000* (DDCS/6000 5765-197), announced with DB2 AIX/6000, provides database access for decision support systems as well as for online transaction processing applications that may reside on a LAN or in a host.

  DDCS/6000 can participate in environments where Distributed Relational Database Architecture (DRDA) is implemented at the level of the Remote Unit Of Work (RUOW).

  Additional services with client/server capabilities have been announced with the AIX* Client Support/6000 (5765-205), and the DB2 Client Application Enabler/6000 (5765-218). These services give AIX/6000* workstation users the capability to access and update remote relational data bases residing in software environments where the DRDA RUOW is implemented.

  Distributed database technology is also available from vendor products such as Ingres**, Sybase**, Oracle** and INFORMIX**. The vendor of Oracle** has expressed the intention to port and support the product on the Scalable POWERparallel System 9076 SP2.

- File

  The following products provide services for distributed file access:

  - Encina** Structured File Services (5696-239) provides a fast, recoverable, record-oriented data storage mechanism. The file organizations are binary-tree clustered files, relative-record files, and entry-sequenced files.
  - Network File System (NFS**), a standard component of AIX/6000*, allows users to have transparent access to file systems distributed over the network.
  - OSI file services allow the exchange and remote management of files between platforms that implement the FTAM protocols.
  - FTP (File Transfer Program), as a standard component of TCP/IP, allows the exchange of files using TCP/IP.
  - NetWare** for AIX/6000* (5696-236) allows RISC/6000* workstations to act as servers for NetWare** LANs. It makes the AIX/6000* resources and applications available to PC LAN users. It also provides file sharing for NetWare** clients and AIX/6000* users.

- Storage Server

  - ADSTAR Distributed Storage Manager (ADSM* 5648-020) allows an MVS or VM system to act as a file backup and archive server for LAN file servers and workstations. AIX/6000* can be a client of ADSM MVS or VM.
  - ADSTAR Distributed Storage Manager/6000 (ADSM/6000, 5765-203) has the same storage management functions as ADSM for MVS and VM on a RISC System/6000*. It allows a RISC/6000* to act as a file backup and archive server for LAN and workstations. This product operates as a server for clients running OS/2*, NetWare**, Windows**, DOS, Macintosh**, SunOS**, DEC** ULTRIX**, HP-UX** and SCO** Open Desktop** software platforms.

    IBM intends to provide ADSM/6000 for backup, recovery and archiving of data on POWERparallel systems.
  - AIX File Storage Facility/6000 (FSF/6000, 5696-708) provides a RISC/6000* client with automatic client disk space management and file migration to any standard NFS** server. It migrates the least used client files to the server disks on policy-based storage management.

### 6.2.5 Application Development

AIX/6000* provides the standard development environment for UNIX** systems. Products that extend this facility are available, among them:

- AIX* Software Development Environment Workbench/6000 (5696-524) to provide the basis for an integrated software development environment.
- AIXwindows Interface Composer/6000 (5756-027) to provide facilities for the design of graphical user interfaces for applications using OSF/Motif**.

Computer Aided Software Engineering (CASE) packages for the AIX/6000* platform are provided by several software vendors.

AIX* Parallel Environment (5765-144) provides support for parallel application development, debug, analysis, execute and tune on POWERparallel systems.

### 6.2.6 System Management

SystemView* is the IBM systems management strategy for managing, planning, coordinating, and operating open, heterogeneous distributed systems. SystemView* includes a SystemView* structure and the SystemView* conforming products. The SystemView* structure is designed to provide a consistent end-user interface, shared data, enhanced automation, and increased integration among systems management products. The structure embraces selected open standards and architectures and defines conformance criteria for the SystemView* products. The SystemView* conforming products are, and will be, developed by IBM and non-IBM vendors.

The System Management Interface Tool (SMIT) facility is a set of menu-driven services to perform local tasks such as installation, configuration, device management, problem determination, and storage management. SMIT is available in both character and graphical interfaces.

AIX* PSSP has a collection of tools to help manage the Scalable POWERparallel System, the software, the users from a central point of control, and to automate configuration and management tasks. A SMIT based Configuration Management Interface (CMI) is provided for management activities.

#### 6.2.6.1 Performance Management and Accounting

A set of tools that allow the tuning of AIX* and application performance is provided with AIX/6000*. Best/1** for AIX/6000* is a local capacity planning tool that assists in making reliable capacity planning and performance management decisions.

#### 6.2.6.2 Operations

AIX* NetView*/6000 (5765-077) provides services for distributed network management. It manages TCP/IP devices, which include SNMP agents. The SNMP management includes fault, performance, configuration, and alert management. It uses a graphical, object-oriented, Motif** based user interface that allows the network topology to be viewed on top of meaningful maps, buildings, or devices.

AIX* PSSP allows the administrator and operator to monitor and manage the Scalable POWERparallel System for a control workstation as a single point of control.

#### 6.2.6.3 Availability and Integrity

AIX/6000* is an implementation of UNIX**, with specific features designed to improve total system availability. Some features among others are:

- The ability to dynamically add device drivers without requiring a kernel rebuild, or a re-IPL
- Specific facilities for file management, such as:
  - Logical volume manager to provide disk mirroring for specified volumes. Once mirroring is activated, management of the data and copies is automatic and transparent to the application and user.
  - Journaled file system for automatic logging of changes to the file systems structure. If the system crashes, the log is used to rebuild the file systems automatically on restart.

– HACMP/6000 to improve application availability in a AIX/6000* clustered configuration.

### 6.2.7 Selected APIs, Protocols, and Facilities

Table 2 summarizes selected protocols, facilities, and APIs available with AIX/6000* and mentioned in this section.

*Table 2. AIX/6000* Selected APIs, Protocols, and Facilities*

| | **AIX/6000*** |
|---|---|
| **Berkeley**Sockets** | YES |
| **POSIX**** | YES |
| **CPI-C** | YES |
| **APPC** | YES |
| **APPN** | YES |
| **NCS**** | YES |
| **NQS**** | YES |
| **OSF/DCE**** | YES |
| **SQL** | YES |
| **DRDA-RUOW** | ANN |
| **NFS**** | YES |
| **FTP** | YES |
| **FTAM** | YES |
| **OSF Motif**** | YES |
| **X-windows**** | YES |
| **NetWare** server** | YES |
| **LAN & workstation server** | YES |
| **SMTP** | YES |
| **NetView** | YES |
| **SNMP** | YES |

**Note:** ANN (announced)

## 6.3 AIX/ESA*

AIX/ESA* (5756-112) is the IBM S/390 member of the AIX family. It provides a UNIX** based software platform that operates natively on the ES/9000* processor family. AIX/ESA* is built upon the Open Software Foundation OSF/1** platform and provides additional IBM enhancements to exploit large processor capabilities. AIX/ESA* is designed to address the needs of UNIX** users that require the high vector and scalar performance, the large capacity, and the high data bandwidth of the ES/9000* processors and the ESA/390* architecture. AIX/ESA* is the follow-on to AIX/370 with a high degree of binary and data compatibility.

Figure 56 shows the elements of the AIX/ESA* software platform

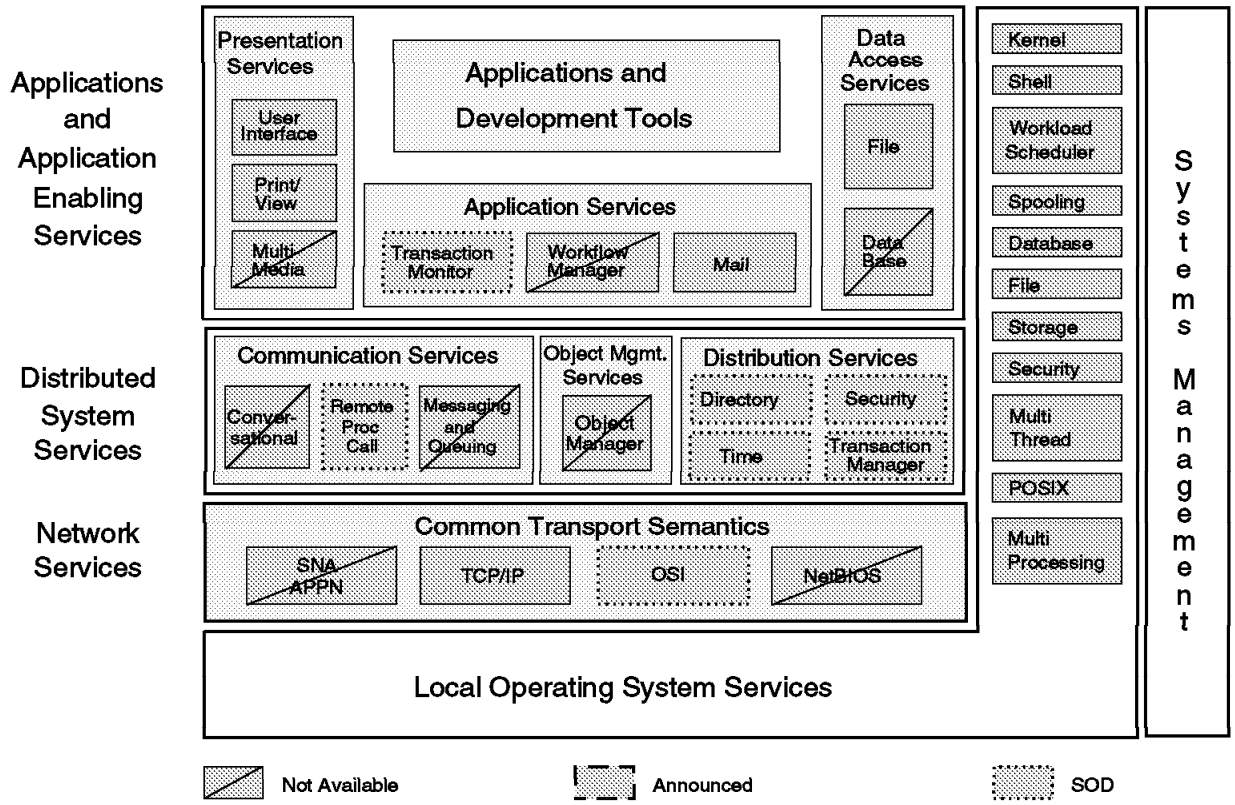


*Figure 56. Elements of the AIX/ESA* Software Platform*

### 6.3.1 Local Operating System Services

The basic structure of a UNIX** operating system can be represented by two layers: (see Figure 8 on page 20).

- The kernel that provides the base services for the applications and that can be divided into two parts: the low-level kernel that interfaces with the hardware, and the kernel that interfaces with the applications.

- A command processing component, called the shell, that interfaces with users, performs parameter substitution, and calls appropriate command programs. The shell also includes a set of utilities for editing, program development, and text processing.

#### 6.3.1.1 Kernel

The system architecture and characteristics of AIX/ESA* are as follows:

- AIX/ESA* is built on the Open Software Foundation OSF/1** operating system.
- The virtual storage space available to each process (the address space) is 2GB in size and provides isolation of one process code and data from the other processes in the system. For each fork or exec command that creates a new process, a new address space is created.
- The low-level kernel is based on the micro-kernel Mach** from Carnegie Mellon University (CMU), and implements four basic concepts: threads, tasks, ports, and messages. The thread is an execution unit, and the task is a protection unit. They implement the traditional process concept of the UNIX** system. Ports and messages are used to communicate between tasks within the operating system. Every service and every resource in the system is represented by a port. Tasks and threads request a service by sending a message to a particular port. All the services are handled by user-state tasks (also called servers) in the kernel. Additional services can be implemented by a user-state task and represented by a collection of ports.
- The kernel provides services for scheduling, recovery, storage management, accounting, inter-process communication, multiprocessor, and multi-thread support. The OSF/1** kernel includes the parallelism extension for multiprocessors from Encore**, the security functions from Sicureware**, and the portable Streams environment from Mentan**. The AT&T** streams are also supported. AIX/ESA* has improved the OSF/1** kernel functions, to scale up to the ES/9000* processor.
- The AIX/ESA* device drivers are from IBM, and provide support for ES/9000* attachable devices. With the dynamic kernel extensions capability of AIX/ESA*, device drivers can be added, modified, and deleted, even while the system is running. Physical DASD sharing among AIX/ESA* platforms is not supported.
- The logical volume manager (LVM) has been ported from AIX/6000* and allows the file systems to span multiple storage devices. While file systems in traditional UNIX** are limited to the size of a single physical storage device, the LVM allows file systems of up to 2GBs. It also offers data mirroring without using hardware mirroring features.
- The program loader of AIX/ESA* supports shared libraries, which means that applications do not have to maintain private subroutine libraries. The loader is designed to handle multiple object module formats, which allows use of programs from different sources, such as BSD** and AIX/370. This gives AIX/ESA* upward compatibility with AIX/370.

- AIX/ESA* conforms to the standard of POSIX** 1003 and meets the specifications of XPG4**.
- The processor complex used by AIX/ESA* can have up to 2GBs of central storage and up to 16TBs of expanded storage.
- The vector facility feature can be used by AIX/ESA* processes for computing intensive applications.
- AIX/ESA* supports the ESCON* architecture for I/O connectivity available on the ES/9000* processors (see the channel subsystem in section 6.5.1, “Local Operating System Services” on page 168) and High Performance Parallel Interface (HIPPI) connections.

Figure 57 shows the AIX/ESA* virtual storage environment.

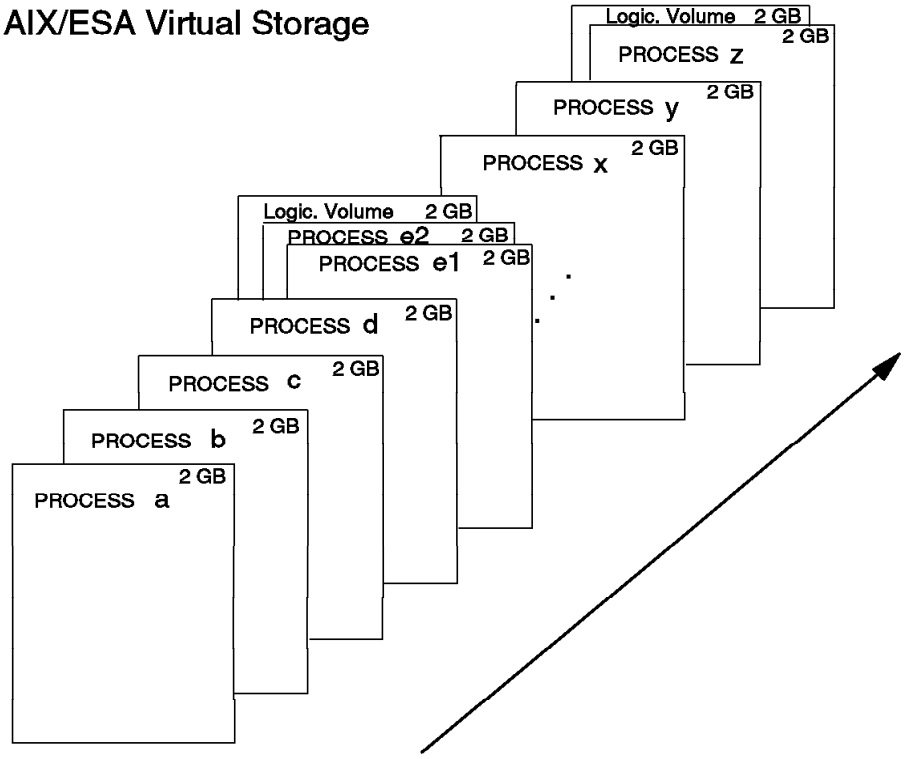


*Figure 57. AIX/ESA* Virtual Storage*

### 6.3.1.2 Shell

The command shell is the basic interface to a UNIX** system, interacting with the user through the keyboard and the terminal display. The shell is actually a procedures language and a command interpreter, and provides several commands and utilities. The AIX/ESA* Shell includes commands and libraries from AIX/6000*, Berkeley′s BSD** 4.3, the Bourne Shell**, the C Shell, and the Korn Shell**.

#### 6.3.1.3 Workload Scheduler

AIX/ESA* provides workload scheduling for interactive applications and for intensive computing applications. Sterling Software′s NQS**, which has been ported to AIX/ESA*, allows submission of batch jobs from UNIX** systems using the Network Query System (NQS**) protocol.

IBM intends to modify LoadLeveler*(5765-145) for AIX/ESA*.

Network Computing System (NCS**) for AIX/ESA* (5765-060), derived from HP**/Apollo NCS**, provides a set of object oriented tools for heterogeneous distributed computing allowing tasks to be distributed across computers in a network. It allows workstation applications that have occasional numeric intensive computing (NIC) requirements to schedule the NIC work on the AIX/ESA* system and to receive the results back at the workstation. The RPC interface is used by NCS**.

In addition to the traditional UNIX** time-slice or fair-share support, AIX/ESA* provides an enhanced scheduling algorithm to provide satisfactory service to a large number of concurrently active users. This includes scheduling policy, priority manipulation, resource limit, and resource usage.

A specialized scheduling algorithm has been implemented for intensive computing applications. The workload scheduler automatically moves the application to a vector processor whenever the need for vector operations is detected.

#### 6.3.1.4 Spooling

The Simultaneous Peripheral Operations On-Line (Spool) facility is provided by AIX/ESA* and is the temporary storage for user-related data, mainly output data for later printing.

#### 6.3.1.5 Database

At present, no IBM database manager has been ported to AIX/ESA*. Database technology is available from vendors, such as Ingres**, Sybase**, and INFORMIX**.

IBM has stated its intention to provide a database manager for AIX/ESA* based on the SQL* interface.

#### 6.3.1.6 File

AIX/ESA* provides a virtual file system (VFS) concept that allows a seamless integration of different file systems for the end user. AIX/ESA* supports the Berkeley** fast file system, the AIX/370 file system, and the NFS** file system. Under control of VFS, the differences among those file systems are transparent to the end user. In addition, all three file systems have been fully parallelized to give improved performance on multiprocessor systems. These file systems are organized with a hierarchical tree-structured directory, and provide access to unstructured byte-string files.

#### 6.3.1.7 Storage Services

The Unitree** product provides a mechanism for automatically archiving and retrieving AIX/ESA* files.

#### 6.3.1.8 Security

AIX/ESA* provides security features to meet the C2 level as defined by the U.S. Department Of Defense (DOD). Among other specifications, the C2 level requires that the users have a password to login, and have control access permission for their files, and also requires that the system administrator may trace an individual user's activities.

AIX/ESA* provides:

- Identification and authentication of users
- A privilege mechanism for system administration
- A security auditing capability.

### 6.3.2 Network Services

AIX/ESA* networking protocols follow the industry standards to allow ease of interconnection and communication. AIX/ESA* uses RISC/6000 and PS/2 gateway nodes to connect to any local-area and wide-area networks supported by TCP/IP in RISC/6000 and PS/2. ASCII devices may be connected through a channel attached RISC/6000, as outboard communication server (OCS), and through the IBM 3172 for Ethernet** and token-ring LAN connected terminals.

Most of the AIX/ESA* communication is based on TCP/IP. AIX/ESA* TCP/IP provides the following functions:

- Terminal Passthrough (TELNET) to allow an user client to remotely logon to another computer with TCP/IP Telnet server function.
- Simple Mail Transfer Program (SMTP) (see section 6.3.4.2, "Application Services" on page 156).
- File Transfer Protocol (FTP) (see section 6.3.4.3, "Data Access Services" on page 156).
- Remote Procedure Call (RPC) for distributed applications.
- The socket interface from Berkeley Software Distribution (BSD**) and the UNIX** sockets, both for writing network applications.

IBM has made a statement of direction (SOD) to implement OSI standards in AIX/ESA*.

Table 3 summarizes the AIX/ESA* networking capabilities.

*Table 3. AIX/ESA* Networking Capabilities*

| | **Ethernet**** | **TokenRing** | **FDDI** | **X.25** | **SDLC** |
|---|---|---|---|---|---|
| **TCP/IP** | YES | YES | YES | YES | - |
| **OSI** | SOD | SOD | SOD | SOD | - |

**Note:** SOD (statement of direction)

### 6.3.3 Distributed System Services

#### 6.3.3.1 Communication Services

The communication services capabilities for AIX/ESA* are provided for:

- Remote Procedure Call

  IBM has made a statement of direction for implementing the OSF/DCE** technology in AIX/ESA* for Remote Procedure Call (RPC).

#### 6.3.3.2 Distribution Services

- Directory

  IBM intends to include in AIX/ESA* the OSF/DCE** for directory services.

- Security

  IBM intends to include in AIX/ESA* the OSF/DCE** for security services.

- Time

  IBM intends to include in AIX/ESA* the OSF/DCE** for time services.

### 6.3.4 Application Enabling Services

The application enabling services available with AIX/ESA* are described in this section.

#### 6.3.4.1 Presentation Services

- User Interface

  The IBM AIXwindows Environment/ESA* (AIXwindows 5696-150) provides a graphical extension to AIX/ESA* and allows an user to interact with other systems providing the X-windows** system. The OSF/Motif** interface is also available.

- Print/View

  AIX/ESA* can be connected through TCP/IP to a PS/2* that runs the Print Service Facility (PSF) under OS/2*. PSF allows the output data to be directed to a wide range of printers.

#### 6.3.4.2 Application Services

- Transaction Monitor

  At present, no IBM transaction monitor has been ported to AIX/ESA*. UNIX System Laboratories has expressed its intent to port its TUXEDO** transaction processing system to AIX/ESA*.

- Mail

  AIX/ESA* provides the Simple Mail Transfer Protocol (SMTP) to send and receive mail over a TCP/IP network.

#### 6.3.4.3 Data Access Services

- Relational

  At present, no IBM distributed database has been ported to AIX/ESA*. Database technology is available from vendors, such as Ingres**, Sybase**, and INFORMIX**.

- File

AIX/ESA* supports industry standard file access protocols such as:

- Network File System (NFS** 5696-149) allowing a user to have transparent file access across UNIX** , MVS, VM, and AIX software platforms.
- File Transfer Protocol (FTP) allowing transfer of files to and from remote workstations and hosts through a TCP/IP network.

## 6.3.5 Application Development

The AIX/ESA* shell provides the traditional development environment for UNIX** based systems. This includes editors like VI and ED, and tools like AWK for pattern scanning, and SCCS (Source Code Control System) for source code change management.

## 6.3.6 System Management

SystemView* is the IBM systems management strategy for managing, planning, coordinating, and operating open, heterogeneous distributed systems. SystemView* includes a SystemView* structure and the SystemView* conforming products. The SystemView* structure is designed to provide a consistent end-user interface, shared data, enhanced automation, and increased integration among systems management products. The structure embraces selected open standards and architectures and defines conformance criteria for the SystemView* products. The SystemView* conforming products are, and will be, developed by IBM and non-IBM vendors.

### 6.3.6.1 Performance Management and Accounting

System performance data is available in AIX/ESA* with AIXMON and the AIX X-Windows Real Time Monitor (XRTM 5665-101). AIXMON collects typical UNIX** data, and also collects samples for the utilization of system resources, such as processor storage, vector, and I/O units. XRTM is a real-time monitor, based on AIXwindows, and offers enhanced graphical presentation of performance data. A threshold notification service allows a user to set values for specific occurrences and receive a warning when these thresholds are reached.

### 6.3.6.2 Operations

For network management, AIX/ESA* provides the agent function of Simple Network Management Protocol (SNMP). The SNMP manager on a RISC/6000 can provide network ownership functions with NetView/6000*.

### 6.3.6.3 Availability and Integrity

AIX/ESA* provides the following facilities for maximum availability:

- Disk mirroring provided by the logical volume manager. Once mirroring is activated, management of the data and copies is automatic and transparent to the application and user.
- Software recovery through kernel recovery, with the objective of isolating the error and allowing the system to continue processing.
- Hardware and device driver recovery. If a processor in a multiprocessor configuration fails, the failing processor is taken offline, and processing continues on the remaining processors.
- Reduced maintenance disruption due to the ability to dynamically add device drivers without requiring a kernel rebuild or reIPL.

### 6.3.7 Selected APIs, Protocols, and Facilities

Table 4 summarizes selected protocols, facilities, and APIs available with AIX/ESA* and mentioned in this section.

*Table 4. AIX/ESA* Selected APIs, Protocols and Facilities*

| | **AIX/ESA*** |
|---|---|
| **Berkeley** sockets** | YES |
| **POSIX**** | YES |
| **CPI-C** | - |
| **APPC** | - |
| **APPN** | - |
| **NCS**** | YES |
| **NQS**** | YES |
| **OSF/DCE**** | SOD |
| **SQL** | SOD |
| **DRDA-RUOW** | - |
| **NFS**** | YES |
| **FTP** | YES |
| **FTAM** | SOD |
| **OSF/Motif**** | YES |
| **X-windows**** | YES |
| **NetWare** server** | - |
| **LAN & workstation server** | - |
| **SMTP** | YES |
| **NetView*** | - |
| **SNMP** | YES |
| **Note:** SOD (statement of direction) | |

## 6.4 IBM PC DOS

IBM PC DOS is an entry-level operating system designed for personal computers. It provides a single user, single-task environment which runs on microprocessors based on the Intel** hardware architecture.

IBM PC DOS results from the evolution of DOS V.1 operating system designed and developed by Microsoft** for IBM in 1981 for the first PC generation. In this chapter, the term DOS will be used to indicate IBM PC DOS V.6.3 (5871-AAA) and below. The content of this section, however, also applies to MS-DOS**, the DOS version marketed by Microsoft**.

Figure 58 shows the elements of the PC DOS software platform.

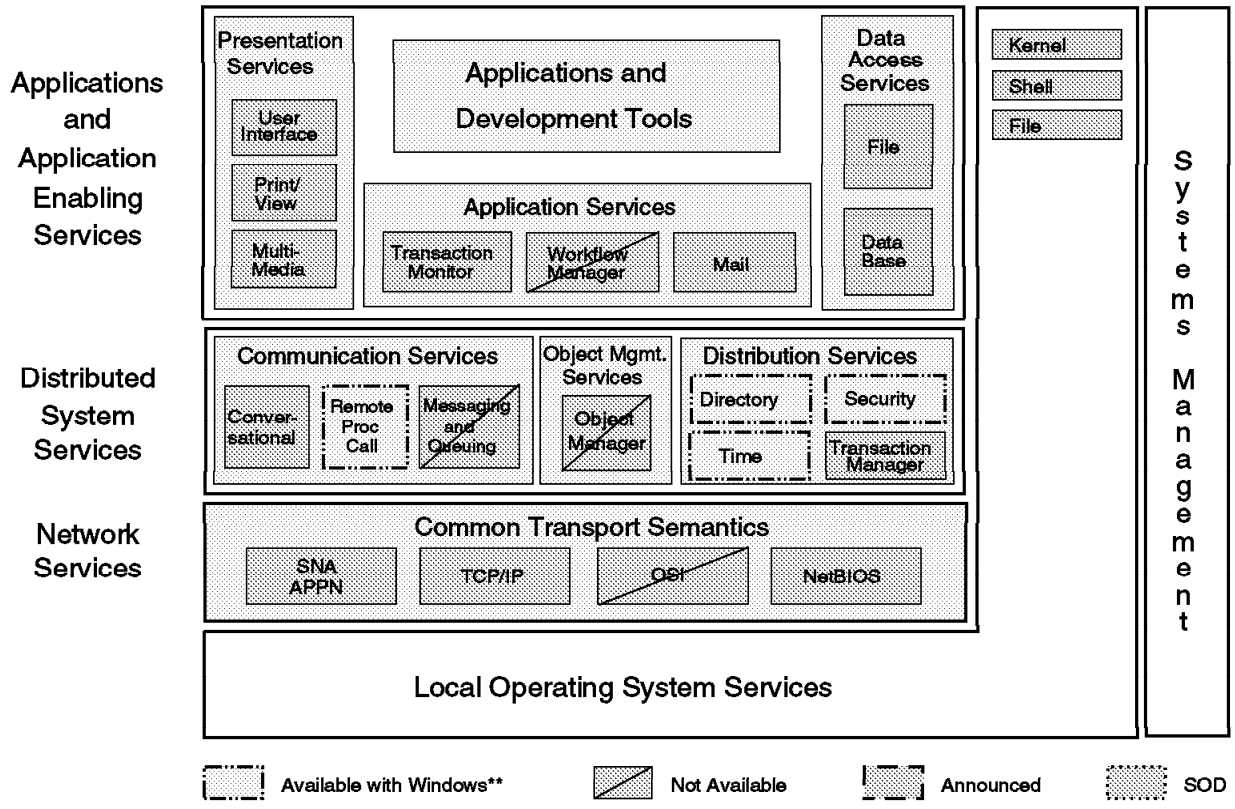


*Figure 58. Elements of the PC DOS Software Platform*

### 6.4.1 Local Operating System Services

The local operating system services of PC DOS can be described referring to the kernel, the shell, and the file system.

#### 6.4.1.1 Kernel

The system architecture and characteristics of the IBM PC Disk Operating System are as follows:

- Conventional memory.

  Personal Computers using DOS can manage 1 megabyte of memory. Only 640KB of this conventional memory is available for use by DOS and applications. The rest of the conventional memory is occupied by video display memory, device buffers, and read-only memory. This later part contains the Basic Input Output System (BIOS), also called the hardware interface code.

  The limitation of a 1MB address space is a carry over from the original IBM Personal Computer, which used an Intel 8088 microprocessor only capable of 20-bit addressing.

- Extended memory.

  Memory above 1MB is called “extended memory” and is only available on systems that use the Intel 80286 processor or above. In order to use this memory, the Intel 80286 processor (or above) must be switched from real mode to protected mode. Real mode of an Intel 80826 is similar to a faster Intel 8088 processor with access to conventional memory and limited access to extended memory. In protected mode, the processor has total access to both conventional and extended memory.

  The mode is a function of the operating system. DOS is a real mode operating system (OS/2 is a protected mode operating system). DOS, through the BIOS, has extended memory service functions, which allow access to extended memory via a fairly slow processor mode switching between real and protected mode.

- Expanded memory.

  This type of memory is the result of the work done by Lotus, Intel, and Microsoft and is called the Lotus/Intel/Microsoft Expanded Memory Specification, or LIM EMS for short. This memory can be installed on all PC′s from the Intel 8088 upwards. Expanded memory consists of two components, a memory manager and the memory itself. The memory manager may make up to 32MB of memory in the 1MB conventional memory space. The expanded memory can be remapped by the expanded memory manager to extended memory.

- The DOS kernel is loaded from disk or diskette, it is called IBMDOS.COM (MSDOS.SYS) and provides the heart of the operating system. It provides hardware independent functions, like file and directory management, memory functions, character input and output device support, and program management. The DOS kernel uses BIOS functions to communicate with the hardware. On top of the kernel is the command processor (COMMAND.COM) which provides the command interpreter and the familiar C> prompt.

- The DOS code is not protected from the application, and an application error may cause a system failure.

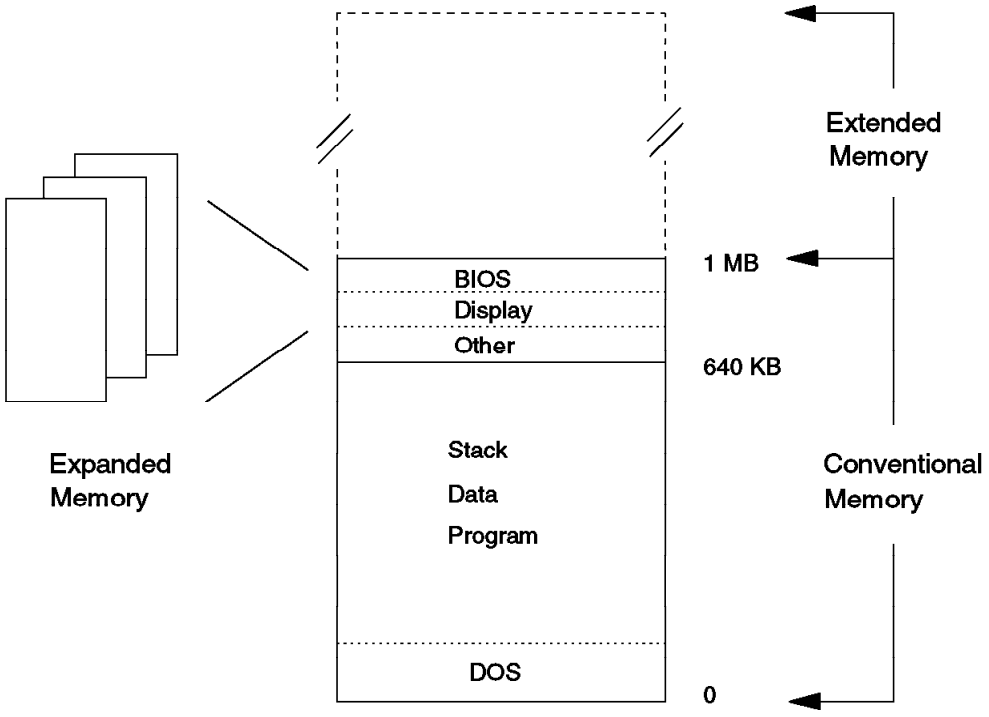


*Figure 59. DOS Real Storage Addressing Range*

Figure 59 shows the real storage addressing range used by DOS and by the applications.

- The file system used by DOS allows the user to organize the files on hard disks or diskettes into tree-structured directories. A hard disk can be subdivided into logical disks or partitions, each with its own directory.
- The Basic Input/Output System (BIOS) provides a compatible interface between the operating system, the software, and the hardware. BIOS allows the application programmer to request an I/O operation independently of the physical details of the hardware architecture. I/O operations cannot be overlapped with instruction execution.
- Device drivers can be written to handle the specific requirements of a particular device. Device drivers make it possible to include support for new devices when the need arise.
- I/O devices are connected to central storage through one channel bus.

### 6.4.1.2 Shell

The DOS shell provides a simple Graphical User Interface (GUI) providing the user with graphical representations of underlying file objects. The shell GUI allows the user to manage programs, view the directories and the contents of the directories, and navigate through them with a graphical interface.

#### 6.4.1.3 Workload Scheduler

DOS, being a single user, single task operating system, executes one application at a time, and therefore, does not normally need a sophisticated workload scheduler. Although a limited manual task swapper is available and also a task scheduler utility program to run DOS tasks at timed intervals.

#### 6.4.1.4 File

The file system available with DOS is File Allocation Table (FAT), which supports unformatted data. Several disk caching programs are available to improve disk performance on systems with more than 1MB of storage.

### 6.4.2 Network Services

Network communications are implemented in DOS by:

- LAN Support Program (5621-300)
- TCP/IP for DOS (5621-219)
- Network Services/DOS (5621-344)
- PC/Host File Transfer and Terminal Emulator Program (FTTERM 5669-352).

These products allow DOS to handle local and remote networking communication with PCs running network protocols, and with other software platforms.

In addition, Personal Communications/3270 (PC/3270, 5622-231) provides 3270 emulation functions for communications with S/370* hosts.

The IBM DOS LAN Requester (DLR) is available with the OS/2 LAN Server (5621-253) for access to the OS/2 LAN Server. DLR also includes the LAN Support Program, which provides a common interface across the LAN for both NetBIOS and token-ring protocols.

TCP/IP for DOS provides the following services and application programming interfaces:

- Terminal passthrough (TELNET) to allow a DOS client to remotely logon to another system with a TCP/IP TELNET server function.
- Simple Mail Transfer Protocol (SMTP) to allow an end-to-end electronic mail exchange.
- SNMP agent function.
- Windows Sockets API with the programmer's toolkit (5621-256).
- Allows existing NetBIOS applications to be routed through a TCP/IP network.
- File Transfer Protocol (FTP) (see section 6.4.4.3, "Data Access Services" on page 165).
- Network File System (NFS**, 5621-257). (see section 6.4.4.3, "Data Access Services" on page 165).
- Line Printer Client (LPR) for remote printer support. (see section 6.4.4.1, "Presentation Services" on page 164).
- X-windows** server function for client applications.
- Allows existing NetBIOS applications to be routed through a TCP/IP network (5622-048).

The LAN Distributed Platform for DOS (LANDP/DOS 5622-107) provides a client/server distributed programming capability well suited to develop distributed services and applications in an heterogeneous LAN environment. The LANDP platform allows integration of heterogeneous systems (DOS, Windows, OS/2*, OS/400* and AIX/6000*) and heterogeneous communication environments (NETBIOS, TCP/IP, SNA, and X.25). In addition to the programming services, LANDP also provides several ready-to-use functions, such as 3270 emulation, shared file, electronic journal, store and forward services and management services.

Network Services/DOS provides CPI-C and APPC interfaces for the DOS platform. DOS can participate in an SNA subarea environment and in Advanced Peer-to-Peer Networking (APPN).

FTTERM provides terminal emulation and file transfer to EBCDIC and ASCII hosts.

Table 5 summarizes the DOS networking capabilities.

*Table 5. DOS Networking Capabilities*

| | Ethernet** | TokenRing | FDDI | X.25 | SDLC |
|---|---|---|---|---|---|
| **TCP/IP** | - | YES | - | YES | - |
| **NetBIOS** | YES | YES | YES (1) | - | - |
| **SNA** | - | - | - | - | YES (2) |

**Note:**
(1) Through LAN Support Program (5621-300)
(2) With an SDLC adapter card

## 6.4.3 Distributed System Services

### 6.4.3.1 Communication Services

The communication services capability within DOS and Microsoft** Windows** are provided for:

- Conversational

  Inter-process communication services for distributed resource managers on a LAN is provided by the LAN Support Program which, using the IEEE 802.2 interface, makes the Advance Program-to-Program Communication for the IBM PC (APPC/PC) available on a token-ring network and a PC network.

- Remote Procedure Call

  IBM DCE Client for Windows** (5696-657) software developer's kit (SDK) provides RPC and threads services necessary to execute a distributed application.

### 6.4.3.2 Distribution Services

- IBM DCE Client for Windows** software developer's kit (SDK) provides the following distribution services:
  - Cell Directory
  - Security
  - Time.

- Transaction Manager

  The CICS* for OS/2 transaction monitor (see below) provides transaction manager functions to ensure transactions and data integrity across distributed systems.

## 6.4.4 Application Enabling Services

### 6.4.4.1 Presentation Services

- User Interface

  With TCP/IP for DOS, the X-windows** server allows client applications to use the graphical interface on the DOS workstation screen.

- Print/View

  TCP/IP for DOS delivers the Line Printer Client (LPR) function that provides client support for distributed printing. LPR sends data to be printed to Line Printer Daemon (LPD) on a specified server platform to a specified printer. The LPD server on the target software platform provides access to the attached printers.

- Multimedia

  IBM Storyboard Live! (5604-408) is a multimedia application that allows the mixing of video, audio, animation and text to produce high quality screen presentations.

### 6.4.4.2 Application Services

- Transaction Monitor

  CICS* OS/2* V.1 (5688-101) provides CICS* transaction management facilities to a DOS software platform. It can operate as a client for CICS OS/2 LAN server, with other CICS family systems, or as a stand-alone system.

  CICS applications can communicate with CICS applications on other software platforms through several mechanisms. This capability exists across all the CICS monitor family (AIX/6000*, OS/2, OS/400*, MVS/ESA*, VSE/ESA*). These mechanisms allow CICS to route transactions to another CICS for execution, allow for transparent access to remote CICS resources, and can invoke a remote CICS application.

  The CICS APIs for distributed transactions are:

  – CICS COBOL and C command-level

  – Transaction routing, and outbound function shipping over NetBIOS from the personal workstation to a CICS OS/2 LAN server.

  The CICS daisy-chaining concept is available for access to host systems.

- Mail

  The Simple Mail Transfer Protocol (SMTP) application of TCP/IP provides client support for transferring electronic mail messages over the TCP/IP network.

  Mail services are also available with other vendor products, such as Lotus Notes** and cc:Mail**.

#### 6.4.4.3 Data Access Services

- Relational

  IBM DATABASE2 for OS/2* (5622-044) is a data base management member of the IBM relational database family. It can be used in a single-user workstation and in a client/server LAN environment. It allows access to an OS/2 database server from DOS and Windows**. The Database Manager provides SQL capabilities and roll forward recovery for database recovery from a system failure. It includes functions for application portability and DB2 compatibility.

  IBM DB2 for OS/2 and Distributed Database Connection Services for OS/2* (DDCS 5622-056) provide database access for decision support and OLTP distributed applications. DDCS/2 implements the distributed relational database architecture (DRDA) at the level of remote unit of work (RUOW).

  The distributed database client enabling feature of IBM DB/2 for OS/2 can be utilized in DOS environments. When combined with DDCS on an OS/2 server, it gives a DOS workstation the capability to access and update remote relational databases on platforms that use the DRDA architecture.

- File

  Services to access and manage distributed data are provided by the following:

  – TCP/IP File Transfer Protocol (FTP) allows file transfer across systems. FTP supports the transfer of both binary and ASCII files.

  – TCP/IP Network File System (NFS**) allows users to have transparent access to hierarchical file structures. File sharing is also supported.

### 6.4.5 Application Development

Several editors, compilers, and debuggers are available for application development on the DOS software platform.

### 6.4.6 System Management

SystemView* is the IBM systems management strategy for managing, planning, coordinating, and operating open, heterogeneous distributed systems. SystemView includes a SystemView structure and the SystemView conforming products. The SystemView structure is designed to provide a consistent end-user interface, shared data, enhanced automation, and increased integration among systems management products. The structure embraces selected open standards and architectures and defines conformance criteria for the SystemView products. The SystemView conforming products are, and will be, developed by IBM and non-IBM vendors.

#### 6.4.6.1 Operations

The following are some of the programs that allow management of DOS workstations connected to a LAN:

- NetView/PC (5669-024) allows implementation of the NetView service point concept on the DOS software platform. It collects network management information, formats it into alerts, and forwards alerts to NetView for centralized network management. Host connectivity is through a Real Time Interface Coprocessor (ARTIC**).

- LAN Network Manager (5621-117) and LAN Station Manager (5621-103) provide heterogeneous LAN management capabilities for token-ring and PC

network attached stations. LAN Network Manager is conforming to SystemView, has graphical display, runs on OS/2 workstations and cooperates with host based Netview for integrated network management. LAN Station Manager can be run on DOS.

### 6.4.7 Selected APIs, Protocols, and Facilities

Table 6 summarizes selected protocols, facilities and APIs available with DOS and mentioned in this section.

*Table 6. IBM PC DOS Selected APIs, Protocols, and Facilities*

| | **IBM PC DOS** |
|---|---|
| **Berkeley** Sockets** | - |
| **POSIX**** | - |
| **CPI-C(1)** | YES |
| **APPC(1)** | YES |
| **APPN(1)** | YES |
| **NCS**** | - |
| **NQS**** | - |
| **OSF/DCE**** | - |
| **SQL(2)** | YES |
| **DRDA-RUOW(2)** | YES |
| **NFS**** | YES |
| **FTP** | YES |
| **FTAM** | - |
| **OSF/Motif**** | - |
| **X-windows**** | YES |
| **NetWare** server** | - |
| **LAN & workstation server** | - |
| **SMTP** | YES |
| **NetView*** | YES |
| **SNMP** | YES |

**Note:**
(1) With Network Services/DOS
(2) Through DB2 for OS/2* client enabler on DOS and DDCS on OS/2* server

## 6.5 MVS/ESA*

MVS/ESA* (V4 5695-047, 048 and V5 5655-068, 069) is a general purpose operating system that runs on the IBM Enterprise Systems/9000* (ES/9000*) processor family and all other processors designed to operate with the Enterprise Systems Architecture/390 (ESA/390*). Among the primary objectives of MVS/ESA* are high performance, security, reliability, and availability in large system environments.

MVS/ESA* makes use of advanced technologies in major functional areas that are critical to implement system-wide solutions. These cover data management, networking, transaction processing, workload scheduler, workload management, security, system management, client/server computing, and application development. MVS/ESA* can manage thousands of users on large ES/9000* systems, and provides high I/O bandwidth and large storage capacity. In addition, MVS/ESA* V5 supports the S/390 Parallel Transaction Server 9672, a solution for online transaction processing.

Figure 60 shows the elements of the MVS/ESA* software platform

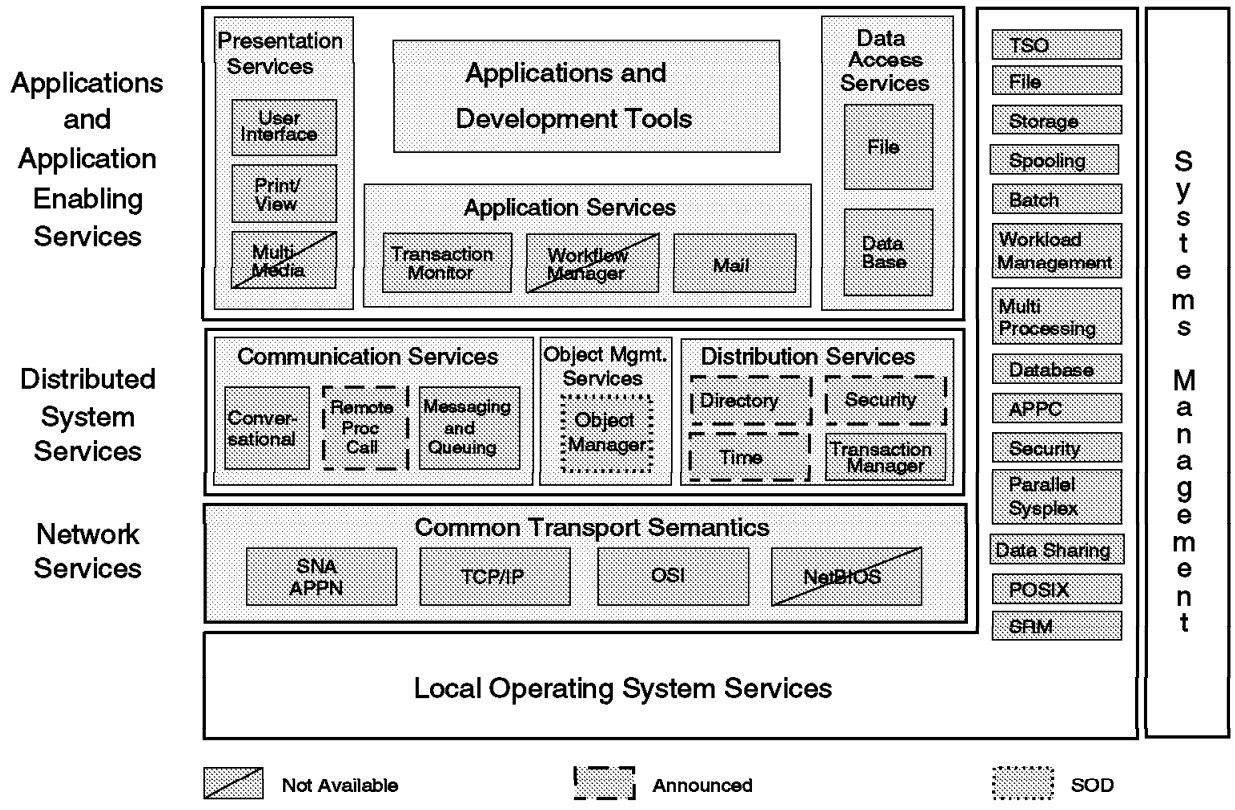


*Figure 60. Elements of the MVS/ESA* Software Platform*

### 6.5.1 Local Operating System Services

The system architecture and characteristics of MVS/ESA* are the following:

- MVS/ESA* (Multiple Virtual Storage/ESA*) provides each user of the system with one (or more) private virtual storage address space of up to 2GB in size. The address space is used for user data and programs, and each address space is isolated from the others by the software and the architecture.
- MVS/ESA* users also have the capability to define additional data spaces of 2GB each and hiperspaces of up to 16TB each for storing data.

  Figure 61 shows the MVS/ESA* virtual storage environment.

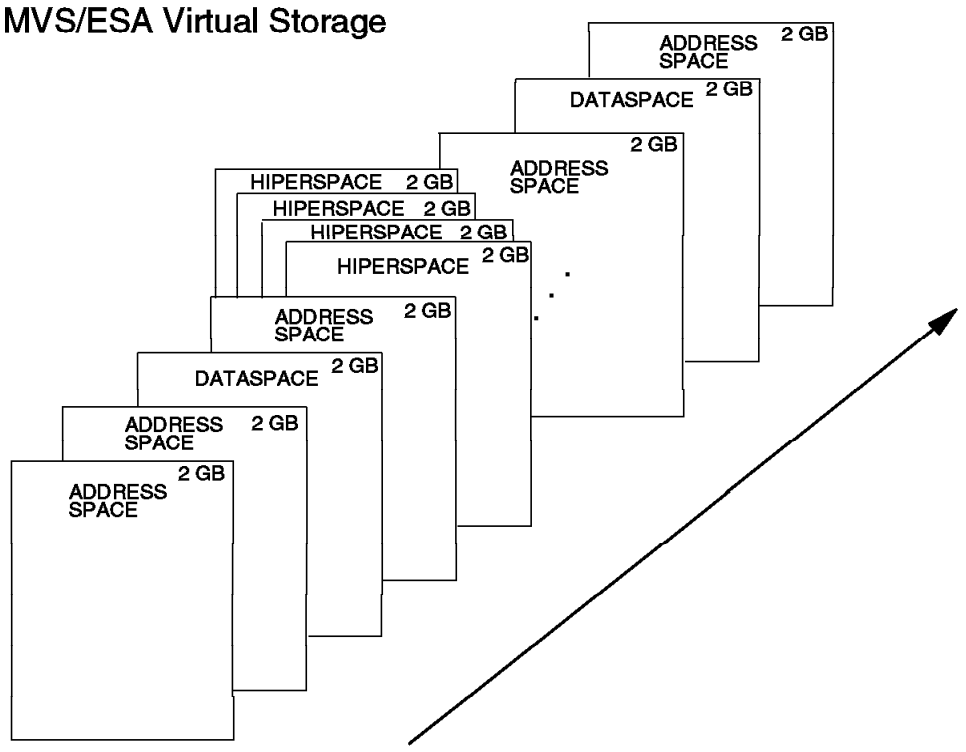


*Figure 61. MVS/ESA* Virtual Storage*

- MVS/ESA* can use the expanded storage feature of the ES/9000* and 9672 processors for backing the virtual storage of the address spaces, data spaces, and hiperspaces.
- The largest central storage is 2GB, while the total system storage size can be up to 16TB.
- MVS/ESA* cross memory services allow authorized users to access other address spaces for program and data reference.
- Virtual-to-real storage relationships are controlled by various paging algorithms, where paging is the process of moving the content of real storage, in blocks of one or more pages of 4KB each, into auxiliary storage

(external DASD devices), or into expanded storage, or vice-versa, from auxiliary or expanded storage to real storage.

- Swapping is used to control the number of users concurrently using real storage and system resources, where swapping is the process of transferring all the real storage belonging to a user into auxiliary or expanded storage and vice-versa.
- MVS/ESA* supports tightly-coupled multiprocessors with up to ten processors and allows multi-processing through multi-tasking.
- The job entry subsystems, JES2 and JES3, both provide facilities for loosely coupling up to eight MVS images with MVS/ESA* V4, and 32 MVS images with MVS/ESA* V5.
- MVS/ESA* V4 supports the concept of Sysplex (system complex), a combination of hardware elements and software services to couple up to eight MVS systems together and still provide a single image computing facility.
- MVS/ESA* V5 supports the S/390 Parallel Sysplex, a structure for enabling parallel processing and data sharing across systems. Up to 32 systems can be configured in a S/390 Parallel Sysplex. This configuration adds the multi-system capability to multi-processing through multi-tasking.
- High performance data sharing is made possible through the Coupling Facility technology: a combination of hardware and software functions supported by MVS/ESA* V5. In a S/390 Parallel Sysplex, authorized applications, such as subsystems and MVS/ESA* V5 components, use the coupling facility services to cache data, share queue status, and access sysplex lock structures to implement data sharing, subsystems workload balancing, and rapid recovery from failures. The subsystems and MVS components transparently provide data sharing, workload balancing, and recovery benefits to their applications.
- MVS/ESA* permits the use of the Enterprise System Connection (ESCON*) architecture and also supports the specialized connection High-Performance Parallel Interface (HPPI) for transmission of data at peak rates greater than 50MB/sec.
- The number of I/O devices locally attachable with MVS/ESA* V4 is 4096, and a single device may be connected with up to eight channel paths. MVS/ESA* V5 extends the practical limit to 5500 devices and the theoretical limit to 64K devices.
- The I/O bandwidth possible with MVS/ESA*, ES/9000 and 9672 processor families is in the order of hundreds of MB of data per second.
- MVS/ESA* provides features for sharing DASD devices with other MVS systems.
- MVS/ESA* allows the use of the following hardware facilities:
  - The vector facility feature of the ES/9000* processor family for computer intensive engineering and scientific applications.
  - The integrated cryptographic resource facility (ICRF) of the ES/9000* processor family for applications requiring a high level of data security.
  - The data compression facility of selected ES/9000* and 9672 processor families for users requiring DASD space saving and communication line transmission time saving.

  - The subsystem storage protection and the subspace group facility to enhance CICS* availability.

- MVS/ESA* V4 and V5 support the standards defined by POSIX** 1003.1 (system interface), 1003.2 (shell and utilities), and 1003.4a (real-time threads), all conforming to XPG4**. These services are provided as part of the OpenEdition* MVS.
- IBM intends to seek XPG4 Base Branding and the X/OPEN** specification 1170 for OpenEdition* MVS.

#### 6.5.1.1 Workload Management

MVS/ESA* provides scheduling facilities for the following categories of workload:

- Interactive workload, which is scheduled using the Time Sharing Option (TSO) facilities.
- Batch workload, which is scheduled by the job entry subsystem (JES). The JES accepts work (jobs) into the system, selects jobs for execution, processes the output, and purges work from the system. Work can be submitted locally by the operator, by TSO users, by RJE workstations, by programs, by other software platforms utilizing the Network Job Entry (NJE) service of JES, and by UNIX** type workstations using NQS (see section 6.5.2, “Network Services” on page 173).
- Online transaction workload, which is scheduled by transaction monitors such as CICS* and IMS.
- Transaction programs, which result from APPC conversations and are scheduled by the specialized MVS service facilities, which are part of the implementation of the SNA LU 6.2 architecture.

The last two workloads are related to distributed system services or to distributed applications.

The utilization of system resources is managed by a system component called the System Resources Manager (SRM). When system resources are overcommitted, the SRM determines their usage based on installation specifications. The installation can instruct the SRM to achieve throughput objectives or response objectives or a mixture of them, and the SRM manages the workload accordingly. Users may be swapped in and out of the system if necessary.

The Workload Manager is an MVS/ESA* V5 system component that allows the installation to define performance goals for the different categories of workload through service policies. This is different from the SRM installation specifications where the user defines how to distribute resources among the categories of workload. With the Workload Manager, the system dynamically allocates resources to meet the performance goals.

In a S/390 Parallel Sysplex environment, the MVS Workload Manager provides, to requesting subsystems, information about current local system performance. This allows the subsystem to make its own management decisions. In a CICS* environment, CICSPlex System Manager/ESA* (CP/SM) (5695-081) cooperates with the Workload Manager and VTAM V 4.2 to dynamically route the CICS* transactions to meet performance goals and to enhance CICS* applications availability.

IBM intends to improve IMS/ESA* TM for the S/390 Parallel Sysplex environment to benefit workload balancing and enhanced availability.

### 6.5.1.2 Spooling

The Simultaneous Peripheral Operations On-Line (Spool) facility is provided by JES and provides temporary storage for user's related data, mainly input for batch execution and output data for later printing. JES provides spool interfaces for all categories of workload: batch, online transaction, TSO users, and transaction program conversations. JES can route the spool output data to local or remote printers connected to the NJE network, thus acting as a print server. The spool facilities of MVS, VM, and VSE systems can be connected over a Network Job Entry (NJE) network for distributed management of spool data.

### 6.5.1.3 Database

DB2* (5665-DB2) is the relational database management system for the MVS/ESA* platform. It may be used to implement both decision support systems and traditional transaction applications. The language access interface is the Structured Query Language (SQL). The Query Management Facility/MVS (QMF/MVS 5706-254) is available as a high-level SQL based query interface with graphic functions.

IMS/ESA* Database Manager (5685-012) is the hierarchical database manager for MVS/ESA*. This database is used by IMS and CICS* transactions and batch applications for local data storage and data manipulation.

IMS/ESA* V5 DB (5695-176) supports the coupling facility to allow block level data sharing in a S/390 Parallel Sysplex. Up to 32 systems can concurrently access IMS databases.

IBM intends to further enhance DB2, IMS/ESA*, CICS/ESA*, DFSMS/MVS* and MVS/ESA* to provide shared data function in a parallel sysplex environment.

There are other vendor products, such as Adabas**, IDMS** and Oracle**, which offer database manager services.

### 6.5.1.4 File

The file services provide storage and access for customer data used by local applications. The file organization can be formatted data (record file) or unformatted data (byte-stream).

DFSMSdfp (part of DFSMS/MVS* 5695-DF1) provides the data management services for MVS/ESA*.

DFSMSdfp allows record data set organization types, such as indexed (VSAM), sequential, partitioned, and direct, and allows data set sharing across multiple applications and multiple MVS systems.

DFSMSdfp provides services, through the Object Access Method (OAM), to access and manage data entities in the form of bit strings, called objects. One specific usage of OAM is with image processing applications.

The Parallel I/O Access Method (PIOAM 5685-137) is a software file striping method that allows for very high data transfer rates for sequential data sets.

MVS/ESA* also provides generation data set groups to allow for tracking and maintaining versions of data, to provide protection and recovery from application logic errors, and to simplify management of repetitive batch jobs.

DFSMSdfp permits the use of a hierarchy of storage devices. The device types include disks, with the advanced functions provided by the IBM 3990 Model 3 and 6 DASD control units (dual copy, DASD fast write, concurrent copy, data striping, and enhanced cache management), tapes, the IBM 3495 tape library, the IBM 9570 disk array, and the IBM 3995 optical library dataserver.

MVS with OpenEdition* services provides POSIX** standards as a base for portability in a heterogeneous environment. With the implementation of standard POSIX** 1003.1, the byte-stream hierarchical directory organization, used in the UNIX** system, is included in DFSMSdfp.

#### 6.5.1.5 Storage Services

Storage services for the MVS/ESA* platform are provided by the Data Facility Storage Management Subsystem for MVS (DFSMS/MVS*). DFSMS/MVS* functions are provided by the following products:

- DFSMSdfp for the management of active data
- DFSMShsm for the management of low activity or inactive data
- DFSMSdss for data movement
- RACF (see section 6.5.1.6, “Security”) for security of data
- Data Facility Sort (DFSORT 5740-SM1) for sorting of data.

DFSMS/MVS* exploits the concurrent copy capability of the IBM 3990 Model 3 and 6 DASD control units, which allows concurrent and unrestricted access to an application′s active data during backup processing.

The Remote Recovery Data Facility (RRDF 5798-RXX) reduces the time required to get a remote site in operation in case of a disaster at the primary site.

The Remote Site Recovery (RSR) is a feature of IMS/ESA* V5 (5695-176) and provides remote recovery for IMS/ESA* V5 Database Manager and IMS/ESA* V5 Transaction Manager. It automates transport, storage and processing of log data, recovery information and optionally, replication, of databases.

#### 6.5.1.6 Security

Security services in MVS/ESA* are provided by the Resource Access Control Facility (RACF, V1 5740-XXH, V2 5695-039). A secure environment up to the specifications known as the B1 level of the U.S. Department Of Defense (DOD) can be provided.

RACF provides services to protect resources from accidental or deliberate unauthorized access, modification, or destruction. The functions and facilities provided by RACF are, among others:

- User identification and verification by userid and password
- Access authorization checking for system managed resources
- Logging and reporting functions for audit purpose.

Additionally, RACF V2 supports the following:

- S/390 Parallel Sysplex, allowing Sysplex data sharing for the RACF data base using the coupling facility.
- OpenEdition* MVS, allowing user and group registration to the RACF database. This facility provides security checking and auditing for the POSIX** environment.
- RACF Secure Signon enables the moving of end-user authentication from RACF verification processing to another authentication server that could be implemented within a LAN client/server function. With Secure Signon a user can enter one logon password for secured access to multiple MVS applications. With the Secure Signon, RACF now supports the use of an alternative to the RACF password called PassTicket. A PassTicket can be generated by the authenticating server and used instead of the password to authenticate a mainframe application user to RACF. The RACF PassTicket removes the need to send the RACF passwords across the network in clear text.

There are other vendor's products, such as ACF/2** and TOP Secret**, which also operate in the area of system security.

The Integrated Cryptographic Service Facility/MVS (ICSF/MVS, 5685-051) permits the use of the Common Cryptographic Architecture (CCA) and contributes to better data security by providing confidentiality and data integrity. A channel attached cryptographic engine, the IBM 4753 Network Security Processor, is also available.

### 6.5.2 Network Services

MVS/ESA* supports a wide range of connectivity options for the three main networking architectures: SNA, TCP/IP, and OSI.

SNA networking services are provided by ACF/VTAM (V3 5685-085, V4 5695-117) with ACF/NCP (5688-231). Several applications are available in the SNA environment, performing functions such as file transfer, electronic mail, document exchange, and remote logon.

Among others, services provided under SNA architecture are:

- Advanced Program-to-Program Communications (APPC) using SNA LU 6.2 architecture
- The SNA Advanced Peer-to-Peer Networking (SNA APPN) technology.

TCP/IP support in MVS/ESA* (5735-HAL) provides the following functions, applications, and application programming interfaces:

- Terminal passthrough (TELNET) allows a user client to remotely logon to another computer with TCP/IP TELNET server function.
- Simple Mail Transfer Protocol (SMTP) allows an end-to-end electronic mail exchange.
- Simple Network Management Protocol (SNMP) is provided for network management.
- File Transfer Protocol (FTP) (see section 6.5.4.3, "Data Access Services" on page 178).
- Network File System (NFS**) (see section 6.5.4.3, "Data Access Services" on page 178).

- Internet Domain Name Server (DNS) for distributed name services.
- X-windows** and Motif** client facilities allow program access to a high resolution display connected to a system running a graphic server system (see section 6.5.4.1, “Presentation Services” on page 176).
- Remote Procedure Call (RPC) library for high level program-to-program communication for distributed applications.
- Network Computing System (NCS**) allows programmers to distribute processing power to other hosts.
- Berkeley** Socket library for distributed applications communication. OpenEdition(*) MVS POSIX** 1003.1 integrates the sockets API.
- Network Queuing System/MVS (NQS** 5695-168) allows remote batch job submission from a UNIX** workstation using NQS** protocol. It runs as a server for the MVS/ESA* JES2 subsystem.

OSI Communications Subsystem (5685-014) enables MVS to communicate with other systems running a compatible set of OSI protocols.

OSI provides the following functions and applications:

- OSI message exchange provides message handling according to the X.400 standard in a multi-vendor environment (see section 6.5.4.2, “Application Services” on page 176).
- OSI file services (see section 6.5.4.3, “Data Access Services” on page 178).
- The Remote Programming Interface (RPI) for development of distributed applications using COBOL and C languages.

Multi-Protocol Transport Networking (MPTN) architecture supports CTS (Common Transport Semantics) in IBM′s Networking Blueprint. CTS provides a common view of networking protocols and makes applications independent of the underlying network transport. MPTN in MVS/ESA is implemented via AnyNet/MVS (5685-085, VTAM V3 R4.2 and MPTF Multi-Protocol Transport Feature - 5695-117, VTAM V4 R2 and the AnyNet Feature), which allows APPC applications to run over a TCP/IP network and TCP/IP Sockets applications to run over an SNA network.

NetView provides a complete set of host network management services for the MVS environment. It can be used as a single point of control for the three network architectures (SNA, TCP/IP and OSI). In addition, NetView may be used to manage other architectures and devices through other NetView products. This implements the service point concept so that the network management data, including alerts, may be exchanged between NetViews.

The NJE function of MVS/JES provides networking facilities that allow file transfer and remote job entry between MVS, OS/400, VM, and VSE platforms. NJE allows these systems to pass data and operate together (see section 6.5.1.1, “Workload Management” on page 170).

IBM intends to provide the Open System Adapter (OSA) feature of selected ES/9000* and 9672 processor families. OSA will provide native open systems connectivity to LANs from the processor. OSA will support FDDI, TokenRing, and Ethernet** LANs, and offload TCP/IP protocol and NFS** application to improve the efficiency of the processor as a server.

Table 7 on page 175 summarizes the MVS/ESA* networking capabilities.

*Table 7. MVS/ESA* Networking Capabilities*

| | Ethernet** | TokenRing | FDDI | X.25 | SDLC |
|---|---|---|---|---|---|
| **SNA** | YES | YES | YES | YES | YES |
| **TCP/IP** | YES | YES | YES | YES | YES(1) |
| **OSI** | YES | SOD | SOD | YES | YES |

**Note:** (1) Host TCP/IP to host TCP/IP with MTPN

## 6.5.3 Distributed System Services

### 6.5.3.1 Communication Services

The communication services capabilities within MVS/ESA* are provided for

- Conversational

  APIs for conversation according to the SNA LU 6.2 architecture. Two API′s are provided, one compliant with SAA CPI-C, and one specifically implemented for MVS/ESA* known as APPC/MVS Server Facilities, or APPC/MVS for short.

- Remote Procedure Call

  IBM has announced support for OSF/DCE** Remote Procedure Call (RPC) in the base code of MVS/ESA* V5.

- Messaging and Queuing

  Message and Queue Manager MVS/ESA* (MQM, 5695-137) that provides an asynchronous program-to-program interface for distributed applications.

### 6.5.3.2 Object Management Services

IBM intends to include, in MVS/ESA*, Object Management Services by implementing IBM′s System Object Model (SOM) in conjunction with Distributed SOM (DSOM). The services will be based on Common Object Request Broker Architecture (CORBA) as defined by Object Management Group′s (OMG**).

### 6.5.3.3 Distribution Services

- IBM has announced support for the following OSF/DCE** Base Service in the base code of MVS/ESA* V5 for clients/servers, which run on a TCP/IP network:
  - Directory

    To use this service, an OSF/DCE** Cell server is required for each cell. AIX DCE Cell Directory Server/6000 (5795-119) on a RISC System/6000* or IBM DCE Software Developer′s Kit (SDK, 5696-657) on a PS/2* can be used.

  - Security

    To use this service, an OSF/DCE** security server is required. AIX DCE Cell Security Server/6000 (5795-118) on a RISC System/6000* or IBM DCE Software Developer′s Kit on a PS/2* can be used.

  - Time

  These OSF/DCE** facilities will allow interoperability and transport independence for distributed applications.

- Transaction Manager

  CICS* and IMS transaction monitors provide transaction manager functions by employing 2-phase commit schemes to insure transaction and data integrity across distributed systems.

## 6.5.4 Application Enabling Services

The application enabling services available with MVS/ESA* are described in this section.

### 6.5.4.1 Presentation Services

- User Interface

  Applications can be designed to use the presentation server of a X-windows** workstation in a TCP/IP network (APIs are provided with TCP/IP), or ScreenView (5695-047). ScreenView provides graphic and navigation services on an OS/2 platform as well as host connection services for host-workstation cooperative processing. ScreenView is a component of MVS/ESA* (4.2 or later) and provides the common enabling services and facilities necessary to integrate SystemView* conforming applications.

  MVS/ESA* transaction monitors and TSO interactive applications use character display for terminals. MVS/ESA* has a comprehensive series of TSO/ISPF dialogues that provide system programming and operations support through menus and help screens using characters. The transaction monitors use their own formatting maps to interface with the end users.

- Print/View

  The Line Printer Client/Line Printer Daemon (LPR/LPD) functions of TCP/IP provides client/server support for remote printing. The LPR sends data to be printed to the LPD on a specific host server for a specific printer. The LPD server on MVS/ESA* provides local and remote users access to MVS supported printers. The LPD server on workstations with TCP/IP provides access to the attached printers.

  Print Services Facility/MVS (PSF/MVS 5695-040) provides the MVS platform with the printing services as defined by the IBM′s Advanced Function Printing* (AFP*) model and architectures.

  LANRES/MVS (LAN Resource Extension and Services, 5695-123) integrates Novell′s NetWare** LAN Server with the print services by MVS/ESA*.

### 6.5.4.2 Application Services

- Transaction Monitor

  The IBM transaction monitors CICS/ESA* (5655-018), CICS/MVS* (5665-403) and IMS/ESA* TM (V4 5685-013, V5 5695-176) qualify as distributed transaction monitors. CICS* can be distributed through multiple CICS* images, known as regions.

  CICS* applications can also communicate with CICS* applications on other software platforms through several distinct mechanisms. This capability exists across all of the monitors in the CICS* family of products (AIX/6000, MVS/ESA*, OS/2*, OS/400*, VSE/ESA*). These mechanisms allow CICS* to route transactions to another CICS* for execution, allow for transparent access to remote CICS* resources, and to invoke a remote CICS* application.

IMS can provide distributed transaction monitor functions by routing transactions to another IMS, utilizing the multiple system coupling facility.

The following APIs are available to assist in developing remote transaction processing in CICS* and IMS environments:

- CICS* transaction routing, function shipping, and distributed program link.
- CICS* implementation of LU 6.2 protocol and interface with CPI-C.
- CICS* to TCP/IP sockets interface to enable CICS* applications to interoperate with partner applications in other platforms connected to a TCP/IP network.
- CallPath CICS/MVS* (5695-089) to communicate with industry standard private branch exchanges for telephone applications.
- Message and Queue Manager for MVS/ESA* (MQM MVS/ESA 5695-137) is an asynchronous program-to-program interface that can be used for CICS* and IMS applications. MQM for MVS/ESA* uses CICS* inter-system communication to provide transportation between queue managers.
- The OpenEdition* DCE** application Server/CICS will provide application access to CICS* host applications from client workstations, using the DCE** Remote Procedure Call.
- IMS transaction routing.
- IMS interfaces with APPC/MVS and CPI-C.
- The OpenEdition DCE** Application Server/IMS will provide application access to IMS host applications from client workstations, using the DCE** Remote Procedure Call.

CICS* and IMS transaction monitors have logs to record all changes. This enables recovery of transactions and data if an unrecoverable DASD error or system failure occurs. Furthermore, MVS/ESA* supports the extended recovery facility (XRF) for CICS* and IMS. XRF provides early failure detection and automatically transfers control to a hot standby CICS* or IMS.

In a S/390 Parallel Transaction environment CICSPlex System Manager/ESA* (CP/SM) can, in case of a CICS* region failure, dynamically route transactions to other CICS* regions to enhance CICS* applications availability.

IBM intends to improve IMS/ESA* TM for the S/390 parallel transaction environment to enhance IMS applications′ availability.

- Mail

  A function, provided by OSI message exchange, that allows message handling according the X.400 standard in a multi-vendor environment.

  OfficeVision/MVS* (OV/MVS* 5685-106) with the X.400 DISOSS connection (MVS 5785-GCF) operates as a CICS* application, can communicate with the OSI message exchange using the X.400 protocol, and can send messages through the network, following the X.400 standard.

### 6.5.4.3 Data Access Services

- Relational

  DB2* (5665-DB2) is the relational data base management system for the MVS platform. DB2* participates in the Distributed Relational Database Architecture (DRDA) with platforms that implement the Remote Unit Of Work (RUOW). This gives the DB2* user the ability to access and update other relational databases in operating environments that use the same architecture. The IBM software platforms where the programming for RUOW has been developed are AIX/6000, OS/2 (client-only function), VM/ESA, VSE/ESA, and OS/400.

  There are other vendor's products, such as Oracle**, that operate in the area of distributed database management.

- File

  Services to access and manage distributed data are provided by the following TCP/IP applications and by DFSMSdfp:

  – File Transfer Protocol (FTP) allows transfer of files to and from a remote host, using the same command syntax as UNIX**, and also allows job submission to MVS JES.

  – Network File System (NFS**) allows a user to have transparent access to formatted files and byte-stream files distributed across the network. Workstations user can treat the OpenEdition* MVS hierarchical file system as an extension of their own file system. An MVS platform can act as a file server for platforms that have the NFS** 3.2 function installed.

  – The Distributed FileManager/MVS (part of ADSM*) provides distributed file access services in heterogeneous distributed environments. Data accessed can be record or byte format. ADSM* operates with SAA-based systems, non-SAA, and non-IBM platforms that support the Distributed Data Management Architecture (DDM). DDM architecture defines the target system as the system where data resides, and the source system as the system where the application resides. The MVS platform provides the functions of a DDM target system through the CICS* DDM support. DDM source system functions are implemented by the Distributed FileManager on the OS2 V2 platform.

  The OSI communication subsystem provides an application for remote data access.

  – OSI File Services (5685-046) allows the exchange and remote management of files between platforms that implement an equivalent set of file transfer, access, and management (FTAM) protocols.

  LAN File Services/ESA (5648-039) provides a file system on MVS/ESA* that is compatible with file systems on LAN workstations. The Workstation file system on MVS supports DOS, OS/2* and UNIX** format files, file operations, multiple directories and locking. Workstations on a LAN can have shared files on the MVS/ESA* platform for storage intensive applications.

  LAN Resource Extension and Services (LANRES 5695-123) integrates Novell's NetWare** LAN Server with the storage facilities provided by MVS/ESA*. High-speed connectivity is achieved by attaching the LAN server station to an ES/9000* channel. The functions and facilities provided by LANRES are:

  – Disk server, where MVS/ESA* disks are used like NetWare** disks.

- Central data distribution, where data can be copied between MVS/ESA* and the server station.

- Storage Server
  - ADSTAR* Distributed Storage Manager (ADSM* 5648-020) allows an MVS system to act as a file backup and archive server for LAN file server and workstations. This product operates as a server for OS/2, DOS, AIX/RISC, SUN** OS, Apple Macintosh**, Windows**, Novell**, DEC** ULTRIX** and SCO** Open Desktop** software platforms.

## 6.5.5 Application Development

TSO and ISPF have been the traditional application development tools under MVS, providing code entry, compile, and debug facilities.

The current IBM strategy for application development and maintenance is AD/Cycle*.

## 6.5.6 System Management

SystemView* is the IBM systems management strategy for managing, planning, coordinating, and operating open, heterogeneous distributed systems. SystemView* includes a SystemView* structure and SystemView* conforming products. The SystemView* structure is designed to provide a consistent end-user interface, shared data, enhanced automation, and increased integration among systems management products. The structure embraces selected open standards and architectures and defines conformance criteria for the SystemView* products. The SystemView* conforming products are, and will be, developed by IBM and non-IBM vendors.

### 6.5.6.1 Performance Management and Accounting

The system management facility (SMF) component of MVS/ESA* collects information that the installation can use to account for the system resources utilized by various type of workloads processed by the system. The resources considered are processor time, I/O data transferred, and central and expanded storage utilized.

MVS/ESA* and the ES/9000* channel subsystem provide information on specific aspects of system performance and physical resource utilization. On installation request and specification, the Resource Measurement Facility (RMF, V4 5685-029, V5 5655-084) collects and reports on this data which can be used for performance measurement, tuning, and capacity planning. RMF, in addition, provides real-time display of system performance and system resources usage, and information about the flow of the workload through the system.

To reduce the complexity of performance management in a S/390 Parallel Sysplex environment, RMF V5 introduces the concept of single system image for performance management reporting on sysplex wide data. It also supports the coupling facility, the Workload Manager of MVS/ESA* V5, and provides information for CICS* and IMS transaction monitors.

Monitors are available for performance data, tuning, accounting, and capacity planning for MVS/ESA* transaction monitors, such as IMSPARS (5798-CQP), CICSPARS (5796-AHJ), DB2* Performance Monitor (5665-354), and NetView Performance Monitor (5665-333).

The Enterprise Performance Data Manager (EPDM) (5695-101) or the Service Level Reporter (SLR) (5665-397) acts as a reporting tool for system management. Various system logs and data sources from the various monitors are collected and summarized according to user specifications. EPDM supports the S/390 parallel sysplex and the Workload Manager of MVS/ESA* V5, has a capacity planner feature, and may act as a centralized reporter for distributed AS/400* and RISC/6000*

Other vendor's products, such as Omegamon**, also operate in the area of performance management and monitoring.

#### 6.5.6.2 Operations

NetView* for MVS/ESA* (5685-111) provides the basis for management of network and systems operations from a central or remote site. For network operations, NetView, together with TCP/IP and OSI, can manage SNA and non-SNA components and heterogeneous LANs.

Automated Operation Control/MVS (AOC/MVS) (5685-151) runs as a NetView application and automates console operations. AOC/MVS monitors messages received from MVS, its subsystems, and various related products, and initiates pre-defined procedures specified for the various messages. AOC/MVS functions are complemented by a family of automated operation offering products, such as CICSAO, IMSAO, OPCAO.

AOC/MVS also provides enterprise resource monitoring through a PS/2* graphical interface where the user can include a S/390 parallel sysplex. Where appropriate, the resources are monitored at the sysplex level, and the graphical interface allows an operator to transfer to other system management products like RMF, SDSF, TSCF, NetView*. AOC/MVS also eases the task of propagating automation policy to multiple systems.

Operation Planning and Control/ESA* (OPC/ESA*, 5695-007) allows automation, planning, and control of the processing of the batch workload. This includes:

- Job submission
- Job dependency control
- Job recovery
- Dynamic re-planning
- Graphic display of networks of job dependencies and workload monitoring
- Remote site control through an ACF/VTAM application program
- Automatic restart of OPC/ESA on another MVS/ESA* within a Sysplex, in case of a system failure.

Other vendor's products, like CA** 7, operate in the area of operations control.

System Display and Search Facility (SDSF) (5665-488) allows monitoring and controlling the operation of an MVS/JES2 system. It supports th increased number of systems in a S/390 parallel sysplex and the MVS Workload Manager component.

The ESCON Manager (5688-008) simplifies the configuration management by collecting, unifying and presenting I/O configuration information in a graphical

form. It allows, in a S/390 parallel sysplex environment a multi-system vary capability for attached devices.

Target System Control Facility (TSCF, 5688-139) provides facilities for S/390* and Parallel Transaction Server 9672 remote operations. It extends NetView* systems operations and automation support to control and monitor multiple target systems.

#### 6.5.6.3 Availability and Integrity

Continuous availability is a combination of high availability and continuous operations. High availability is the elimination of unplanned outages and mainly concentrates on fault tolerance and recovery. Continuous operation is the elimination of planned outages and mainly concentrates on non-disruptive change and maintenance. Both are considered in the design of MVS/ESA* and its related hardware components.

MVS/ESA* manages hardware errors by retrying the operation or by fencing the damaged component. This includes processor storage, I/O channels, control units, I/O devices, and processors in a multiprocessor system, MVS images and the coupling facility in a S/390 parallel sysplex. In a multiprocessor system MVS/ESA* can move the work being done on a failing processor to the remaining processors.

The operator can also use system commands to fence some discrete hardware components (device, control unit, channel path, processor storage, processor), and allow hardware maintenance concurrent with system operations.

With the ESCON* architecture, it is possible to modify the physical configuration without interrupting the operations through the hardware configuration definition (HCD) function.

MVS/ESA* has software error recovery logic in every component of the operating system. In the event of an error, recovery is attempted in such a manner that the recovery processes are transparent to the user, or if this is not possible, that the damage is limited to the minimum.

The availability of data is a combination of attributes of the hardware, the operating system, the transaction manager, and the applications.

MVS/ESA* subsystems CICS*, IMS, DB2*, and IMS/DB employ 2-phase commit schemes to insure transactions and data integrity across distributed systems. These subsystems have journaling logs to record all changes. This enables recovery of jobs, transactions, and data if an unrecoverable DASD error or system failure occurs.

The dual copy capability (disk mirroring) possible with the advanced functions of the 3990 Model 3 and 6 control units enable the system to operate after a disk failure occurs, and concurrent copy can be used to backup databases without stopping the applications. In addition, the 3990 Model 6 control unit provides a remote copy extended function for disaster recovery purposes. Two implementations are available:

- Extended Remote Copy (XRC) which is an ESCON-distance asynchronous volume shadowing.
- Peer-to-Peer Remote Copy (PPRC) which is an ESCON-distance synchronous volume copying.

S/390 parallel sysplex provides the base for developing non-stop, shared data applications and servers. S/390 parallel sysplex also enhances continuous operations through the ability of the subsystems and applications to redirect workload from one system to another in case of a failure.

### 6.5.7 Selected APIs, Protocols, and Facilities

Table 8 summarizes selected protocols, facilities, and APIs available with MVS/ESA* and mentioned in this section.

*Table 8. MVS/ESA* Selected APIs, Protocols, and Facilities*

| | **MVS/ESA*** |
|---|---|
| **Berkeley** Sockets** | YES |
| **POSIX**** | YES |
| **CPI-C** | YES |
| **APPC** | YES |
| **APPN** | YES |
| **NCS**** | YES |
| **NQS**** | YES |
| **OSF/DCE**(1)** | ANN |
| **SQL** | YES |
| **DRDA-RUOW** | YES |
| **NFS**** | YES |
| **FTP** | YES |
| **FTAM** | YES |
| **OSF/Motif**** | YES |
| **X-windows**** | YES |
| **NetWare** server** | YES |
| **LAN & workstation server** | YES |
| **SMTP** | YES |
| **NetView*** | YES |
| **SNMP** | YES |

**Note:**
ANN (announced)
(1) base services

## 6.6 OS/2*

OS/2* V 2.1 (5604-467) is the IBM operating system for the personal computer industry in general, and in particular, for the IBM PS/2*, PS/1*, PS/ValuePoint*, and ThinkPad* families of processors. It exploits the 32-bit feature of the 80386/80486 microprocessors family designed by Intel**, and provides the ability to run existing 16/32-bit OS/2*, DOS, and Windows** applications concurrently.

OS/2* V 2.1 results from the evolution of the 16-bit OS/2* V 1 operating system designed in 1987.

Figure 62 shows the elements of the OS/2* V 2.1 based software platform.

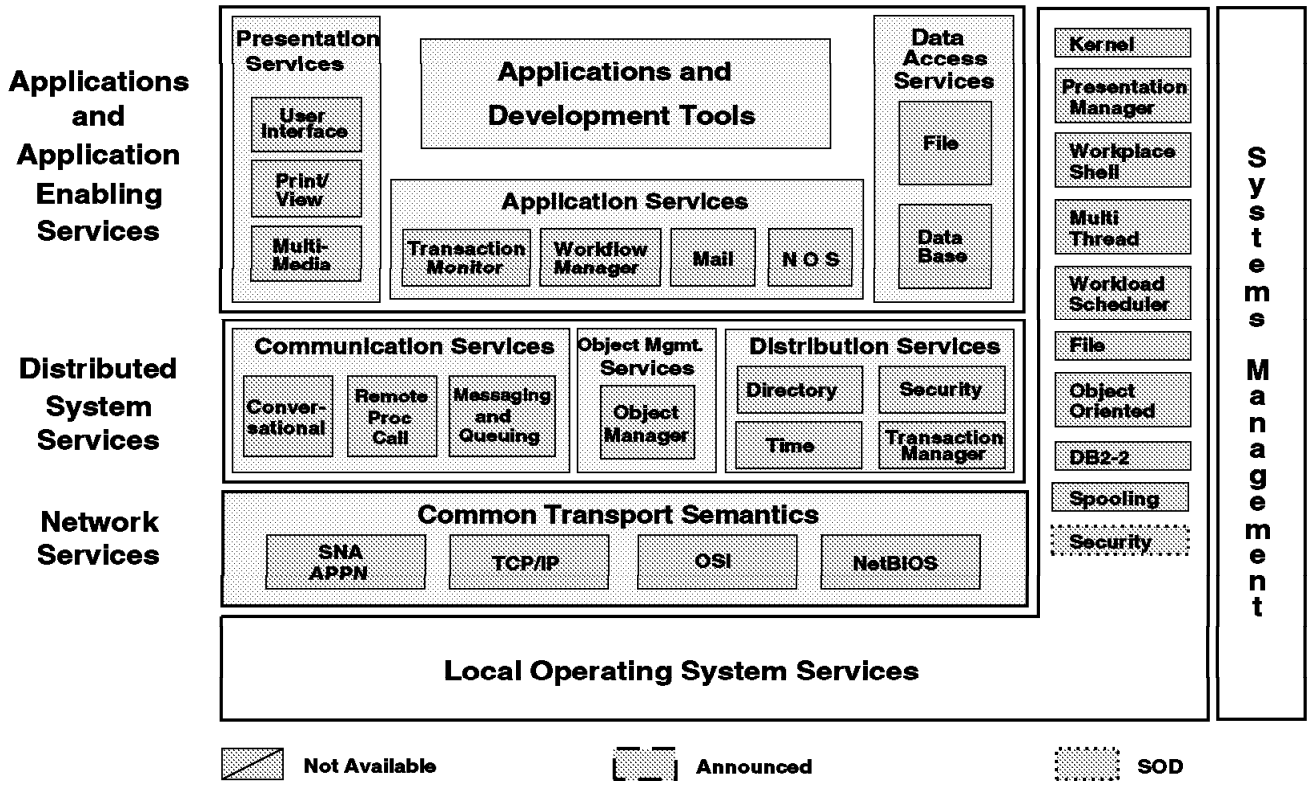


*Figure 62. Elements of the OS/2* V 2.1 Software Platform*

### 6.6.1 Local Operating System Services

OS/2 is a powerful, 32-bit multitasking, single-user operating system, designed and optimized for IBM PCs and compatible PCs which are based on the Intel x86 processor family. OS/2 includes capabilities for running DOS, Windows**, and OS/2 applications on a wide range of PC hardware, in an efficient and reliable manner.

The high-level structure of OS/2 is shown in Figure 63 on page 184.

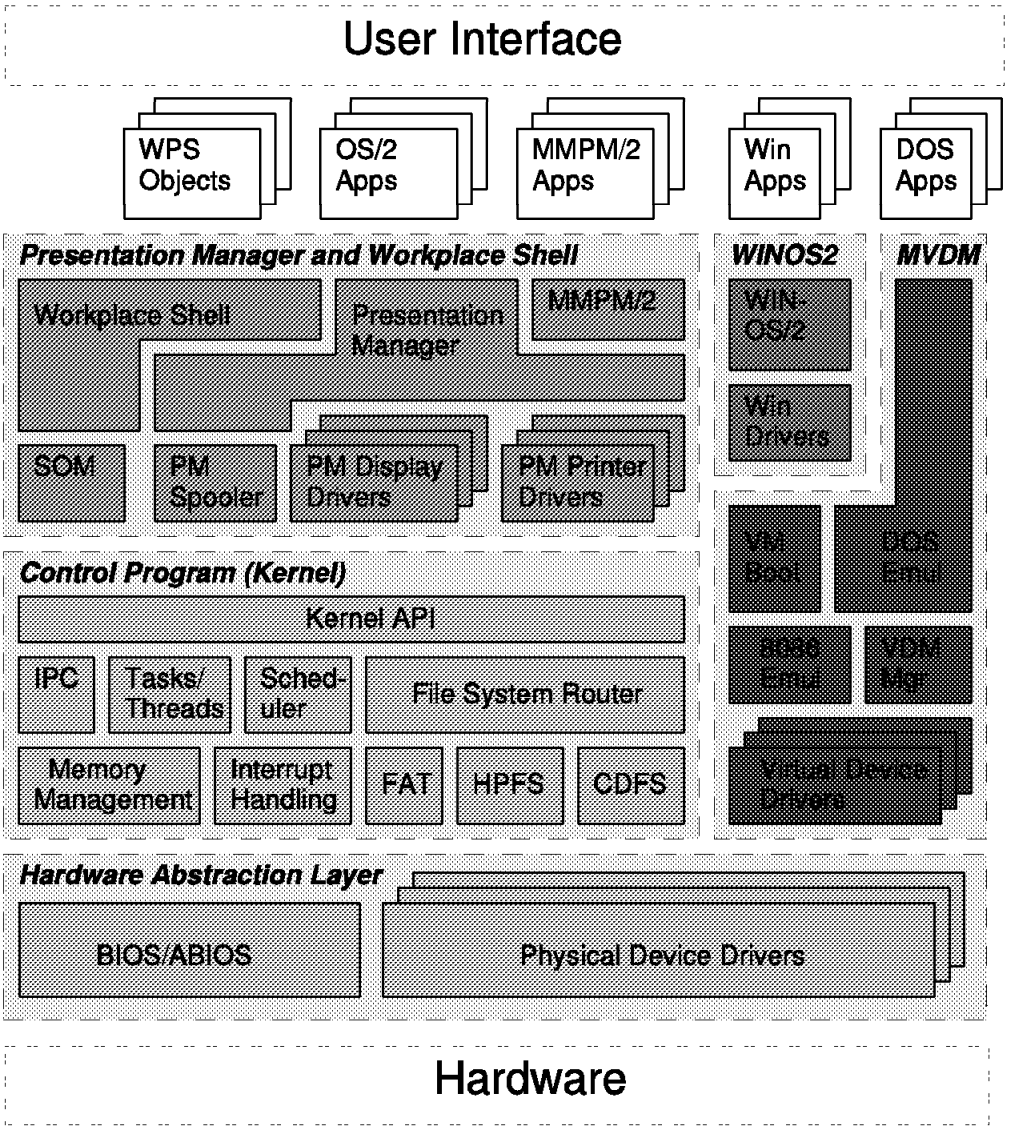


*Figure 63. OS/2 Structure*

The major components of OS/2 are:

- A hardware abstraction layer, which insulates OS/2 from specific hardware architecture details and provides an interface to specific devices.
- The control program, also called the kernel, which provides the fundamental operating system facilities.
- Presentation manager (PM) and the workplace shell, which provide the graphical user interface (GUI) and object-oriented desktop, and support for running OS/2 PM applications.
- Multiple virtual DOS machines (MVDM), which provide the support for running DOS applications.

- WIN-OS/2, which provides the support within MVDMs for running Windows 3.1 applications.

Components and subcomponents of OS/2 consist of executable programs and dynamic link libraries (DLLs), which are loaded when called by other programs. The DLL concept is very powerful, and major components of the system are implemented as DLLs.

Components of OS/2 itself can run at one of three protection levels:

- Ring 0, also called kernel mode, running the fundamental OS/2 subsystems
- Ring 2, used mainly for I/O operations, such as display drivers
- Ring 3, where user applications run, as well as many of the OS/2 subsystems.

Programs and data in the lower rings are protected by hardware from being corrupted by programs running in the higher rings.

#### 6.6.1.1 Hardware Abstraction Layer

Although OS/2 only runs on PCs using the Intel x86 family of processors, there are many details of the PC hardware which vary between different IBM PCs and compatible PCs.

OS/2 is insulated from the PC hardware architecture and the hardware devices (such as disks and video adapters) by a combination of the BIOS (or ABIOS) and the physical device drivers.

The BIOS, is a layer of firmware provided in ROM on every PC, provides the lowest level of system software, and insulates the operating system from the hardware. The BIOS is designed for use with the DOS operating system.

ABIOS is a similar layer of firmware designed for use with multitasking operating systems such as OS/2. ABIOS is used in IBM PS/2s based on the micro channel bus, and is implemented either in ROM or is loaded from disk into memory.

Physical device drivers provide installable software modules specific to each hardware device. Physical device drivers provide a standard interface to OS/2, and to higher level device drivers (such as display or printer drivers). Devices such as the keyboard, display, disks, and printer ports all have corresponding physical device drivers. All but the most basic physical device drivers are specified in the CONFIG.SYS file and are loaded at boot time.

#### 6.6.1.2 Control Program(Kernel)

The control program is an advanced 32-bit multitasking kernel for the operating system, providing a range of fundamental facilities.

Protected virtual memory is provided, using paged memory techniques and based on the hardware capabilities of the Intel x86 processors. Pages are loaded on demand and swapped out to an intermediate file on disk. Protection is provided between memory of different tasks using virtual memory, and this is supported by hardware. Each process can currently use up to 512MB of memory within a flat memory address space of 4GB.

Both processes and threads are supported. Processes have resources such as memory and open files associated with them, and can contain one or more

threads. Threads are executing programs, and can share memory and other resources within the same process.

Processes are loaded into memory by the loader, and the scheduler then determines which thread should run next based on a priority scheme. Thread priority can also be dynamically varied. Threads can be blocked when awaiting I/O, or can be interrupted when their timeslice has expired. OS/2 thus provides preemptive multitasking. The dispatcher runs the next ready thread, based on the decisions made by the scheduler (see section 6.6.1.7, “Workload Scheduler” on page 189).

External interrupts, such as from hardware devices, can interrupt the currently running thread. Interrupts are either handled directly or are transferred to physical device drivers.

A range of interprocess communication capabilities are provided to enable processes to interoperate and synchronize with each other. These include shared memory, pipes, queues, and semaphores.

OS/2 includes support for a number of file systems. Built-in support is provided for the DOS-compatible FAT file system. Installable file systems can also be loaded to provide support for additional file systems, such as the high-performance HPFS, and the CDFS for CD-ROMs (see section 6.6.1.10, “File” on page 189).

The system object module (SOM) provides language-neutral support for interaction between objects and classes, thus providing the basis for object-oriented subsystems (such as the Workplace Shell) to be built in OS/2 (see sections 6.6.1.11, “Object Oriented” on page 190 6.6.1.3, “Presentation Manager and the Workplace Shell”).

OS/2 provides an OS/2 command line which can be used either full-screen or in a window. The OS/2 command line provides a superset of the facilities provided by the DOS command line, and supports a primitive batch language. The REXX procedures language is also supported by OS/2 and provides more powerful facilities for both batch and interactive programs.

### 6.6.1.3 Presentation Manager and the Workplace Shell

OS/2′s native graphical user interface is provided by presentation manager (PM), and the workplace shell provides an object-oriented desktop on top of this.

Presentation manager is an advanced GUI, consisting of a set of DLLs, and PM display drivers and printer drivers.

PM itself has three major subcomponents. PMWIN is responsible for creating, maintaining and destroying windows on the PM desktop. PMGPI provides the API interface enabling applications to use the graphics environment. PMGRE is the graphics engine which is at the heart of PM.

PMGRE interacts with the PM display drivers and printer drivers, and also with an intelligent font interface such as Adobe** type manager. In addition, interfaces are provided for applications to exchange data using the clipboard and dynamic data exchange, and for PM applications to share access to the display with other applications, such as Windows** applications.

The PM display driver takes the output from PMGRE, and displays the information on the screen, usually interacting with the physical device driver. PM display drivers implement a wide set of functions for a specific device, such as drawing a character or a circle. Because of the differences between devices, there will normally be one PM display driver for each screen type (such as VGA, XGA, or SVGA).

The PM printer driver provides the equivalent capability for printers. Each printer driver can normally support a wide range of different printers, such as Postscript printers.

The PM spooler manages the sharing of printers between applications, spooling the printer output via disk (see section 6.6.1.8, “Spooling” on page 189).

On top of PM, the workplace shell provides an object-oriented user interface (OOUI) and desktop, which is centered around objects, such as data files, folders, and printers. The workplace shell is built using object-oriented techniques based on SOM and PM. The SAA common user access (CUA91) specification was used as the basis for the workplace shell.

A number of workplace shell objects and utilities are provided along with the desktop, enabling the user to easily change colors and fonts, or create new objects, using drag-and-drop and other advanced capabilities of the user interface.

OS/2 also includes an integrated multimedia subsystem, MMPM/2. MMPM/2 provides the advanced support needed to provide real-time digital audio and video, including software motion video (see section 6.6.4.1, “Presentation Services” on page 193).

### 6.6.1.4 Multiple Virtual DOS Machines

Support for DOS applications is provided by the multiple virtual DOS machines (MVDM) component of OS/2. MVDM exploits the hardware support of Intel 386 and higher processors for running individual 8086 virtual machines, protected by hardware from each other and from the main operating system. This enables support for multiple concurrent DOS applications to be provided in a first-class way under OS/2.

The 8086 emulation builds on the hardware support to provide an 8086 environment, and traps interrupts and any attempts to access the real hardware, reflecting these to device drivers. Either emulated DOS or a real copy of DOS can be run inside this 8086 environment.

DOS emulation then builds on this layer to provide an emulated DOS 5 environment for each virtual machine. Most of the DOS function is implemented outside the virtual 8086 machine, thus saving precious memory space. DOS emulation also emulates the EMS, XMS and DPM memory extenders using OS/2 memory. Although each DOS machine is encapsulated, there are a few ″holes in the wall″, such as named pipes and virtual device drivers, which enable DOS applications to communicate with OS/2 applications and subsystems. DOS applications can run either full-screen or windowed (except for some graphics applications which can only run full-screen). When run in a DOS window, capabilities, such as the clipboard, are available to the user.

Virtual device drivers emulate the interface provided to DOS applications by DOS device drivers by interacting with the physical device drivers. This enables

hardware devices to be shared between DOS and OS/2 applications, and also reduces the memory needed within each DOS machine. It is also possible to load DOS device drivers directly in a DOS machine; the DOS machine would then exclusively own the physical hardware device.

Virtual machine boot (VM boot) enables a real copy of DOS to be loaded and run in a virtual 8086 machine. This may be used if the application requires specific DOS facilities not provided by DOS emulation. VM boot can load DOS from diskette, from a partition on the fixed disk, or from a diskette image on the fixed disk.

The VDM manager handles the creation, deletion and management of the virtual DOS and 8086 machines.

MVDM also provides a DOS command line, which can be used either full-screen or in a window. The DOS command line runs a modified version of the DOS COMMAND.COM command interpreter in a virtual DOS machine. It provides the same facilities as the real DOS command line, and supports a primitive batch language.

#### 6.6.1.5 WIN-OS/2

Support for Windows 3.1 applications is provided by the WIN-OS/2 component. WIN-OS/2 is a version of Windows 3.1 which has been modified to enable it to run under MVDM and to share the display with OS/2 applications. WIN-OS/2 uses the underlying MVDM facilities.

WIN-OS/2 consists of three major subcomponents, along with WIN-OS/2 display drivers and Windows printer drivers. The WIN-OS/2 subcomponents handle kernel functions, such as memory management, the windowing and user interface, and the graphics. WIN-OS/2 can use either the ATM or the true type font managers. Support is also provided for applications to exchange data using the clipboard and DDE, and for compound documents using OLE. WIN-OS/2 cooperates with PM to share access to the display between applications.

Both full-screen and seamless WIN-OS/2 display drivers are used. These are based on the Windows display driver, with additions to coordinate access to the screen with PM.

WIN-OS/2 uses windows printer drivers, and also provides spooling facilities, either through the PM spooler or using the WIN-OS/2 spooler (see section 6.6.1.8, “Spooling” on page 189).

#### 6.6.1.6 System Software Extensions

The following services are provided by system software extensions to OS/2*:

- Pen for OS/2* extends OS/2* for pen-based systems, including interpretation of gestures, and handwriting recognition. Pen for OS/2 can be used to extend pen-unaware applications, as well as providing API extensions for writing new pen-aware applications.
- Video IN provides the capability to record software motion video clips, using appropriate hardware. These software motion video clips can then be replayed on OS/2* systems using the built-in MMPM/2 facilities (see section 6.6.1.11, “Object Oriented” on page 190).

#### 6.6.1.7 Workload Scheduler

OS/2* V.2 workload may range from interactive processing for personal productivity, to multiple background DOS and Windows** applications. Subsystems, such as the communication manager, the database manager, the LAN server, and CICS* can also operate concurrently.

In OS/2* V.2, multiprogramming allows concurrent execution of multiple applications, and multitasking distributes processor time among multiple programs, providing in turn for each one a defined (short) period of processor time. OS/2* V.2 also provides a preemptive time-slicing scheduler and a multi-level priority scheme, with dynamic priority variation and round-robin dispatching within priority levels.

In OS/2*, the thread is the basic unit of execution and concurrency, and CPU allocation and applications can operate with multiple threads, allowing multiple parts (threads) of the same program to execute simultaneously. A thread can be assigned to one of four priority classes, and within each class the scheduler recognizes 32 priority levels. Priority classes are defined as:

- Time-critical for real-time applications and communications
- Fixed-high for good responsiveness
- Regular for normal execution
- Idle-time for very low priority.

#### 6.6.1.8 Spooling

The Simultaneous Peripheral Operations On-Line (Spool) facility is provided by OS/2* V.2 for the DOS, Windows**, and workstation user for the temporary storage of output data for later printing.

#### 6.6.1.9 Database

IBM DATABASE 2 for OS/2* (DB2/2 5622-044) is the relational database management system. It is a member of the IBM relational database family, and can be used in a single-user workstation and in a client/server LAN environment. It includes functions previously provided by Extended Services for OS/2* (5621-213). DB2/2 provides SQL capabilities and roll-forward recovery for database recovery after a system failure. It includes functions for application portability and DB2* compatibility. It has graphical tools to configure and manage the database and its directories. For distributed relational database access, see section 6.6.4.3, “Data Access Services” on page 195.

#### 6.6.1.10 File

The following file systems are available, and represent a hierarchy of files on physical disks:

- Enhanced file allocation table (FAT) for compatibility with DOS files.
- High performance file system (HPFS), which provides support for disk sizes up to 2GB, caching for reading and writing, and long file names. The file system exploits the enhanced performance feature of the small computer system interface (SCSI) disk drives. The HPFS make it possible for OS/2* to support large databases on personal computers.
- CD-ROM file system (CDFS) for files residing on compact disks.

OS/2* V.2 provides locking and sharing facilities to allow concurrent file access in a multitasking environment.

OS/2* V.2 introduces an important feature to the file systems, namely the extended attributes. This feature allows each file to have up to 64KB of file related descriptive information, which can be used to create object-oriented file systems.

#### 6.6.1.11 Object Oriented

The system object model (SOM) is used to build the workplace shell, and enables developers to integrate their application fully into the workplace shell using object-oriented technology.

The SOMObjects Toolkit, and associated runtime products, extend the object-oriented capabilities of SOM. SOM itself supports interaction between objects and classes within the same process. SOM II extends this to cross-process interaction, and DSOM extends this further to cross-machine interaction (see section 6.6.3.2, “Object Management Services” on page 193). A SOM compiler is available for C programming.

#### 6.6.1.12 Security

OS/2* V.2 is a single-user operating system; therefore, there is less need for user identification and authentication to grant access to system resources. OS/2* offers user profile functions that can be used to limit access for individual users.

IBM has stated the intention to use the PS/2* systems and the OS/2* operating system as the platform to provide a personal computing environment that satisfies the requirements of the C2 level of security and integrity as defined by the United States Department of Defense (DoD).

When OS/2* V.2 participates in distributed processing, it provides security and auditability services, such as userid and password management. The utilization of those services is under application control.

### 6.6.2 Network Services

Network services for OS/2* V.2 are implemented by:

- Communications Manager/2 (CM/2 5622-078)
- Network Transport Services/2 (5622-022)
- TCP/IP for OS/2* (5622-086)
- LAN Server for OS/2* (5621-253)
- NetWare** Requester for OS/2*
- LAN Support Program (5621-300).

These products allow OS/2* to handle local and remote networking communication with PC′s running Novell** or Microsoft** network protocols, and systems running on other software platforms.

- SNA/APPN

  CM/2 and CM CS/2 are the separation and further enhancement of the communication manager component included in the IBM OS/2* Extended Edition and Extended Services for OS/2*. The CM/2 manager supports SNA and NetBIOS communications. Remote connectivity is provided for SNA, SDLC, and X.25 protocols. The available APIs are:

- CPI-C APIs are available for Advanced Program-to-Program Communications (APPC), using SNA LU 6.2 protocol.

The SNA Advanced Peer-to-Peer Networking (APPN) technology is also supported by the communication manager.

The SNA LU 6.2 protocols provided by the OS/2 communication manager are used by the IBM Entry LAN to LAN Wide Area Network Program (LTLW 5622-067) to interconnect remote local area networks (LANs) across wide area networks (WANs).

CM/2 also provides emulation capabilities for IBM 3270, IBM 3101, and DEC** VT 100 terminals, and file transfer facilities to send or receive files across emulator sessions and OS/2* or DOS windows.

The SNA gateway is part of the communication manager and is a non-dedicated server that provides its clients a shared link to a S/390 host.

- NetBIOS

  NetBIOS is a protocol for LAN based program-to-program communications. It runs on Ethernet** and Token-Ring LANs and interfaces with TCP/IP and OSI. It provides application programming interfaces for LAN applications on the LAN network.

  The IBM Network Transport Services/2 (NTS/2) provide communication programming interfaces necessary for LAN enablement in those environments where the OS/2* LAN Server is not required.

- TCP/IP

  TCP/IP for OS/2* provides the following services and APIs:

  - Terminal passthrough (TELNET) allows a user client to remotely logon to another computer with TCP/IP TELNET server function.
  - Simple Mail Transfer Protocol (SMTP) allows an end-to-end electronic mail exchange.
  - Simple Network Management Protocol (SNMP) allows OS/2* to be both a managing and a managed system.
  - File Transfer Protocol (FTP) (see section 6.6.4.3, “Data Access Services” on page 195).
  - Network File System (NFS**). (see section 6.6.4.3, “Data Access Services” on page 195).
  - LPR/LPD remote printer support (see section 6.6.4.3, “Data Access Services” on page 195).
  - X-windows** server for client applications. The presentation manager is used as the window manager.
  - Remote Procedure Call (RPC) library that allows program-to-program communication for distributed applications with high level program calls.
  - Network Computing System (NCS**) allows high-level calls to procedures to be executed on other systems on the network.
  - Berkeley** socket library for distributed applications communication.

- OSI

  The OSI Communications Subsystem for OS/2* (5601-124) with OSI File Services/2 (5601-211) enables OS/2* users to transfer and manage files between software platforms where a compatible set of OSI protocols exist.

The OSI functions and APIs are:

- OSI File Services (see section 6.6.4.3, “Data Access Services” on page 195)
- The Remote Programming Interface (RPI) for development of distributed applications using COBOL and C languages.

The LAN Distributed Platform for OS/2* (LANDP/2 5622-108) provides a client/server distributed programming capability well suited to develop distributed services and applications in an heterogeneous LAN environment. The LANDP platform allows integration of heterogeneous systems (DOS, Windows, OS/2*, OS/400* and AIX/6000*) and heterogeneous communication environments (NetBIOS, TCP/IP, SNA, and X.25). In addition to the programming services, LANDP also provides several ready-to-use functions, such as 3270 emulation, shared file, electronic journal, store and forward services and management services.

Interprocessor communication and data transfer across a network are also available with Named Pipes. They provide a file-like programming interface for two-way exchange of data, similar to the API normally used for sequential file processing.

Multi-Protocol Transport Networking (MPTN) architecture supports CTS (Common Transport Semantics) in IBM* Networking Blueprint. CTS provides a common view of networking protocols, and it makes applications independent of the underlying networking protocols. CTS is implemented in OS/2 through AnyNet/2 (5622-178, 260). AnyNet/2 allows APPC applications to run over the TCP/IP network and TCP/IP Sockets to run over an SNA network.

Table 9 summarizes the OS/2* networking capabilities.

*Table 9. OS/2* Networking Capabilities*

| | **Ethernet**** | **TokenRing** | **FDDI** | **X.25** | **SDLC** |
|---|---|---|---|---|---|
| **SNA** | YES | YES | - | YES | YES |
| **TCP/IP** | YES | YES | - | YES | - |
| **OSI** | - | - | - | YES | - |
| **NetBIOS** | YES | YES | YES(1) | - | - |

**Note:** (1) Through LAN Support Program

## 6.6.3 Distributed System Services

### 6.6.3.1 Communication Services

The communication services capabilities within OS/2* are provided for:

- Conversational

  CPI-C APIs for conversation, according to the SNA LU 6.2 protocols and APPC programming interfaces.

- Remote Procedure Call

  DCE Client for OS/2 (5696-692) provides RPC that allows programs to work across heterogeneous systems by masking the differences between data representations and the network details of different software platforms.

- Messaging and Queuing

The product ezBRIDGE** Transarc** on OS/2 (5787-EDD) provides IBM′s Message Queue Interface (MQI) to distributed applications running on the OS/2* platform.

#### 6.6.3.2 Object Management Services

- Object Manager

  System Object Module (SOM) technology and Distributed SOM (DSOM) technology is available on the OS/2* platform with the IBM SOMobjects Developer Toolkit and the associated runtime products (Workstation Enablers and Workgroup Enablers) (5604-479, 482, 484). SOMobjects comply with the Object Management Group′s (OMG**) Common Object Request Broker Architecture (CORBA).

#### 6.6.3.3 Distribution Services

- Directory

  DCE Client for OS/2* (5696-692) for directory services. DCE for OS/2* offers a Software Developer′s Kit (SDK, 5696-657) for DCE Cell Directory Services.

- Security

  DCE Client for OS/2* (5696-692) for security services. DCE for OS/2* offers a Software Developer′s Kit (SDK, 5696-657) for DCE Security Services.

- Time

  DCE Client for OS/2* (5696-692) for time services.

- Transaction Manager

  The CICS* for OS/2* transaction monitor has a transaction manager function to ensure transaction and data integrity across distributed systems. Encina transaction manager from Transarc**, an IBM Business Partner, will also be available for OS/2.

### 6.6.4 Application Enabling Services

The application enabling services available with OS/2* are described in this section.

#### 6.6.4.1 Presentation Services

- User Interface

  The OS/2* Workplace Shell* provides a user interface, based on the presentation manager, that was developed using the IBM System Object Model (SOM) object-oriented technology. The Workplace Shell* gives the user a single interface to manage multiple objects, including devices such as printers, disk drives, files, and programs. Applications can be integrated with the Workplace Shell* and represented on the desktop with icons.

  The workplace shell provides the cut-and-paste service for graphic and text information between Presentation Manager, DOS, and Windows** applications. The workplace shell is integrated with the LAN environment. It provides an iconic view of LAN-based resources, which can be manipulated through drag and drop techniques like any other desktop resource.

- Print/View

  TCP/IP for OS/2* delivers the Line Printer Client/Line Printer Daemon (LPR/LPD) functions that provides client/server support for distributed printing. The LPR sends data to be printed to the LPD on a specified server

platform for a specified printer. The LPD server on the target software platform provides users access to the attached printers.

LAN Server for OS/2* and NetWare** for OS/2* allow sharing of print resources among client workstations connected to a LAN (see section 6.6.4.3, “Data Access Services” on page 195).

The IBM Print Services Facility/2 (PSF/2 5601-298) provides the OS/2 platform with the printing services as required by the IBM′s Advanced Function Printing* (AFP*) model and architectures. PSF/2 can be used as a printer driver on a stand-alone system, as a print server on a local area network environment, and as a print server on a distributed print network.

- Multi-Media

  OS/2* Multi-Media Presentation Manager/2*, integrated in the OS2* V 2.1, provides a common programming interface for multi-media applications. MMPM/2 includes support for multi-media logical devices, such as audio adapters, CD-ROM drivers, and video disc players.

  Multimedia applications can be configured as a stand-alone personal workstation or as a workgroup system consisting of workstations interconnected to a LAN server or to a host system.

### 6.6.4.2 Application Services

- Transaction Monitor

  CICS OS/2* V.2 (5648-036) provides CICS* transaction management to multiple users of LAN attached programmable workstations. ASCII terminals can also be connected to CICS OS/2* using the Real Time Interface Co-processor card (ARTIC) and the Programmable Network Access (PNA) software. CICS OS/2* can operate as a server for CICS OS/2* workstation clients, as CICS OS/2* standalone, or as a client, and also supports cooperative processing with other CICS* family systems.

  CICS* applications can communicate with CICS* applications on other software platforms through several distinct mechanisms. This capability exists across the family of CICS* monitors for the IBM software platforms (AIX/6000, OS/2*, OS/400*, MVS/ESA*, VSE/ESA*). These mechanisms allow CICS* to route transactions to another CICS* for execution, allow for transparent access to certain remote CICS* resources, and allow CICS* to invoke a remote CICS* application.

  The CICS OS/2* APIs provided to develop distributed transactions are:

  – CICS* COBOL and C command level
  – Transaction Routing, Function Shipping, and Distributed Program Link (DPL)
  – CICS* implementation of LU 6.2 protocols and interface with CPI-C
  – External call interface (ECI) that provides an RPC-like communication between applications and is compatible with CICS* DPL.

  IMS Client Server/2* (V2 5622-113) allows an OS/2* application program to run an IMS transaction transparently to the both the OS/2* application and the host.

- WorkFlow Manager

FlowMark OS/2 (5621-290) provides a set of workflow management functions to document, control, and progressively improve business processing.

- Mail

  The Simple Mail Transfer Protocol (SMTP) application of TCP/IP provides an electronic mail protocol for transferring electronic mail messages. The interface with SMTP is the presentation manager application LaMail. The LaMail program allows users to view, create, edit, spell check, and send mail.

  Lotus cc:Mail** provides electronic mail exchange among LAN workstations running the OS/2* workplace shell, DOS, Windows**, Macintosh**, and UNIX**, with gateways to OfficeVision/VM*, OS/400, and DEC**.

#### 6.6.4.3 Data Access Services

- Relational

  IBM Database 2 for OS/2* (DB2/2 5622-044) is a relational database management system and a member of the IBM relational database family. It can be used in a single-user workstation and in a client/server LAN environment. It allows access to an OS/2* database servers from OS/2* DOS and Windows** database client workstations. DB2/2 provides SQL capabilities and roll-forward recovery for database recovery from a system failure. It includes functions for application portability and DB2* compatibility. It has graphical tools to configure and manage the database and its directories.

  DB2* for OS/2* and Distributed Database Connection Services for OS/2* (DDCS/2 5622-056) provide database access for decision support systems, as well as for online transaction processing applications that may reside on a LAN or in a host. DDCS/2 participates in the Distributed Relational Database Architecture (DRDA) with platforms that implement the remote unit of work (RUOW). This allows OS/2* users to access and update relational databases residing on other operating environments that use the same architecture. The IBM software platforms that use RUOW are: SQL/DS on VM/ESA*, OS/400*, DB2* on AIX/6000, MVS/ESA*, and OS/2*.

- File

  Services to access and manage distributed data are provided by the following:

  - TCP/IP File Transfer Protocol (FTP) allows transfer of files from one machine to another. FTP supports the transfer of both binary and ASCII files, and may use the HPFS file system and its long file names. An OS/2* TCP/IP node can be an FTP client, or server, or both. An FTP client includes a presentation manager front end for a graphical interface.
  - TCP/IP Network File System (NFS**) allows a user to have transparent access to a hierarchical file structure. It allows users to share files. Any NFS** client, including UNIX** platforms, can share files with an OS/2* platform.
  - OSI File Services/2 (5601-211) enables OS/2* users to transfer and manage files between software platforms.
  - NetWare** requester for OS/2* allows an OS/2* client to request services from Novell** LAN Servers, and also allows NetWare** clients to utilize the OS/2* LAN server, allowing shared disk files.

- LAN Server for OS/2* (5621-253) acts as a central hub for requests from client workstations in the LAN. The requests are routed from the client workstation to the server by a LAN requester program (included in the LAN Server product). The resources that can be shared on a LAN Server include disk directories, disk files, printers, serially attached devices, application programs, and network services. Access to those resources is completely transparent to the workstation client.
- The Distributed FileManager with Distributed Access Services (DAS is of ADSM* 5648-020) allows OS/2 V2-based applications to access distributed record data on target systems, as defined by the Distributed Data Management (DDM) Architecture. ADSM* is intended to provide distributed file access services in heterogeneous distributed environments. DFDSM* operates with SAA-based systems, non-SAA, and non-IBM platforms that support the DDM architecture. The DDM architecture defines the target system as the system where data resides, and the source system as the system where the application resides. DDM source system functions are implemented by the Distributed FileManager on the OS2 V2 platform. Target system support is currently provided on the following systems, among others: OS/400 platform, and MVS and VSE platforms through the CICS* DDM support.

- Storage

  ADSTAR Distributed Storage Manager (ADSM*) allows an MVS, VM, or AIX/6000 system to act as a file backup and archive server for LAN file servers and workstations. OS/2* can be a client of ADSM* (see section 6.5.4.3, “Data Access Services” on page 178).

#### 6.6.4.4 Network Operating System

OS/2* can provide the functions of a Network Operating System (NOS) with the LAN Server for OS/2*, which allows application networking. OS/2*, as a network operating system, provides applications with networking capabilities through its standard APIs and commands. The NOS is a typical implementation of a client/server concept, where a server acts as a central hub for requests from the clients. Network devices, programs, files, and printers can be centralized on a single server or distributed among multiple servers connected to a LAN. With Microsoft** LAN Manager, and the NetWare** requester for OS/2*, the client capability can be extended to Windows** and Novell** platforms.

### 6.6.5 Application Development

To assist in developing OS/2* applications, these programs are available:

- IBM OS/2* V.2 Tools for Application Development (5621-078, 5604-478) is designed to provide language-independent development tools (such as the linker and dialog box editor), detailed documentation, sample programs, and also the header files and libraries needed to compile and link applications.

  The OS/2 toolkit is also included on the Developer Connection CD, which is a quarterly subscription service CD enabling new releases of tools and beta versions of OS/2 products to reach developers quickly and easily. It also includes additional online documentation, such as the OS/2 redbooks. The Device Driver Source Kit for OS/2 is also provided on a subscription service CD, and is for developers of OS/2 device drivers. Along with documentation, it also includes sample device driver source code which can be used as a basis for developing device drivers for new hardware.

- IBM WorkFrame/2 (5621-302) provides a programmers′ visual workbench, into which language-specific compilers and editors can be plugged. Workframe/2 also helps to manage the project-based development of applications.
- IBM C set++ (5604-464, 465) provides all the language-specific tools needed to develop C and C++ applications. It includes a full 32-bit C and C++ compiler for OS/2. C is the standard programming language for the PC, and C++ is an object-oriented version of C, which is being increasingly widely used. Also included are interactive visual debuggers, class browsers, and execution trace analyzers.
- IBM C set++ libraries (5604-466) are provided to help developers build object-oriented applications faster. An example of the available class libraries is the User Interface Class library, which can be used for developing OS/2 and Workplace Shell applications in C++.
- VisualAge for OS/2* (5621-387) provides an interactive, visual tool for fast development of applications. VisualAge is based on the Smalltalk language.
- Multimedia Presentation Manager Toolkit/2 (5604-376) and Ultimedia* Builder/2 (5604-401) are available to assist in the development of multi-media applications.

## 6.6.6 System Management

SystemView* is the IBM systems management strategy for managing, planning, coordinating, and operating open, heterogeneous distributed systems. SystemView* includes a structure and the conforming products. The SystemView* structure is designed to provide a consistent end-user interface, shared data, enhanced automation, and increased integration among systems management products. The structure embraces selected open standards and architectures and defines conformance criteria for the SystemView* products. The SystemView* conforming products are, and will be, developed by IBM and non-IBM vendors.

### 6.6.6.1 Performance Management and Accounting

The System Performance Monitor/2 (SPM/2 5622-010) provides a powerful tool for monitoring the performance and usage of an OS/2 system, including memory and disk usage. SPM/2 also includes the ability to present this information visually. In addition to collecting performance data locally on a given system, SPM/2 allows data collection from OS/2* LAN servers and LAN requesters. It also has a distributed feature that allows remote monitoring for servers and requesters. SPM/2 conforms to SystemView* and is based on presentation manager.

### 6.6.6.2 Operations

The following are some of the programs that allow LAN management, and management of OS/2* workstations connected to a LAN:

- LAN NetView* family is a series of products that allow configuration, installation, monitoring, and management of LAN systems.
- NetView/PC* (5669-024) allows the implementation of the NetView* Service Point concept on the OS/2* software platform. It collects network management information, formats it into alerts, and forwards alerts to NetView* for centralized network management. Host connectivity is through any SNA link supported by the OS/2* communication manager.

- LAN Network Manager (5621-117) and LAN Station Manager (5621-103) provide heterogeneous LAN management capabilities for token-ring and PC network attached stations. LAN Network Manager conforms to SystemView* and uses graphical displays. LAN Network Manager cooperates with a centralized network management NetView.
- Distributed Console Access Facility (DCAF 5621-414) addresses the need for a personal computer-based, central site help desk function. It can be used to facilitate network management, network administration, and application assistance involving personal computer workstations distributed across SNA networks and on SNA-connected IBM Token-Ring LANs.

#### 6.6.6.3 Availability and Integrity

In a multitasking environment, a serious error occurring in one application must not be allowed to damage other applications that are running in the system. In case of an error, OS/2* terminates the application unless the application has requested to handle the error. Protection features of the processor architecture are utilized to protect applications from each other and to protect the operating system kernel from the applications.

### 6.6.7 Selected APIs, Protocols, and Facilities

Table 10 summarizes the protocols, facilities, and APIs available with OS/2*.

*Table 10. OS/2* Selected APIs, Protocols, and Facilities*

| | **OS2*** |
|---|---|
| **Berkeley** Sockets** | YES |
| **POSIX**** | - |
| **CPI-C** | YES |
| **APPC** | YES |
| **APPN** | YES |
| **NCS**** | YES |
| **NQS**** | - |
| **OSF/DCE**** | YES |
| **SQL** | YES |
| **DRDA-RUOW** | YES |
| **NFS**** | YES |
| **FTP** | YES |
| **FTAM** | YES |
| **OSF/Motif**** | YES |
| **X-windows**** | YES |
| **NetWare** server(1)** | YES |
| **LAN & workstation server(1)** | YES |
| **SMTP** | YES |
| **NetView*** | YES |
| **SNMP** | YES |
| **Note:** (1) Server for LAN attached workstations | |

## 6.7 OS/400*

OS/400* (5738-SS1) is the operating system of the Application System/400* (AS/400*).

The AS/400* system is a family of midrange data processing systems optimized for application solutions for commercial data processing, office, and communications environments.

Figure 64 shows the elements of the OS/400* software platform.

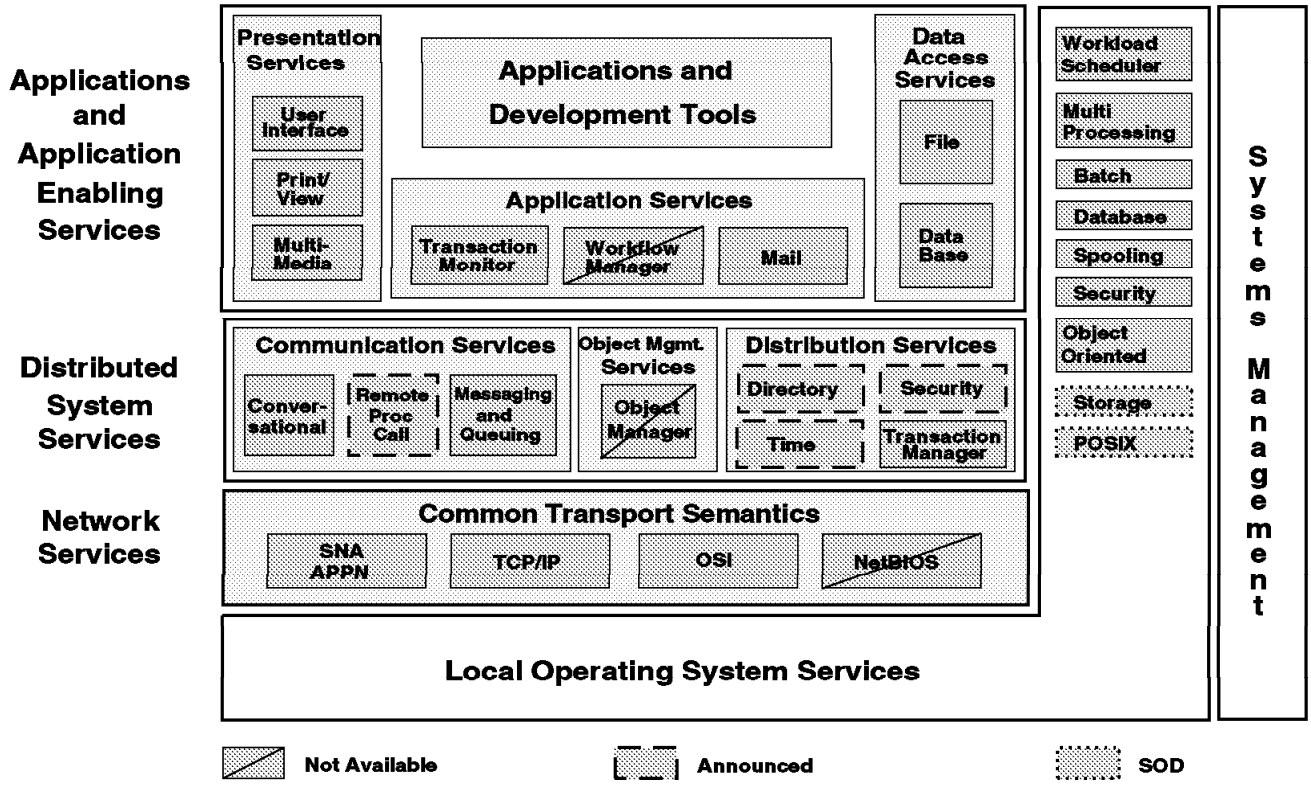


*Figure 64. Elements of the OS/400* Software Platform*

### 6.7.1 Local Operating System Services

A basic design concept of OS/400* is that the behavior of the system is defined by the software and not by the hardware. The main characteristics of the AS/400* system and of the OS/400* software platform are:

- Multi-layered machine interface
- Object-oriented architecture
- Single-level storage.

They have already been discussed in section 2.7, “IBM Application System/400* (AS/400*)” on page 24.

OS/400* is a multi-user operating system, and multiprocessing is also implemented.

The application environment can consist of OS/400* applications and applications migrated from the System/36 and the System/38 platforms. Applications that have a mixture of the above are also possible.

DASD sharing with other AS/400* systems is not supported.

IBM intends to include the following POSIX** standards in the OS/400* platform:

- 1003.1 (system interface)
- 1003.2 (shell and utilities)
- 1003.4 (real-time threads).

IBM intends to seek XPG4** Base Branding from X/OPEN for a future release of OS/400*.

#### 6.7.1.1 Workload Scheduler

OS/400* is a multi-user operating system designed for interactive workloads. No distinction is made between interactive workloads and other types of workloads, such as batch. Program management, initiation, and termination occur in exactly the same way for both batch and interactive programs.

OS/400* provides the system operator with tools to assign system resources to specific types of workloads (for example, batch), if necessary.

#### 6.7.1.2 Spooling

The Simultaneous Peripheral Operations On-Line (Spool) facility is provided by the OS/400* for the temporary storage of user related data, mainly output data for later printing. PC Support/400 (PC/400 5738-PC1) allows a personal computer to use the AS/400* system printers. Personal computer printer data can be transparently directed to the AS/400* spool files.

#### 6.7.1.3 Database

The OS/400* relational database support is integrated into the operating system, which also provides features for a high degree of data integrity. For user-friendly data access, the SQL/400* (5738-ST1) and Query/400 (5738-QU1) programs are available.

OS/400* databases can also be accessed using programming languages. Application programming interfaces are also provided for applications that require access to hierarchical data structures.

Compatibility is maintained with the S/36 data access interfaces for applications that have been migrated from the S/36 system environments.

#### 6.7.1.4 Security

There are many levels of security available with OS/400* to simplify the task of security management. System services based on password control are provided to control the access to system objects for read, update, add, and delete. Operating system integrity is protected by denying access to the inner layers of the machine-level interface.

Cryptographic Support/400 (5738-CR1) provides data protection by encryption. It allows encrypted data to be sent over the network, or to be stored in the OS/400* databases.

### 6.7.2 Network Services

OS/400* supports a wide range of connectivity options for the three main network protocols, SNA, TCP/IP, and OSI:

- SNA/APPN

  SNA communication protocols are provided as an integral part of the OS/400* operating system, including the LU 6.2 and APPC services.

  The SAA CPI-C interface is provided to assist in the development of distributed applications using the SNA conversation protocol.

  The SNA Advanced Peer-to-Peer Networking (APPN) and the Low Entry Networking (LEN) support are provided by OS/400*. Several applications are available to perform functions such as file and object transfer, electronic mail, document exchange, CallPath support, and remote logon.

- TCP/IP

  TCP/IP Connectivity Utilities/400 (5738-TC1) allows an OS/400* user to communicate with other systems over a TCP/IP network. The functions and applications provided are:

  – Terminal passthrough (TELNET) allows a user client to remotely logon to another computer with the TCP/IP TELNET server function.

  – Simple Mail Transfer Protocol (SMTP) allows an end-to-end electronic mail exchange.

  – File Transfer Protocol (FTP) (see section 6.7.4.3, “Data Access Services” on page 204).

  – Network File System (NFS**) (see section 6.7.4.3, “Data Access Services” on page 204).

  IBM intends to include the following functions in TCP/IP for OS/400*:

  – Simple Network Management Protocol (SNMP).

  – X-windows** and Motif** client facilities to allow programs to access a high resolution display connected to an X-windows** graphical workstation acting as a server.

  IBM Connection Program/400 (5798-RZB) provides the following functions to the TCP/IP UNIX** environment:

  – 5250 terminal emulation

  – OS/400 remote database access using SQL APIs

  – Remote print from OS/400 to workstation

  – Remote command from OS/400 to workstation.

- OSI

  OSI Communication Subsystem/400 (5738-OS1) enables OS/400* to communicate with other systems over an OSI network.

  The OSI functions and applications provided are:

  – OSI Message Services/400 (5738-MS1) for message handling according to the X.400 standard in a multi-vendor environment (see section 6.7.4.2, “Application Services” on page 203).

  – OSI File Services (5738-FS1) (see section 6.7.4.3, “Data Access Services” on page 204).

Multi-Protocol Transfer Networking (MPTN) architecture supports CTS (Common Transport Semantics) of IBM Networking Blueprint. IBM intends to provide support for MPTN for OS/400 to allow applications to use multiple communications protocols (such as APPC conversations and TCP/IP sockets) independently of the underlying transport protocol.

LAN Distributed Platform/400 (LANDP/400 5733-107) provides server capabilities to programmable workstations running LANDP/DOS and LANDP/OS2. The LAN Distributed Platform also provides a client/server distributed programming capability well suited to develop distributed services and applications in an heterogeneous LAN environment. The LANDP platform allows integration of heterogeneous systems (DOS, Windows, OS/2*, OS/400* and AIX/6000*) and heterogeneous communication environments (NETBIOS, TCP/IP, SNA, and X.25).

PC Support/400 connects personal computers connected directly via adapter cards, or connected through LANs. It allows personal computer users to access AS/400* resources over the network and can be used for file transfer, print services, and AS/400* terminal emulation.

Table 11 summarizes the OS/400* networking capabilities.

*Table 11. OS/400* Networking Capabilities*

| | **Ethernet**** | **TokenRing** | **FDDI** | **X.25** | **SDLC** |
|---|---|---|---|---|---|
| **SNA** | YES | YES | - | YES | YES |
| **TCP/IP** | YES | YES | - | YES | YES |
| **OSI** | YES | SOD | - | YES | SOD |

**Note:** SOD (statement of direction)

## 6.7.3 Distributed System Services

### 6.7.3.1 Communication Services

The communication services capabilities within OS/400* are provided for:

- Conversational

  OS/400* provides the CPI-C APIs for APPC conversations using SNA LU 6.2 protocol.

- Remote Procedure Call

  IBM has announced the DCE/400 application toolkit PRPQ which enables application developers to build distributed applications based on DCE RPC.

- Messaging and Queuing

  IBM MQM/400 (5733-103) and ezBRIDGE** Transact** (5787-EDA) provides IBM's Message Queue Interface (MQI) to distributed applications running on OS/400* platform.

### 6.7.3.2 Distribution Services

- IBM has announced the DCE/400 application toolkit PRPQ which will have the following client functions for distributed applications:
  - Directory
  - Security
  - Time.

- Transaction Manager

  OS/400* has a native transaction manager with journaling capability and commit control for transaction integrity and recoverability.

  CICS/400 transaction monitor provides transaction manager functions by employing two-phase commit schemes to insure transactions and data integrity across distributed systems.

## 6.7.4 Application Enabling Services

The application enabling services available with OS/400* are described in this section.

### 6.7.4.1 Presentation Services

- User Interface

  OS/400* uses character displays for directly attached terminals. OS/400* has a comprehensive series of character based dialogues that provide system operator support through menus and help screens. Graphical user interface is available using personal computers and software products like Windows Connection, RUMBA/400, and Easel**.

  IBM intends to provide a client/server application between an OS/400 system and a personal workstation running OS/2* or Microsoft** Windows** with the OS/400 Graphical Operations. It will be based on an iconic, object-oriented, graphical user interface for using and operating the OS/400 system.

- Print/View

  The AFP Utility/400 (5738-AF1) provides the AS/400 platform with distributed printing services as required by the IBM′s Advanced Function Printing* (AFP*) model and architectures.

- Multi-Media

  Ultimedia Host Support/400 (5799-ENY) provides a multimedia front-desk interface for desk-top applications.

  Ultimedia Video Delivery System/400 (5799-EPJ) enables OS/400 applications to manage video presentation to users, including the control of video windows, video devices, and selection of video sources.

### 6.7.4.2 Application Services

- Transaction Monitor

  Transaction processing is a native capability on the OS/400* platform. In addition, transaction processing is also provided by CICS/400* (5738-DFH), which is a member of the CICS* family of products. Interoperability with other CICS* systems is accomplished through Inter-System Communication (ISC) using:

  – Transaction routing

  – Function shipping

  – Distributed program link

  – Distributed transaction processing.

- Mail

  In the office environment, Office Vision/400* (OV/400* 5738-WP1) can interact with OV/VM* and OV/MVS*, sending notes and documents through the network. A function provided by OSI Message Services/400 (5738-MS1) allows message handling according to the X.400 standard, with a direct connection to OV/400*.

#### 6.7.4.3 Data Access Services

- Relational

  Relational database management is a native capability of the OS/400* platform. OS/400* implements the Distributed Relational Database Architecture (DRDA) at the level of the Remote Unit Of Work (RUOW). That architecture provides the OS/400* user with the capability to operate with other relational database environments that implement the same architecture. The OS/400* relational databases can also be accessed from other platforms. The IBM database management systems that implement DRDA RUOW on other software platforms are DB2 AIX/6000, DB2 on MVS/ESA*, SQL/DS on VM/ESA* and VSE/ESA*, and OS/2 database manager (as a RUOW client only).

- File

  Services to access and manage distributed data are also provided by the File Transfer Protocol (FTP) TCP/IP application. FTP allows transfer of files to and from a remote host, and job submission.

  TCP/IP File Server Support/400 (5798-RYW) is a server implementation of NFS** and allows NFS** clients to access and store files on an OS/400* system. Files may be either database or document library object (DLO).

  OSI Communication Subsystem provides OSI File Services (5738-FS1) for remote data access. OSI File Services allow the exchange and remote management of files between platforms that implement an equivalent set of file transfer, access, and management (FTAM) protocols.

  LAN Resource Extension and Services/400 (LANRES/400 5733-CSA) integrates Novell′s NetWare** LAN server with the storage facilities, print services and data distribution services provided by OS/400*.

  Distributed data management (DDM) is a function of OS/400* that supports distributed file and distributed relational data access on remote systems that provide the same DDM functions. Examples are any CICS* systems, other OS/400* systems and System/36 and System/38 systems.

  PC Support/400 allows a personal computer user to access resources on local or remote AS/400*s, and can also be used for file transfer.

- Storage Server

  IBM intends to provide ADSTAR* Distributed Storage Manager/400 (ADSM/400) for the OS/400* system and will support the same clients as the MVS and VM based products (see section 6.5.4.3, “Data Access Services” on page 178).

### 6.7.5 Application Development

The OS/400* participates in the AD/Cycle strategy for application development. The AD/Cycle Cooperative Development Environment/400* (AD/Cycle CODE/400*) provides tools for high level programming language application development and maintenance. The tools operate on OS/2 workstations.

### 6.7.6 System Management

SystemView* is the IBM systems management strategy for managing, planning, coordinating, and operating open, heterogeneous distributed systems. SystemView* includes a SystemView* structure and the SystemView* conforming products. The SystemView* structure is designed to provide a consistent end-user interface, shared data, enhanced automation, and increased integration among systems management products. The structure embraces selected open standards and architectures and defines conformance criteria for the SystemView* products. The SystemView* conforming products are, and will be, developed by IBM and non-IBM vendors.

#### 6.7.6.1 Performance Management and Accounting

Performance management on the OS/400* platform is provided by Performance Tools AS/400 (5738-PT1). It is a set of commands and programs which, using data collected by the OS/400*, assists the user in activities such as workload scheduling, system tuning, performance analysis, and capacity planning.

Accounting services are provided by multiple levels of job accounting, and the capture of job related information through user accounting codes.

#### 6.7.6.2 Operations

SystemView* System Manager/400 (5738-SM1), together with PC Support/400, provides centralized system management functions for a network of interconnected AS/400*s and personal computers.

Local or remote automatic power-on and IPL is provided. Power-off can be done by an interactive command or under program control.

#### 6.7.6.3 Availability and Integrity

The following features are available to expedite recovery in case of hardware or software failure:

- Journaling of the changes to records in a file.
- Commitment control for multiple database update operations.
- User auxiliary storage pools (ASPs) that divide the auxiliary storage, allowing isolation of different applications on different disk arrays.
- Disk mirroring to enable the system to continue to operate after a disk failure.

### 6.7.7 Selected APIs, Protocols, and Facilities

Table 12 summarizes selected protocols, facilities, and APIs available with OS/400*.

*Table 12. OS/400* Selected APIs, Protocols, and Facilities*

| | **OS/400*** |
|---|---|
| **Berkeley** Sockets** | SOD |
| **POSIX**** | SOD |
| **CPI-C** | YES |
| **APPC** | YES |
| **APPN** | YES |
| **NCS**** | - |
| **NQS**** | - |
| **OSF/DCE**(1)** | ANN |
| **SQL** | YES |
| **DRDA-RUOW** | YES |
| **NFS**** | YES |
| **FTP** | YES |
| **FTAM** | YES |
| **OSF/Motif**** | SOD |
| **X-windows**** | SOD |
| **NetWare** server** | YES |
| **LAN & Workstation server** | - |
| **SMTP** | YES |
| **NetView*** | YES |
| **SNMP** | SOD |

**Note:**
ANN (announced)
SOD (statement of direction)
(1) base service

## 6.8 VM/ESA*

Virtual Machine/Enterprise System Architecture (VM/ESA* 5684-112) is an interactive, multi-user operating system. VM/ESA* users can be interactive users or other operating systems.

VM/ESA Release 2 and subsequent releases can execute on any ESA-capable processor, while VM/ESA Release 1.5 (S/370 feature) is still available for execution on non ESA-capable processors.

Figure 65 shows the elements of the VM/ESA* software platform.

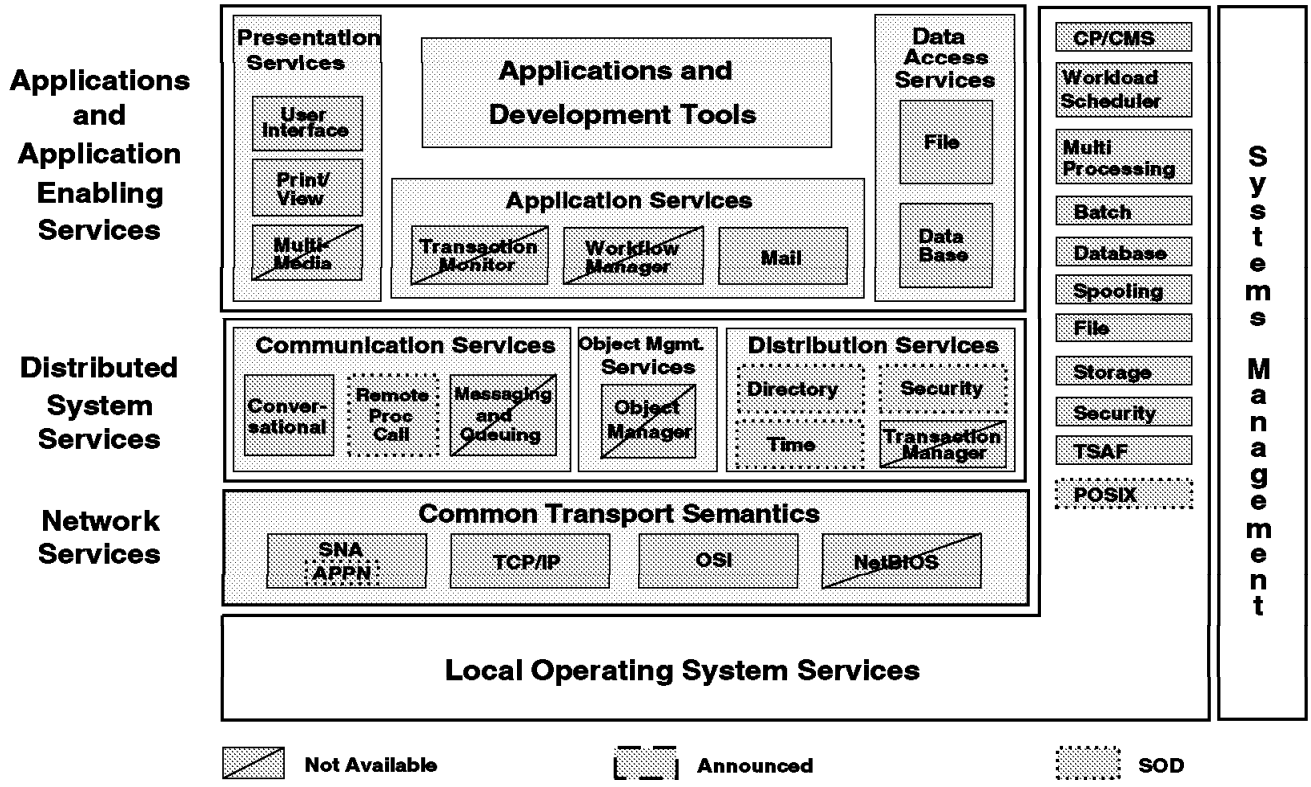


*Figure 65. Elements of the VM/ESA* Software Platform*

### 6.8.1 Local Operating System Services

The basic structure of a VM system is the virtual machine concept, which provides the ability to activate multiple software platforms, called the virtual machines, concurrently on the same hardware processor. Each virtual machine is provided with I/O devices, processor storage, and hardware capabilities, including multiprocessing, along with functions and facilities similar to those of a real machine.

The virtual machine model separates the issues of managing the system resources from those of managing the users. The result is the separation of these two elements into the two primary components of the VM operating system:

- The control program (CP)
- The conversational monitor system (CMS).

#### 6.8.1.1 Control Program

The control program, also called a hypervisor because it supervises multiple supervisors (the managed operating systems), is responsible for resource management. CP manages and operates the real hardware, and the total physical resources are divided into multiple logical entities, the virtual machines. The following are the major features of the VM/ESA* CP:

- Simulation of S/370, S/370-XA, and ESA/390* architectures. Except for timing, the interface provided by CP for the virtual machine is the same as the interface that would be provided by a real processor.

  The operating systems hosted as guest systems include AIX/370 and AIX/ESA*, MVS/ESA*, VM/ESA* itself, VSE/ESA*, TPF, and the non-IBM operating environment, MUMPS**. See Figure 66.

  In fact, any operating system that follows the S/370*, S/370-XA*, ESA/370*, or ESA/390* architecture is capable of executing in a virtual machine.

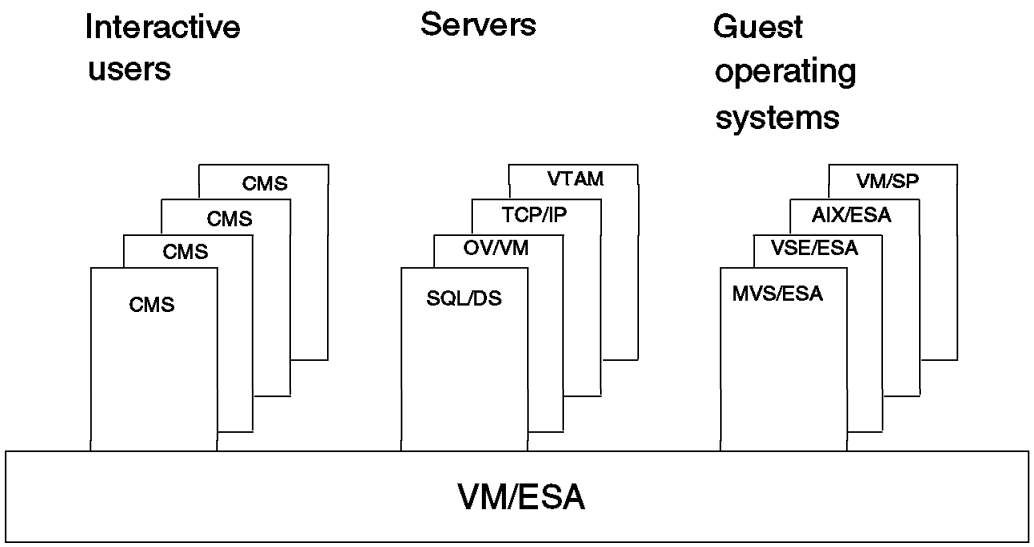


*Figure 66. VM/ESA* Environment*

- Isolation of virtual machines. Events inside a virtual machine cannot affect the operation or the integrity of any other virtual machine.
- Inter-user communication vehicle (IUCV) allows inter-virtual machine communications through a control program interface, instead of through shared storage. This communication capability between virtual machines

allows a VM platform to be considered as a sort of “software local area network,” where some virtual machines might be set up to operate as servers for the other virtual machines in the system.

For example, virtual machines can be activated as telecommunication servers, security servers, or data servers. Services between the client virtual machines and the server virtual machines are exchanged through the IUCV interface. See Figure 67.

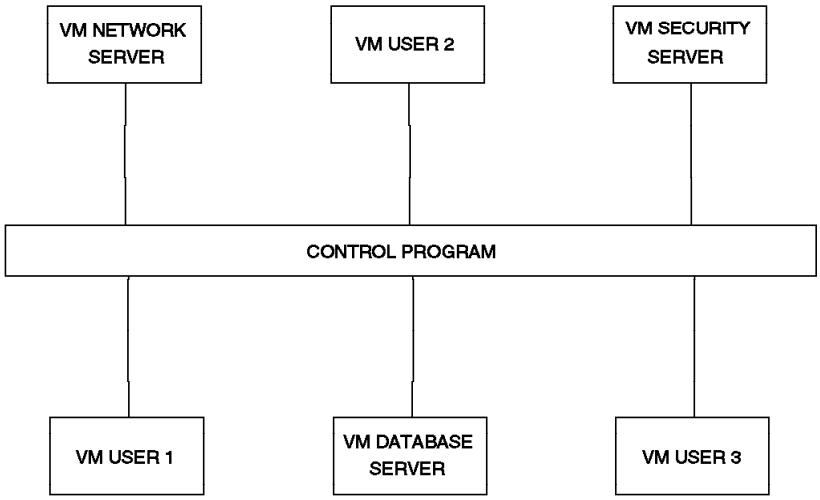


*Figure 67. VM Software Local Area Network*

- Tightly-coupled multiprocessing support allows multiprocessing operations for the virtual machines which have been assigned more than one virtual processing unit (CPU). The dispatchable unit of work for VM/ESA* is the virtual machine CPU.
- The start interpretive execution (SIE) instruction of the ESA/390* architecture is utilized to establish the virtual machine execution environment. The SIE instruction is also used to schedule the virtual machines on the real hardware and to provide microcode emulation of the virtual machine architecture. The functions provided under SIE include execution of privileged instructions, address translation, interrupt handling, and timing facilities.
- VM/ESA* uses the PR/SM* facility to define up to six preferred virtual machines. Preferred machines provide better performance because they benefit from dedicated processor storage and improved I/O operations through the SIE-assist facility.

- The vector facility feature can be used by VM/ESA* guests for computing intensive applications.
- The asynchronous data mover facility (ADMF) can be used by VM/ESA* preferred guests to move data between expanded and central storage asynchronously from the CPU.
- The subspace group facility can be used by VM/ESA* guests to enhance CICS* availability.
- The ESCON* architecture is supported for I/O connectivity available on the ES/9000 processors (see section 6.5.1, “Local Operating System Services” on page 168).
- Up to 65,536 devices can be attached, and a single device may be connected with up to eight channel paths.
- Both central and expanded storage can be utilized. Central storage can be up to 2GB and expanded storage up to 16TB.

IBM has stated its intention to enhance VM/ESA* to include the following POSIX** standards, which will conform to XPG4**:

- 1003.1 (system interfaces)
- 1003.2 (shell and utilities)
- 1003.4 (real-time threads).

#### 6.8.1.2 CMS

The conversational monitor system (CMS) is the user management component of VM. It is an interactive, single-user operating system, designed to run in a virtual machine created and managed by the CP. The main features of CMS are:

- Interactive computing for program development, text editing, problem solving, as well as for productivity applications such as CADAM and OfficeVision/VM* (OV/VM* 5684-084).
- The same language processors and compilers available in VSE and MVS operating systems, and implementation of many of the access methods used by those operating systems.
- Its own file system that uses portions of real disks, called minidisks. Each CMS virtual machine directly manages the format and contents of the attached minidisks. This is similar to a personal computer environment where the user directly manages the hard disks. It also implements a Shared File System (SFS) with hierarchical directories and extensive sharing capabilities for local and remote users.
- Bimodal execution of either the S/370* architecture with a maximum of 16MB of virtual storage, or the S/370-XA* architectures with a maximum of 2047MB of virtual storage.
- CP allows CMS users, using S/370-XA* architecture, to define and share up to 1022 VM dataspaces, each up to 2GB in size. This facility allows large amounts of storage to be made available for data sharing among CMS users and server virtual machines.
- Pipelines provides a programming interface to write chained programs where the output of one program can be the input for the next. This concept is similar to the UNIX** pipes concept.

### 6.8.1.3 Workload Scheduler

The main workload to be managed in a VM/ESA* environment, apart from the guest operating systems, is the CMS interactive virtual machines. These virtual machines might also be seen as early examples of today′s personal computing environments.

The CMS batch facility is a virtual machine available to all CMS users for running programs that would otherwise tie up their virtual machine. The user submits the job to CMS Batch for scheduling, execution, and output processing.

To deal with the interactive and batch workload, VM/ESA* uses a scheduling algorithm that allows all virtual machines to access the processing resources on a priority basis. The interactive users can be given CPU access more frequently than the batch users, but for a shorter period of time.

For a user, VM/ESA* Release 2.2 allows specification of a maximum limit of consumption for the CPU resource, which prevents the user from monopolizing the system.

### 6.8.1.4 Spooling

VM/ESA* provides spool facilities for both CMS virtual machines and guest operating system virtual machines. The RSCS virtual machine (see section 6.8.2, “Network Services” on page 212) can transfer spool data among MVS/ESA*, VM/ESA* and VSE/ESA* systems.

### 6.8.1.5 Database

SQL/DS V3.R4 (5688-103) is the relational database management system for VM/ESA*. It operates as a virtual machine and may be used for decision support systems as well as traditional data processing applications. The Query Management Facility/VM (QMF 5706-255) is available as a high-level query language with graphic capabilities.

The SQL/DS virtual machine is available to CMS users as a server for the development, testing, and execution of interactive applications.

A VSE/ESA virtual machine guest of a VM/ESA* system, with the implementation of the SQL/DS guest sharing feature, can execute transactions to access information residing on a VM/ESA* SQL/DS virtual machine.

### 6.8.1.6 File

CMS has it own flat file system. Flat files (minidisks) are contiguous allocations of disk space owned by a specific user.

The Shared File System (SFS) is an extension of the CMS file system that offers additional file management and sharing functions. Data sharing is allowed among users and across multiple VM/ESA* systems. The file structure is a hierarchy of directories and is managed by an SFS server virtual machine. The SFS server controls the physical placement of data and the CMS users access to data in the local VM/ESA* system, or in distributed VM/ESA* systems connected with either a SNA link, or a transparent service access facility (TSAF) connection. The SFS uses the coordinated resource recovery (CRR) services to guarantee the integrity of data (see section 6.8.6.3, “Availability and Integrity” on page 217).

The performance of I/O operations in VM/ESA* can be improved with the IBM 3990 Model 3 and 6 DASD control units for normal caching, fast write

capabilities, dual copy, and concurrent copy. Data-in-memory caching through VM dataspaces, virtual disks, and minidisk caching, can provide additional performance benefits.

#### 6.8.1.7 Storage Services

Data Facility Storage Management Subsystem/VM (DFSMS/VM*) is a feature of VM/ESA*. It provides a system managed storage environment with, among other features:

- Customer defined, policy based, automated space management function for Shared File System files
- Support for fixed block architecture (FBA) disks, in addition to the traditional Count Key Data (CKD) architecture, allowing easy migration to and from the two architectures.

#### 6.8.1.8 Security

The Resource Access and Control Facility (RACF 5740-XXH) provides the access control functions of user identification and verification, resource authorization, and logging and reporting of access events.

The ES/9000* Integrated Cryptographic Facility is available to VM/ESA* guests.

### 6.8.2 Network Services

VM supports a wide range of connectivity options for the three main network protocols, SNA, TCP/IP, and OSI.

- SNA

  ACF/VTAM (5684-095) with ACF/NCP provides SNA networking services with a wide range of connectivity options. Several applications are available in the SNA environment to perform functions such as, file transfer, electronic mail, document exchange, and remote logon.

  APPC/VM provides the application programming interface for conversations based on the SNA LU6.2 protocols.

  IBM intends to support SNA′s Advanced Peer-to-Peer Networking (APPN) interface for VM/ESA*.

- TCP/IP

  TCP/IP V.2 (5735-FAL) allows VM users to intercommunicate on a TCP/IP network. Applications include the ability to send mail, transfer files, logon to a remote host, and to provide multiple server functions for network users.

  The major functions and applications provided by TCP/IP are:

  – Terminal passthrough (TELNET) allows a user client to remotely logon to another computer with TCP/IP TELNET server function.
  – Simple Mail Transfer Protocol (SMTP) allows an end-to-end electronic mail exchange.
  – Simple Network Management Protocol (SNMP) is provided for network management.
  – File Transfer Protocol (FTP) (see section 6.8.4.3, “Data Access Services” on page 215).
  – Network File System ( NFS** ) (see section 6.8.4.3, “Data Access Services” on page 215).

- X-windows** and Motif** client facilities allow program access to an high resolution display connected to a system running a graphic server system (see section 6.8.4.1, “Presentation Services” on page 214).
- The Remote Procedure Call (RPC) library allows program-to-program communication for distributed applications with higher level program calls.
- Network Computing System (NCS**) allows high level calls to procedures to be executed on remote systems.
- Berkeley** socket library for distributed applications communication.

SNAlink allows TCP/IP communications over an SNA network.

- OSI

  OSI Communication Subsystem (5684-013) provides OSI support in VM/ESA* environment. The major OSI functions and applications provided are:

  - OSI message exchange for message handling according to the X.400 standard (see section 6.8.4.2, “Application Services” on page 214).
  - OSI File Services (see section 6.8.4.3, “Data Access Services” on page 215).
  - The Remote Programming Interface (RPI) for development of distributed applications.

NetView* for VM/ESA* (5756-051), together with VTAM, TCP/IP, and OSI provides management functions for a heterogeneous network.

The Remote Spooling Communications Subsystem (RSCS 5684-096) is a VM networking application that runs in a virtual machine and provides data transfer services. It allows users in a VM system to send messages, files, mail, commands, and jobs to other systems within a network. The other systems can be AIX, MVS, VM, and VSE.

The VM pass-through facility (PVM 5684-100) enables VM users on a local or remote 3270 terminal, or personal workstation that emulates a 3270 display, to interactively access applications on a remote system, with BSC network and CTC connections.

Table 13 summarizes the VM/ESA* networking capabilities.

*Table 13. VM/ESA* Networking*

| | **Ethernet**** | **TokenRing** | **FDDI** | **X.25** | **SDLC** |
|---|---|---|---|---|---|
| **SNA** | YES | YES | YES | YES | YES |
| **TCP/IP** | YES | YES | YES | YES | YES (1) |
| **OSI** | YES | SOD | SOD | YES | - |

**Note:**
SOD (statement of direction)
(1) Host TCP/IP to host TCP/IP with SNALink

## 6.8.3 Distributed System Services

### 6.8.3.1 Communication Services

The communication services capabilities for VM/ESA* are provided for:

- Conversational

  Conversation protocols using the SNA LU 6.2 interface or the APPC/VM interface are available.

- Remote Procedure Call

  IBM intends to include, in VM/ESA*, the OSF/DCE** programming interfaces for Remote Procedure Call (RPC). The DCE RPC allows programs to work across heterogeneous systems, masking the differences between data representations on different software platforms and differences in networking protocols.

### 6.8.3.2 Distribution Services

IBM intends to include, in VM/ESA*, the following OSF** DCE** base services for distributed applications:

- DCE Cell Directory Service (CDS) Client
- DCE Security Service Client
- DCE Distributed Time Service (DTS) Server and Client.

## 6.8.4 Application Enabling Services

The application enabling services available with VM/ESA* are described in this section.

### 6.8.4.1 Presentation Services

- User Interface

  VM/ESA* uses character displays for attached terminals. However, applications can be written to use the presentation server of an X-windows** workstation in a TCP/IP network (APIs are provided), or the presentation manager of a PS/2 through an APPC application.

- Print/View

  Print Services Facility/VM (PSF/VM 5684-141) provides the VM platform with the printing services as required by the IBM′s Advanced Function Printing* (AFP*) model and architectures.

  TCP/IP V.2 (5735-FAL) provides a Line Printer Client/Line Printer Daemon (LPR/LPD) component with client/server support. The client LPR acts as the CMS print command, and the server LPD can use locally attached printers, RSCS connections, and remote printers on systems that use LPR/LPD protocol.

### 6.8.4.2 Application Services

- Mail

  OSI message exchange, provided with OSI Communications Subsystem, allows message handling according to the X.400 standard in a multi-vendor environment.

Office Vision/VM* (OVM 5684-084), with the X.400 PROFS* Connection/VM (5785-GCG), provides mail functions using the X.400 OSI protocol.

#### 6.8.4.3 Data Access Services

- Relational

  SQL/DS V3 R4 (5688-103) is the relational database management system for VM/ESA*. It operates in a virtual machine and may be used for decision support systems as well as traditional data processing applications.

  CMS users can use the SQL/DS virtual machine as a server for development, testing, and execution of interactive applications.

  SQL/DS implements the Distributed Relational Database Architecture (DRDA) at the level of the Remote Unit of Work (RUOW). Other IBM relational data base management systems that also implement DRDA at the RUOW level are DB2 on AIX/6000, DB2* on MVS/ESA*, the OS/400* database manager, SQL/DS on VSE/ESA*, and the OS/2* database manager (only as a client).

- File

  Services to access and manage distributed data are provided by the following TCP/IP applications:

  – File Transfer Protocol (FTP) allows file transfer to and from a remote host using a command syntax compatible with UNIX**. FTP also allows job submission.

  – Network File System (NFS**) allows a user to have transparent access to formatted files and byte-stream files distributed over a network.

  The OSI Communications Subsystem combined with OSI File Services (5684-038) provides an application with remote data access capability. OSI File Services allows the exchange and remote management of files between platforms that implement an equivalent set of file transfer, access, and management (FTAM) protocols.

  The LAN File Services/ESA (5648-039) provides a file system on VM/ESA* that is compatible with file systems on LAN workstations. The workstation file system on VM supports DOS, OS/2*, and UNIX** format files, file operations, multiple directories, and locking. Workstations on a LAN can have shared files on the VM/ESA* platform for storage intensive applications.

  LAN Resource Extension and Services/VM (LANRES/VM 5684-142) integrates Novell's NetWare** LAN server with the storage facilities, print services, and data distribution services provided by VM/ESA*. High speed connectivity is achieved by attaching the LAN server PC to an ES/9000 channel using a special adapter card. The function and facilities provided by LANRES are:

  – Disk server, where the VM/ESA* disks are used like NetWare** disks.

  – Print server provides for both LAN-to-Host and Host-to-LAN printing.

  – Central data distribution, where data can be copied between VM/ESA* and the server station.

- Storage Server

  ADSTAR* Distributed Storage Manager (ADSM*, 5648-020) allows a VM/ESA* system to act as a file backup and archive server for LAN file servers and workstations. It operates as a server for OS/2*, DOS, AIX/6000, SUN** OS,

Apple Macintosh**, Windows**, Novell**, DEC** ULTRIX** and SCO** Open Desktop** software platforms.

## 6.8.5 Application Development

VM/ESA* offers extensive testing capabilities for both VM applications and application development for guest operating systems. VM/ESA* participates in AD/Cycle with the AD/Cycle Cooperative Development Environment/370 (AD/Cycle CODE/370). It provides a seamless interface between VM/ESA* and OS/2 for the edit, compile, and debug environments.

## 6.8.6 System Management

SystemView* is the IBM systems management strategy for managing, planning, coordinating, and operating open, heterogeneous distributed systems. SystemView* includes a SystemView* structure and the SystemView* conforming products. The SystemView* structure is designed to provide a consistent end-user interface, shared data, enhanced automation, and increased integration among systems management products. The structure embraces selected open standards and architectures and defines conformance criteria for the SystemView* products. The SystemView* conforming products are, and will be, developed by IBM and non-IBM vendors.

VM/ESA* provides systems management tools for one or more VM systems as well as for locally attached and remote workstations.

### 6.8.6.1 Performance Management and Accounting

When accounting record creation is active, CP collects accounting statistics for each virtual machine while the virtual machine is operating, and data about resource utilization.

There are several licensed programs that process the data collected by CP Monitor. Among them are:

- VM Monitor Analysis Program (VMMAP 5664-191) and VM Performance Reporting Facility (VMPRF 5684-073) that provide reports for performance measurement, tuning, and capacity planning.
- Real Time Monitor VM/ESA* V1.R5.2 (RTM 5798-DWD) for real time statistical analysis. This monitor can send alerts to the SystemView* Host Management Facility/VM (5684-157) when an installation defined threshold is exceeded, and automatic actions can be initiated.

### 6.8.6.2 Operations

NetView* for VM/ESA* (5756-051) allows management of multiple remote VM systems from a single location. With functions provided by SystemView* Host Management Facility/VM, it is possible to automate console operations. Furthermore, NetView* can analyze performance data collected by the real time monitor (RTM) against installation defined thresholds and, if exceeded, automatically initiate customer specified actions.

VM/ESA* uses the programmable operator facility for automation of host virtual machine servers. It does this by intercepting all messages and requests directed to virtual machines, and handling them according to defined actions.

#### 6.8.6.3 Availability and Integrity

The operator can use VM/ESA* CP commands to fence discrete hardware components (devices, control units, channel paths, and processors) and allow concurrent hardware maintenance and system operation.

With the ESCON* architecture, it will be possible to install and use devices without interrupting operations through the VM/ESA* Hardware Configuration Definition* (HCD*) function.

The CMS coordinated resource recovery (CRR) facility ensures that an application program can update multiple resources while maintaining resource integrity. This means that all updates to resources within a transaction are either done (committed) or not done (backed out). CRR coordinates the updating of protected resources, whether the resources are within the same processor or distributed among processors within a TSAF collection or an SNA network.

The dual copy capability (disk mirroring), possible with the advanced functions of the 3990 model 3 and 6 control units, enables the system to operate after a disk failure, with no loss of data or interruption to service.

### 6.8.7 Selected APIs, Protocols, and Facilities

Table 14 summarizes selected protocols, facilities, and APIs available with VM/ESA*.

*Table 14. VM/ESA* Selected APIs, Protocols and Facilities*

| | **VM/ESA*** |
|---|---|
| **Berkeley** Sockets** | YES |
| **POSIX**** | SOD |
| **CPI-C** | YES |
| **APPC** | YES |
| **APPN** | SOD |
| **NCS**** | YES |
| **NQS**** | - |
| **OSF/DCE**** | SOD |
| **SQL** | YES |
| **DRDA-RUOW** | YES |
| **NFS**** | YES |
| **FTP** | YES |
| **FTAM** | YES |
| **OSF/Motif**** | YES |
| **X-windows**** | YES |
| **NetWare** server** | YES |
| **LAN & Workstation server** | YES |
| **SMTP** | YES |
| **NetView*** | YES |
| **SNMP** | YES |
| **Note:** SOD (statement of direction) | |

## 6.9 VSE/ESA*

VSE/ESA* (5750-ACD) is a member of IBM's family of ESA/390* operating systems and is designed for transaction and batch processing on small and intermediate ES/9000* processors.

VSE/ESA* is used primarily in the following environments:

- Primary operating system for low-end ES/9000* systems
- Operating system on distributed ES/9000* nodes managed from a central site.

Figure 68 shows the elements of the VSE/ESA* software platform

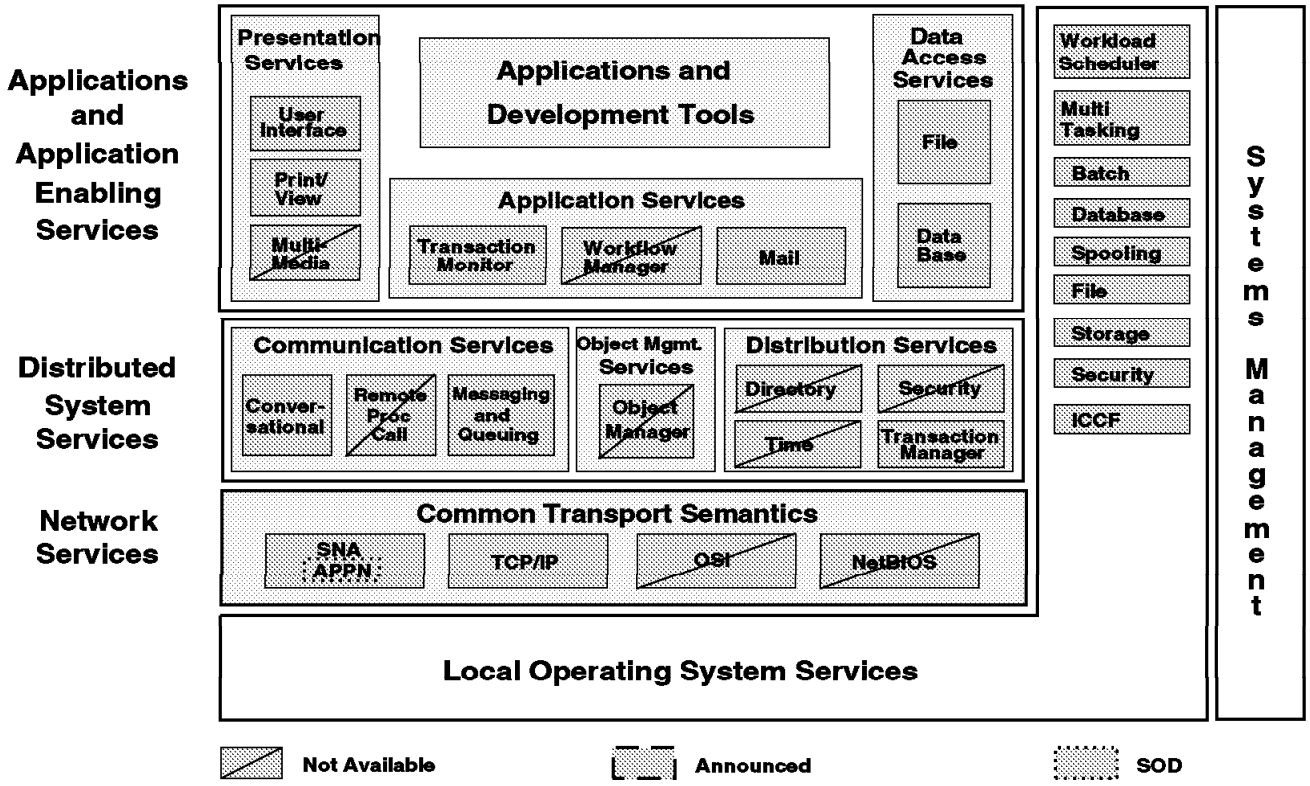


*Figure 68. Elements of the VSE/ESA* Software Platform*

### 6.9.1 Local Operating System Services

VSE/ESA* fully exploits the ESA architecture and has the following main characteristics (the information provided here applies to VSE/ESA* V1.3):

- Virtual storage addressing
  - The virtual addressing capability of VSE/ESA* is 2GB, and the complete range is known as an address space. Work is executed in partitions, and the maximum number of partitions is 200. A partition can be assigned to a batch job, to a system function, or to a subsystem, such as CICS*.

– VSE/ESA* allows an application to use dataspaces for additional virtual storage. Normal address spaces may contain programs and data. Dataspaces can only contain data and can be up to 2GB in size.

– VSE/ESA* provides virtual disks to allow data, which would otherwise be stored on DASD, to reside in virtual storage. Virtual disks are implemented in dataspaces and appear to the user as normal DASD devices. Data on virtual disks is volatile, and therefore, not appropriate for permanent files.

– Total virtual storage available for address spaces and dataspaces (including virtual disks) is 90GB.

– Maximum real storage size is 2GB, and ES9000* expanded storage is not utilized.

Figure 69 shows the VSE/ESA* virtual storage structure.

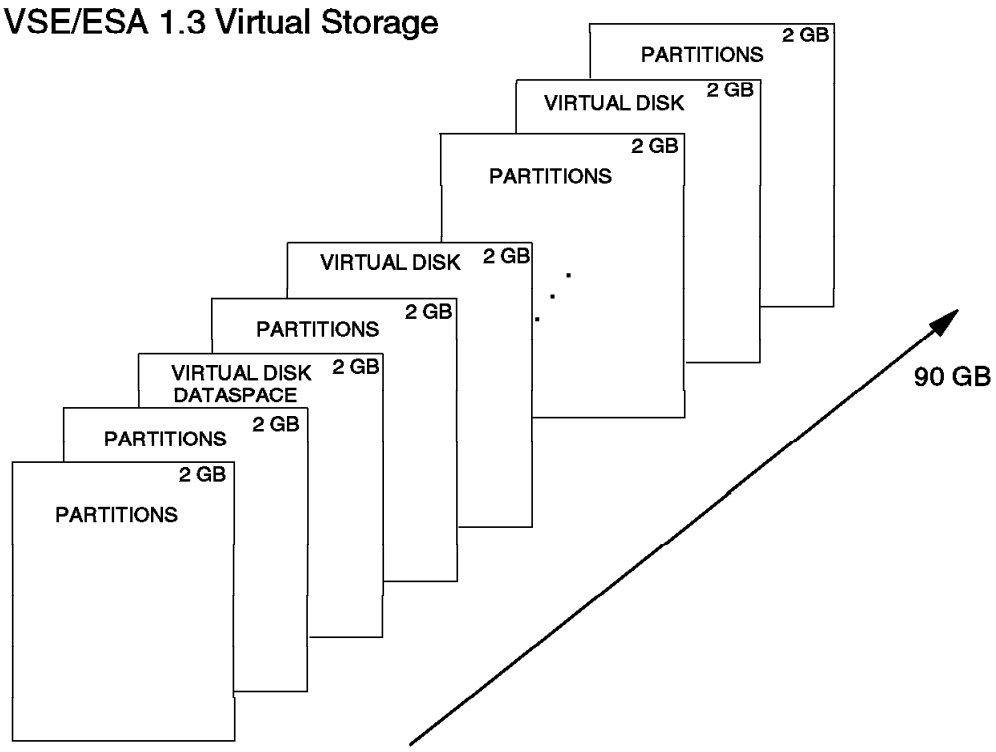


*Figure 69. VSE/ESA* 1.3 Virtual Storage*

- VSE/ESA* is a multi-user, multitasking operating system. Multiprocessing is not supported.
- VSE/ESA* supports the ESCON* architecture for I/O connectivity on the ES9000* processors (see section 6.5.1, “Local Operating System Services” on page 168)
  – Up to 1024 local channel attached I/O devices are supported with as many as four channel paths for a single device.

  - DASD data set sharing among VSE operating systems is possible with the “lock file” facility.

- VSE/ESA* can be used with the S/370 architecture. In this case, virtual storage is limited to 16MB, and the ESA architecture for I/O operations is not available.

IBM has also made a statement of direction to support, on the VSE/ESA* platform, the following hardware facilities:

- The data compression facility of selected ES/9000* processor family for users requiring DASD space saving and communication line transmission time saving.
- The subsystem storage protection facility to enhance CICS* availability.

#### 6.9.1.1 Workload Scheduler

VSE/ESA* can schedule different categories of workload:

- Batch processing through VSE/POWER (5686-033) job entry subsystem. Jobs can be submitted locally, by Remote Job Entry (RJE), and by Interactive Computing and Control Facility (ICCF 5686-036) users.
- Online transaction processing through the CICS* transaction monitor.
- ICCF user, which runs as a CICS* application.

VSE/ESA* is a priority scheduler system that utilizes a partition load balancing algorithm to manage the workload. Partitions can be grouped into classes which have a user specified priority. Within each class, the partitions are rotated from the first to the last position, thus giving equal access to the CPU resource. VSE/ESA* does not swap users out of storage when computing resources are overcommitted.

#### 6.9.1.2 Spooling

VSE/POWER is a spooling system for automatic storage of user-related data, mainly output data for later printing, and priority scheduling of jobs queued for execution. VSE/POWER provides spool interfaces for all categories of workload, batch, CICS* online transactions, and ICCF users. VSE/POWER can send and receive spool files over an Network Job Entry (NJE) network connecting the spooling system of MVS/ESA*, VM/ESA*, and VSE/ESA* platforms.

#### 6.9.1.3 Database

SQL/DS (5798-DQL) provides relational data base services for SQL based applications in the VSE/ESA* environment.

DL/I DOS/VS (5746-XX1) is the hierarchical database management system for VSE/ESA*.

Both database managers can be used by CICS* applications and batch programs.

In addition, there are other vendor products for data management, such as IDMS**.

#### 6.9.1.4 File

VSE/ESA* supports traditional data access methods, such as VSAM, direct access, sequential, and partitioned, and provides a service to share data sets among VSE platforms. The integrity of shared data sets, however, is an application responsibility.

The functions of the IBM 3990 Model 3 and 6 DASD control (basic cache, dual copy, DASD fast write) are used by VSE/ESA* to improve performance and data availability for I/O operations.

#### 6.9.1.5 Storage Services

VSE/ESA* does not have functions that automatically manage space on storage devices or perform data migration and backup, but there are other vendor products that provide these services, such as Dynam**.

#### 6.9.1.6 Security

VSE/ESA* provides security through its access panel. The user is required to enter ID and password to logon to CICS* and to ICCF, and password verification is done by applications.

In addition, VSE/Access Control Logging and Reporting (ACLR 5746-XE7) can monitor and report user access to protected data sets, libraries, and programs, as well as user identification and verification for batch jobs.

Data and network security is provided by database managers and networking subsystems.

Some vendors provide security packages that can be used for security services, among them, ACF2**.

### 6.9.2 Network Services

VSE/ESA* provides networking capabilities for wide area networks (WANs), LANs, POWER job entry subsystem networking, and network management.

ACF/VTAM (5666-363), ACF/NCP (5668-738, 5668-854) and Emulator Program (EP) allow WAN communication using SDLC, binary synchronous, start-stop, and X.25 protocols. LAN support in VSE/ESA* allows communication with IBM TokenRing, Ethernet**, and FDDI.

- SNA

  The CPI-C, through CICS* (see section 6.9.4.2, “Application Services” on page 223), and LU 6.2 programming interfaces used by APPC are available for conversations over a SNA network.

  IBM has also made a statement of direction for Advanced Peer-to-Peer Networking (APPN) on the VSE/ESA* platform.

- TCP/IP

  OpenConnect System** TCP/IP for VSE enables client/server access to application, by providing TCP/IP network connections to VSE systems. Connectivity between TCP/IP networks and VSE systems is provided by an Outboard Communication Server (OCS) gateway, available as software (5758-PC3) on AIX/6000 and other UNIX** software platforms. The functions and applications provided are:

- OpenConnect System** terminal passthrough (TELNET 5758-PC2) client allows a user to remotely logon to another computer with TCP/IP TELNET server function.
- OpenConnect System** File Transfer Protocol (FTP 5758-PC0, PC1) client and server applications (see section 6.9.4.3, “Data Access Services” on page 223).

IBM and OpenConnect System** intend to include the Sockets application programming interface in the TCP/IP for VSE/ESA*.

- OSI

  The Open System Interconnect (OSI) remote programming Interface (RPI) enables OSI applications in VSE/ESA* that communicate with the OSI communication subsystem on VM/ESA* or MVS/ESA* over an SNA network. This function is provided by ACF/VTAM.

POWER networking provides facilities that allow file transfer and remote job entry between MVS/ESA*, OS/400*, VM/ESA*, and VSE/ESA* platforms. POWER networking allows these systems to exchange data and to interoperate (see section 6.9.1.1, “Workload Scheduler” on page 220).

Table 15 summarizes the VSE/ESA* networking protocols.

*Table 15. VSE/ESA* Networking*

| | Ethernet** | TokenRing | FDDI | X.25 | SDLC |
|---|---|---|---|---|---|
| **SNA** | YES (1) | YES | YES (1) | YES | YES |
| **TCP/IP** | YES (2) | YES (2) | YES (2) | YES (2) | YES (2) |
| **OSI** | - | - | - | - | YES (3) |

**Note:**
(1) 3172 required.
(2) TCP/IP gateway provided by OCS on RISC/6000.
(3) OSI applications use VTAM to communicate.

## 6.9.3 Distributed System Services

### 6.9.3.1 Communication Services

The communication services capabilities within VSE/ESA* are provided by:

- Conversational

  Communication services for distributed applications are available on the VSE/ESA* platform in the form of conversations provided by SNA LU 6.2 protocols and APPC.

- Messaging and Queuing

  The product ezBRIDGE** Transact** (5787-ECX) on VSE/ESA provides IBM’s Message Queue Interface (MQI) to distributed applications.

### 6.9.3.2 Distribution Services

- Transaction Manager

  CICS transaction monitor qualifies as a transaction manager because it employs 2-phase commit to ensure transaction and data integrity across distributed systems.

### 6.9.4 Application Enabling Services

The application enabling services available within VSE/ESA* are described in this section.

#### 6.9.4.1 Presentation Services

- User Interface

  VSE/ESA* V 1.3, with the VSE/ESA* Workstation Platform (5686-028), the ScreenView* Feature for VSE/ESA* (5775-BDX), and the OS/2 technology, provides graphic and navigation services on the OS/2 platform and services for host-workstation cooperative processing.

  VSE/ESA* has a comprehensive series of CICS*/ICCF dialogues, called the interactive user interface (IUI). IUI is a character interface, providing system programming and operations support through menus and help screens.

- Print/View

  Print Services Facility/VSE (PSF/VSE 5686-040) provides the VSE platform with the printing services as required by the IBM's Advanced Function Printing* (AFP*) model and architectures.

#### 6.9.4.2 Application Services

- Transaction Monitor

  CICS/VSE (5686-026) is the transaction monitor on the VSE/ESA* platform and is a member of the CICS* family of transaction monitors, available on other IBM software platforms (AIX/6000, OS/2*, OS/400*, MVS/ESA*).

  Members of the CICS* family of transaction monitors can communicate through several distinct mechanisms, such as transaction routing, function shipping, distributed program link, and distributed transaction processing. These mechanisms allow CICS* to route transactions to another CICS* for execution, allow transparent access to remote CICS* resources, and allow usage of remote CICS* applications.

  CICS* also provides an interface to CPI-C for program-to-program communications, and supports the CICS* Callpath family to communicate with industry standard private branch exchanges for telephone applications.

  The inter-system communication facility (ISC) of CICS*/VSE allows communication between CICS* and any other CICS* or IMS transaction monitor.

- Mail

  In the office environment, VSE/ESA*, with a DISOSS (5666-270) host-based application (which runs with the CICS* monitor), can send notes and documents through the network to other DISOSS systems.

#### 6.9.4.3 Data Access Services

- Relational

  SQL/DS (5688-103) is the relational database management system for VSE/ESA*. It operates in a partition, and may be used as a decision support system as well as for traditional data processing applications. The Query Management Facility/VSE (QMF 5666-292) is available as a high level query language with a graphical user interface.

  SQL/DS and VSE/ESA* allow client applications that support Distributed Relational Database Architecture (DRDA) Remote Unit Of Work (RUOW) to

access relational data residing in VSE/ESA*. Other platforms that have implemented the DRDA RUOW are AIX/6000, DOS, MVS/ESA*, OS/2*, OS/400*, and VM/ESA*.

- File

  Distributed Data Management (DDM 5686-018) allows access to VSAM data sets from AS/400 and work stations running OS/2* with the DDM function installed. It operates with CICS* transaction monitor as a transaction program and uses SNA LU 6.2 conversation protocols.

  OpenConnect System** File Transfer Protocol allows transfer of files to and from a remote host or workstation connected to a TCP/IP network, using the same command syntax as UNIX**.

### 6.9.5 Application Development

Application development in VSE/ESA* is enhanced by participation in AD/Cycle. VSE/ESA* participates in Cross System Product/Application Development (CSP/AD 5668-813) and Cross System Product/Application Execution (CSP/AE 5668-814) application generators and execution platforms. Interactive support is provided by CICS*.

The VSE workstation platform provides an application development workplace on an OS/2 workstation, integrated with the VSE/ESA* system.

### 6.9.6 System Management

SystemView* is the IBM systems management strategy for managing, planning, coordinating, and operating open, heterogeneous distributed systems. SystemView* includes a SystemView* structure and the SystemView* conforming products. The SystemView* structure is designed to provide a consistent end-user interface, shared data, enhanced automation, and increased integration among systems management products. The structure embraces selected open standards and architectures and defines conformance criteria for the SystemView* products. The SystemView* conforming products are, and will be, developed by IBM and non-IBM vendors.

#### 6.9.6.1 Performance Management and Accounting

The VSE/ESA* interactive user interface of ICCF provides dialogues that simplify systems programming and operational tasks. A dialogue gives a snapshot of system performance factors, such as processor utilization, jobs executing, and I/O device counts. Accounting services are provided by VSE/POWER.

CICSPARS (5666-329) is available for performance data, tuning, accounting, and capacity planning for a CICS/VSE transaction monitor.

Other performance monitors and tools are available from other vendors, such as Omegamon** and Explore**.

#### 6.9.6.2 Operations

NetView* for VSE/ESA* (5686-038) provides network and systems management functions. It provides remote automatic operations and recovery, network monitoring and control, along with remote node support.

VSE/Operator Communication Control Facility (OCCF 5746-XC5) provides services to reduce or automate the operator interaction necessary to run remote

systems. It can also direct messages to NetView* for system interaction and response.

#### 6.9.6.3 Availability and Integrity

CICS* availability under VSE/ESA* can be improved by the XRF feature, and alert management can be provided by VSE when used with NetView*.

Hardware maintenance concurrent with systems operations can be provided by operator action to fence physical resources. The dual copy capability (disk mirroring) available with the advanced functions of the 3990 Model 3 and 6 control units, enables the system to continue operations after a DASD failure.

### 6.9.7 Selected APIs, Protocols, and Facilities

Table 16 summarizes selected protocols, facilities, and APIs available with VSE/ESA*:

*Table 16. VSE/ESA* Selected APIs, Protocols and Facilities*

| | **VSE/ESA*** |
|---|---|
| **Berkeley** sockets** | SOD |
| **POSIX**** | - |
| **CPI-C (1)** | YES |
| **APPC** | YES |
| **APPN** | SOD |
| **NCS**** | - |
| **NQS**** | - |
| **OSF/DCE**** | - |
| **SQL** | YES |
| **DRDA-RUOW(SQL/DS V3.4)** | YES |
| **NFS**** | - |
| **FTP** | YES |
| **FTAM** | - |
| **OSF/Motif**** | - |
| **X-windows**** | - |
| **NetWare** server** | - |
| **LAN & Workstation server** | - |
| **SMTP** | - |
| **NetView*** | YES |
| **SNMP** | - |

**Note:**
SOD (statement of direction)
(1) Through CICS*

# Appendix A. APIs, Protocols, and Facilities Description

This appendix provides a brief description of the services and APIs mentioned in the summary tables for each of the IBM software platforms.

**Berkeley** sockets**

APIs for the services of Transport Connect Protocol Internet Protocol (TCP/IP), User Datagram Protocol (UPD), and Internet Control Message protocol (ICMP) under TCP/IP. They are peer-to-peer APIs and were introduced in the UNIX** Berkeley Software Distribution (BSD) 4.2 generic interfaces for UNIX**-to-UNIX** network communications.

**POSIX****

Portable Operating System Interface definition is a set of specifications for open systems developed by the Technical Committee on Operating Systems (TCOS) of the Institute of Electrical and Electronic Engineers (IEEE**). Some of the POSIX** committees are:

- 1003-0 Open system
- 1003-1 System interfaces
- 1003-1a System interface extensions
- 1003-2 Shell and utilities
- 1003-3 Testing and verifications
- 1003-4 Real-time threads
- 1003-5 ADA language bindings
- 1003-6 Security extension
- 1003-7 System administration
- 1003-8 Networking
- 1003-9 Fortran language bindings
- 1003-10 Supercomputing
- 1003-11 Transaction processing
- 1003-12 Protocol-independent network API
- 1003-13 X.500 namespace and directory services
- 1003-14 Real time profile
- 1003-15 Batch processing
- 1003-16 Multi-processing

POSIX** standards are included in the X/Open Portability Guide (XPG4**).

**CPI-C**

Common Programming Interface for Communication is the IBM System Application Architecture* (SAA*) API for program-to-program communications (using SNA LU6.2). IBM provides CPI-C language bindings for all its SAA* compilers.

**APPC**

Advanced Program-to-Program Communication is the name given to various implementations of program-to-program communications based on the conversation model and the SNA LU6.2 architecture. Actual APPC implementations may differ from platform to platform and from product to product.

**APPN**

Advanced Peer-to-Peer Networking is an extension to the IBM SNA architecture for peer-to peer networking.

**NCS****

Network Computing System is an interface that allows high level program calls to procedures in a distributed system that is connected to a TCP/IP network. Its functions are similar to the RPC functions.

**NQS****

Network Queueing System allows batch job submission from workstations to hosts in a TCP/IP network.

**OSF/DCE****

DCE is a high level software platform, defined by OSF**, that provides services for application interoperability among heterogeneous software platforms. It consists of such services as:

- Thread services
- Remote Procedure Call
- Time service
- Naming service
- Security service
- Distributed file service.

**SQL**

Structured Query Language is a standard language interface for relational database access and manipulation. It can be used for interactive query language, database programming language, database definition, and data administration.

**DRDA**

Distributed Relational Database Architecture is the IBM architecture for distributed relational database operations.

**NFS****

Network File System is a distributed file system for hierarchical file structure. NFS** uses the RPC to communicate between client and server, and allows file sharing among users.

**FTP**

File Transfer Program allows file transfer from one system to another, over a TCP/IP network. FTP supports the transfer of both binary and ASCII files, and its services can be invoked using commands or API calls.

**FTAM**

File Transfer Access and Management is a standard protocol available with OSI. It allows exchange and remote management of files between platforms implementing these protocols.

**Motif****

Motif** is a graphical user interface developed by the Open Software Foundation

**X-windows****

X-windows** is a client/server presentation service protocol. The X-windows** server is a workstation that allows its display screen and mouse to be monitored by a remote X-windows** client. The server can have several windows, each of which is controlled by one client. The server provides APIs for remote programming over the network to share its display, keyboard, and mouse.

**NetWare** server**

Disk, file, backup, and archive services provided by a host to a NetWare** client.

**LAN & workstation server** Disk, file, backup and archive services provided by a host to a LAN file server and to workstations.

**SMTP** Simple Mail Transfer Protocol provides an electronic mail protocol for transferring messages over a TCP/IP network, from a sender client to a receiver server.

**NetView*** NetView* is a family of products that provide the basis for heterogeneous network management and systems operations from a central or remote site. NetView* participates in the IBM* SystemView* infrastructure.

**SNMP** Simple Network Management Protocol is used for managing TCP/IP multivendor networks. SNMP separates the job of network management into two tasks, the agent that monitors the network node, and the manager that communicates with the distributed agents for information and command processing.

# List of Abbreviations

| | |
|---|---|
| ***ABIOS*** | Advanced Basic Input/Output Bios |
| ***API*** | Application Programming Interface |
| ***APPC*** | Advanced Program to Program Communications |
| ***APPN*** | Advanced Peer to Peer Networking |
| ***BIOS*** | Basic Input/Output System |
| ***BCU*** | Bus Control Unit |
| ***CAE*** | Common Application Environment |
| ***CISC*** | Complex Instruction Set Computers |
| ***CP*** | Central Processor |
| ***CPI-C*** | Common Programming Interface for Communications |
| ***CMOS*** | Complementary Metal Oxide Semiconductor |
| ***DMA*** | Direct Memory Access |
| ***DME*** | Distributed Management Environment |
| ***DRDA*** | Distributed Relational Database Architecture |
| ***ESCON*** | Enterprise System Connection |
| ***EUI*** | End User Interface |
| ***FDDI*** | Fiber Distributed Data Interface |
| ***FSV*** | Full System Vendor |
| ***FTAM*** | File Transfer Access and Management |
| ***FTP*** | File Transfer Program |
| ***HSB*** | High Speed Buffer |
| ***ICU*** | Interconnect Unit |
| ***IEEE*** | Institute of Electrical and Electronics Engineers |
| ***IMPI*** | Internal Microprogramming Interface |
| ***ISA*** | Industrial Standard Architecture |
| ***IWS*** | Intelligent Workstation |
| ***LAN*** | Local Area Network |
| ***MLI*** | Machine Level Interface |
| ***MVS*** | Multiple Virtual Storage |
| ***NCS*** | Network Computing System |
| ***NFS*** | Network File System |
| ***NOS*** | Network Operating System |
| ***NQS*** | Network Queuing System |
| ***OSI*** | Open Systems Interconnections |
| ***POSIX*** | Portable Operating System Interface for X |
| ***PWS*** | Programmable Workstation |
| ***RDBMS*** | Relational Database Management System |
| ***RISC*** | Reduced Instruction Set Computers |
| ***RYO*** | Roll Your Own |
| ***RPC*** | Remote Procedure Call |
| ***RUOW*** | Remote Unit of Work |
| ***SCSI*** | Small Computer System Interface |
| ***SCV*** | Software Compatible Vendor |
| ***SMTP*** | Simple Mail Transfer Protocol |
| ***SYSPLEX*** | System Complex |
| ***SNA*** | System Network Architecture |
| ***SNMP*** | Simple Network Management Protocol |
| ***SQL*** | Structured Query Language |
| ***TCM*** | Termal Conduction Module |
| ***TCOS*** | Technical Committee on Operating Systems |
| ***TCP/IP*** | Transmission Control Protocol/Internet Protocol |
| ***VLSI*** | Very Large Scale Integration |
| ***VRM*** | Virtual Resource Manager |
| ***VTA*** | Virtual Storage Translator |
| ***WAN*** | Wide Area Network |

# Index

## C



## D

## E



## F

## H



## I



## K



## L



## M

## N



## O

## P

## Q



## R

## S

## T



## U

## V



## W



## X

# ITSO Technical Bulletin Evaluation RED000

**The Library for Systems Solutions**
**Computing Technology Reference**

**Publication No. GG24-4100-00**

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

**Please rate on a scale of 1 to 5 the subjects below.**
**(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)**

| | | | |
|---|---|---|---|
| **Overall Satisfaction** | ____ | | |
| Organization of the book | ____ | Grammar/punctuation/spelling | ____ |
| Accuracy of the information | ____ | Ease of reading and understanding | ____ |
| Relevance of the information | ____ | Ease of finding information | ____ |
| Completeness of the information | ____ | Level of technical detail | ____ |
| Value of illustrations | ____ | Print quality | ____ |

**Please answer the following questions:**

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

**Comments/Suggestions:** **( THANK YOU FOR YOUR FEEDBACK! )**

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape **Please do not staple** Fold and Tape



Cut or Fold
Along Line

Printed in U.S.A.