# Introduction to Networking Technologies

Document Number GG24-4338-00

April 1994

International Technical Support Organization
Raleigh Center

**Take Note!**

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

**First Edition (April 1994)**

This edition applies to currently existing technology available and announced as of April, 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. 985, Building 657
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

**© Copyright International Business Machines Corporation 1994. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

# Abstract

There are many different computing and networking technologies — some available today, some just now emerging, some well-proven, some quite experimental. Understanding the computing dilemma more completely involves recognizing technologies; especially since a single technology by itself seldom suffices, and instead, multiple technologies are usually necessary.

This document describes a sampling of technologies of various types, by using a tutorial approach. It compares the technologies available in the three major technology areas: application support, transport networks, and subnetworking. In addition, the applicability of these technologies within a particular situation is illustrated using a set of typical customer situations.

This document can be used by consultants and system designers to better understand, from a business and technical perspective, the options available to solve customers' networking problems.

(202 pages)

© Copyright IBM Corp. 1994

# Contents



© Copyright IBM Corp. 1994

# Figures



© Copyright IBM Corp. 1994

# Special Notices

This publication is intended to provide consultants, systems designers, and system architects information about the alternatives that are available in developing an open networking infrastructure so that they can help customers make network protocol decisions to meet their existing and growing business requirements.

The information in this publication is not intended as the specification of any programming interfaces that are provided by any products in this document. See the PUBLICATIONS section of the IBM Programming Announcement for the products in this document 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 Harbor 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.

You can reproduce a page in this document as a transparency if that page has the copyright notice on it. The copyright notice must appear on each page being reproduced.

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:

| | |
|---|---|
| Advanced Peer-to-Peer Networking | AnyNet |
| APPN | CICS |
| IBM | OfficeVision/VM |

© Copyright IBM Corp. 1994

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

| | |
|---|---|
| AppleTalk | Apple Corporation |
| Banyan Vines | Banyan Systems, Incorporated |
| cisco | CISCO Systems, Incorporated |
| cc:Mail | cc:Mail, Incorporated |
| DECnet, All-in-One | Digital Equipment Corporation |
| EDA/SQL | Information Builder, Incorporated |
| Ethernet | Xerox Corporation |
| IDNX | Network Equipment Technologies, Incorporated |
| IPX, NetWare | Novell Corporation |
| UNIX | UNIX Systems Laboratories, Incorporated |
| X/Open | X/Open Company, Ltd. |

# Preface

This book provides computer consultants, system designers, system architects, networking specialists, marketing representatives, trade press reporters, and others information about some of today's existing and emerging networking technologies.

Since there are many considerations that must be kept in mind as as one evaluates alternative solutions to complex connectivity and interoperability problems (and since there are many technologies), this book:

- First *identifies* the technologies by categories
- Then *describes* the operating characteristics of each technology
- Finally *compares* the technologies in a representative set of typical customer situations, including the advantages and disadvantages of each of the technologies in a particular situation

The extensive index makes this a handy reference book, as well as a general information book.

## How This Book is Organized

The book is organized as follows:

- Chapter 1, “Fundamental Concepts of Technologies”

  This chapter discusses (in a general manner) some computing/networking models, their components, and some relationships among these models and components. It also identifies networking technologies, by groups and model layers.

- Chapter 2, “Positioning and Usage of Technologies”

  This chapter describes existing and emerging technologies and points out their inherent advantages and disadvantages.

- Chapter 3, “Typical Customer Situations”

  This chapter evaluates, through typical customer situations, the use of various existing and emerging technologies.

- Appendix A, “Master Foil Set”

  This appendix contains all of the figures within this book in large format so they can be used in presentations.

## Related Publications

The following publications offer more information about the topics discussed in this book:

IBM publications:

- *Networking Blueprint: Executive Overview*, IBM order no. GC31-7057.
- *The Networking Blueprint*, (a fanfold card), IBM order no. SX33-6090.
- *Open Blueprint Introduction*, a white paper, April 1994.

© Copyright IBM Corp. 1994

Other publications:

- Cypser, Rudy, *Communications for Cooperating Systems: OSI, SNA, and TCP/IP*, The Systems Programmer Series, Addison-Wesley Publishing Company, 1991, ISBN #0-201-50775-7.
- Halsall, Fred, *Data Communications, Computer Networks and OSI*, Addison-Wesley Publishing Company, 1988, ISBN #0-201-18244-0.
- IEEE, *Guide to POSIX Open Systems Environment*, 1992.
- Tanenbaum, Andrew S., *Computer Networks.*, Prentice-Hall, Inc., 1981, ISBN #0-13-165183-8.

## 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 Organization 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 ITSO Technical Bulletins (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 IBM branch office.

You may order individual books, CD-ROM collections, or customized sets, called GBOFs, which relate to specific functions of interest to you.

## Acknowledgments

The advisor for this project was:

Loretta Mirelli
International Technical Support Organization, Raleigh Center

The authors of this document are:

Burnie Blakeley
IBM Networking Systems

Deborah J. Boyd
IBM U.S. Federal

Steve Smith
IBM Networking Systems

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

Thanks to Diane Pozefsky, IBM Networking Systems, for her invaluable advice and guidance in the production of this document.

Thanks to the following people for reviewing the document and providing invaluable feedback:

Robert Louden
Marketing and Services, RTP

John Maas
TCP/IP Strategy, RTP

Kathleen Riordan
AnyNet Planning, Strategy and Marketing Support, RTP

Thanks also to the following people for their invaluable assistance in editorial and graphics support:

Peter Andrews
Gray Heffner
Shawn Walsh
Gail Wojton
Janet Yoho
International Technical Support Organization, Raleigh Center

# Chapter 1. Fundamental Concepts of Technologies

The objectives of this chapter are to:

- Discuss, in a general manner, some computing/networking models, their components, and some relationships among these models and components
- Identify networking technologies by groups and model layers

## 1.1 Introduction

There are many different computing and networking technologies — some available today, some just now emerging; some well-proven, some quite experimental. Understanding and solving today's computing dilemma more completely involves recognizing technologies; especially since a single technology by itself seldom suffices and, instead, multiple technologies are usually necessary. Some technologies are being obsoleted, some are maturing, some are adequate, and some are vital.

A single and simple frame of reference is most helpful in understanding the concepts of individual networking technologies, seeing how they operate, and recognizing relationships among technologies. The various technologies share many fundamental concepts.

This chapter provides an introduction to the world of networking technologies. It establishes a very generalized reference model, and then classifies technologies into categories relative to this model.

This chapter provides introductory information to better understand the technologies defined in Chapter 2, “Positioning and Usage of Technologies” on page 47. If you are already familiar with reference models and wish to skip our discussion of them, we suggest you quickly examine Figure 6 on page 12 to discover our layer labeling conventions and then continue reading with Chapter 2, “Positioning and Usage of Technologies” on page 47.

© Copyright IBM Corp. 1994

## 1.2 A Reference Model

A complete and generalized computing reference model is quite helpful in describing different technologies and their relationships.

Many different groups in the computing industry have been involved during the last decade in developing computing reference models — some models for operating systems, some for data bases, some for application systems, and some for communications networking — but only recently have efforts begun in earnest to combine these various models into a single, more complete, but yet simpler reference model. Such a generalized model can be based easily upon established networking models.

There are, indeed, many different technologies available in today's marketplace that provide solutions for a variety of networking problems; but, to comprehend how these technologies function, one must start with a reference model. One of the best reference models that exists today for networking is the *OSI Reference Model*. We will utilize this reference model for all discussion throughout this book in a slightly simplified format.

Figure 1 relates the terminology used in this book to an approximation of the terminology used in the Open Systems Interconnection (OSI) model[1] created by the International Standards Organization (ISO).

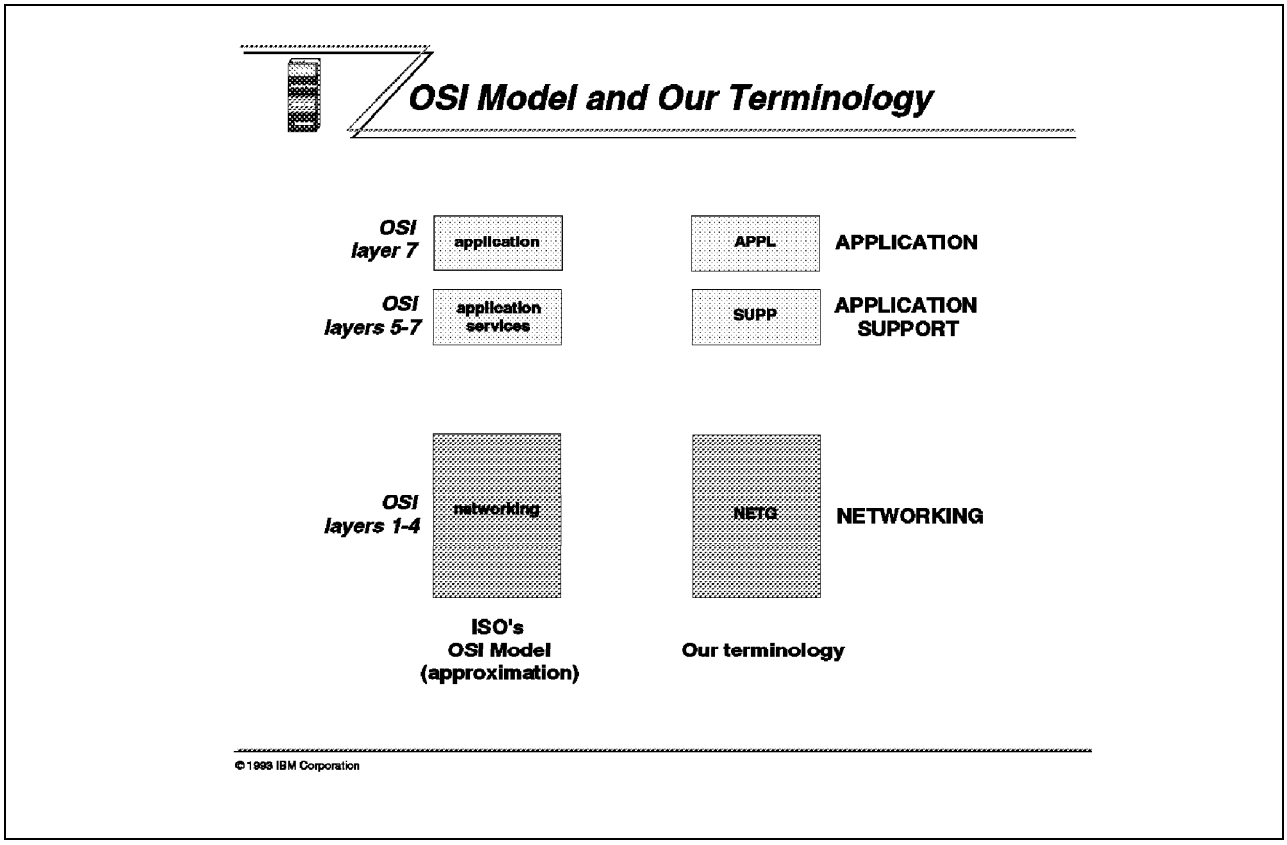



*Figure 1. OSI Model (Approximation) and Our Terminology*

[1] The OSI model is called simply a model and not a networking model.

Referring to the right side of the above diagram, the *application* programs (the APPL box in the diagram) conditionally depend upon the *application support* programs (the SUPP box in the diagram), which in turn conditionally depend upon the *networking* programs (the NETG box in the diagram). That is, sometime during their execution, application programs may require application support programs (or in some cases they may not). Likewise, sometime during their execution, application support programs may require networking programs (or in some cases they may not).

Referring to the left side of the above diagram, and speaking loosely, the "bottom" (4 layers) of the OSI model[2] accomplishes networking; the "top" (3 layers) of the OSI model accomplishes application and application services. In the pure OSI networking model, application services are defined to be entirely within layer 7 (at the bottom of the application layer). We have stretched the OSI application services definition to include those networking middleware services provided by OSI layers 5 and 6.

Relative to networking, the simple model shown in Figure 1 on page 2 is extremely similar to the more detailed IEEE POSIX Open System Environment (OSE) model[3] , as shown on the extreme left side of Figure 2 on page 4, using just the 'communication entities' (our networking), 'application platform entities' (our application support), and 'application entities' (our application) parts of the OSE model.

---

[2] Strictly speaking, there is no bottom and top defined for the OSI model. We have coined these terms for our own convenience in comparing terminologies.

[3] You will find the POSIX-OSE model described in the P1003.0/D16.1 draft dated October 1993, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society. Many other computing models are also similar to the POSIX OSE model, but, when simplified, most models reduce to our simple model with respect to networking and ignoring the people and information exchange entities represented in the complete OSE model.

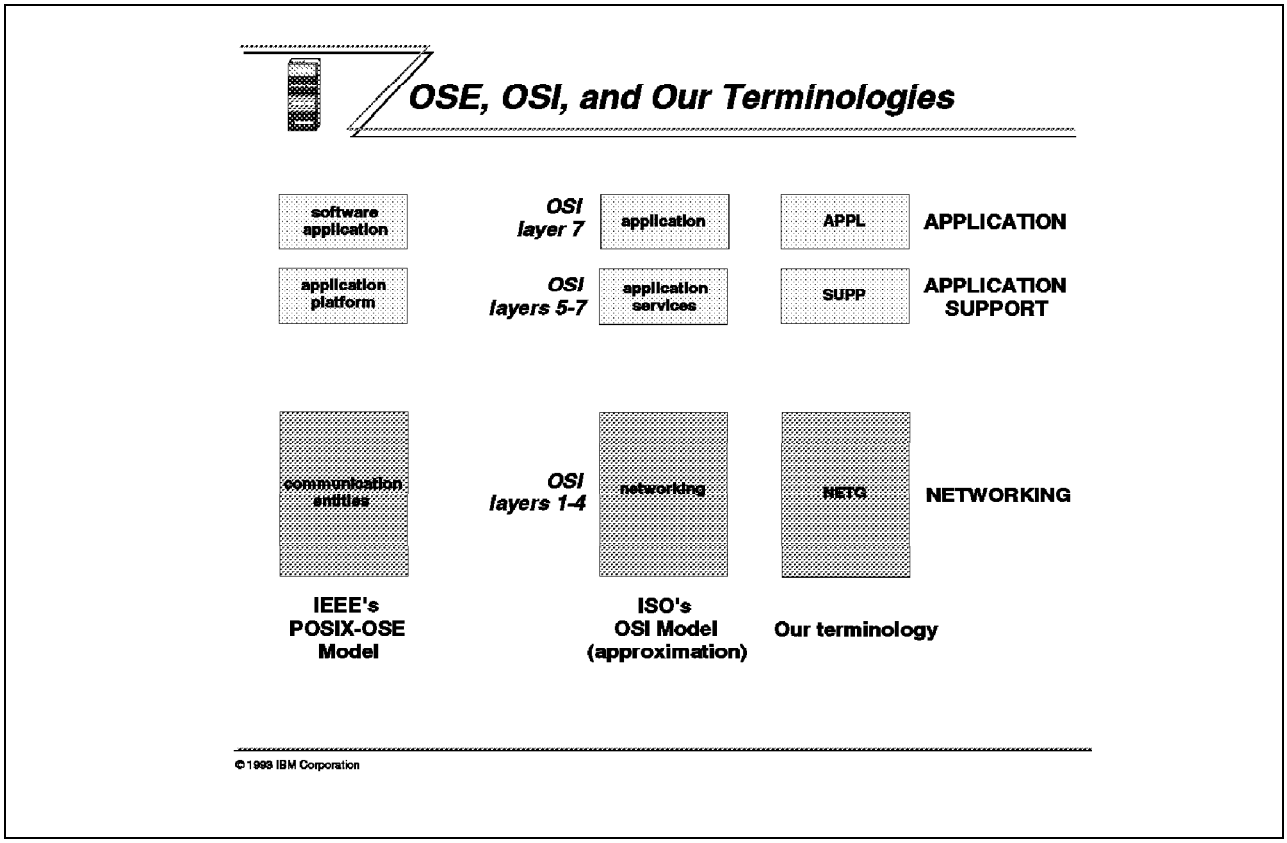



*Figure 2. OSE, OSI, and Our Terminologies*

The OSE model (see the extreme left of Figure 2) —— although not, strictly speaking, a layered model —— shows *communication entities (or networking)* programs being separated from *software application* programs by *application platform* programs. We have labeled this separator software as simply *application support* (SUPP) in our simple reference model. Other models, similar to OSI (an older model) and OSE (a newer model), have similar labeling.

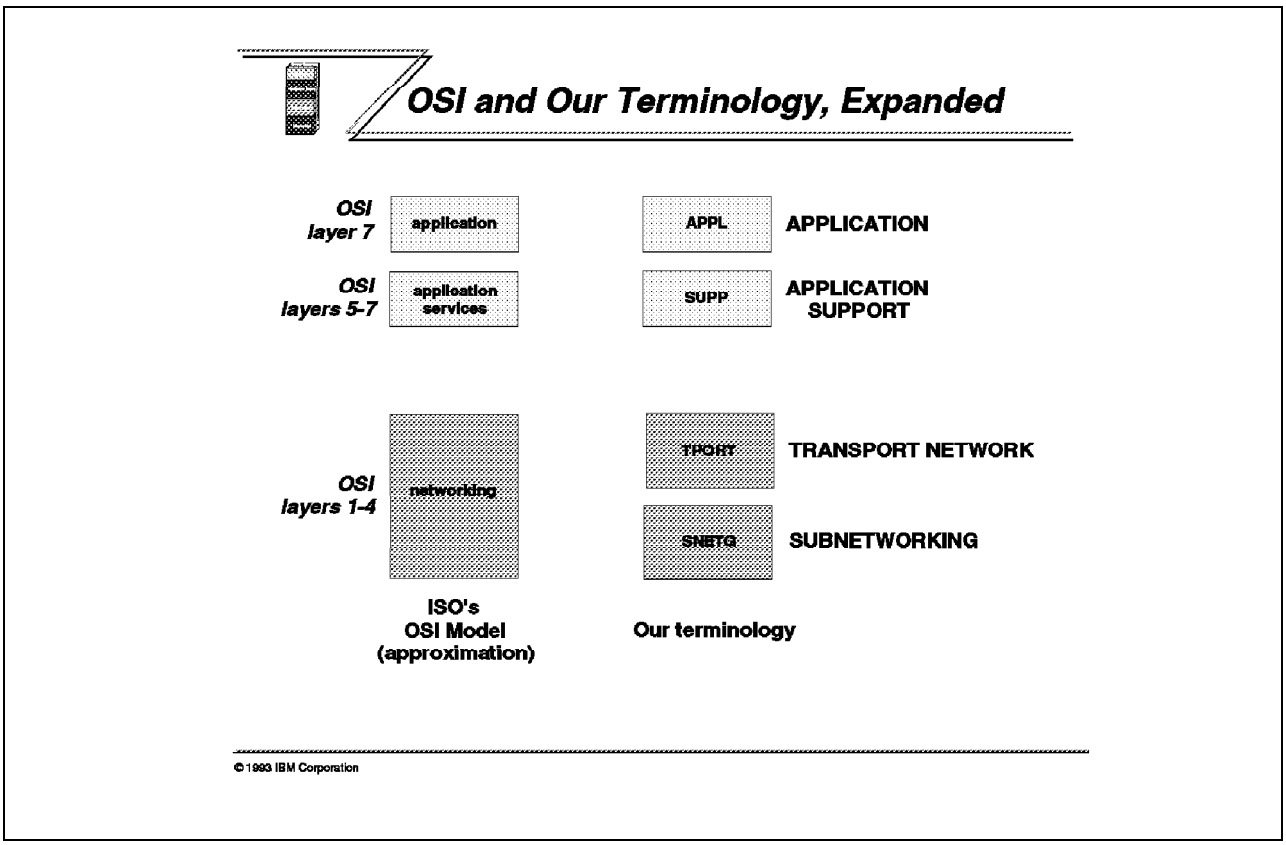



*Figure 3. OSI Model (Approximation) and Our Terminology, Expanded*

The OSI networking layers (layers 1 through 4) can be subdivided, approximately in half, into transport networking (layers 3 and 4) and subnetworking (layers 1 and 2), as shown in Figure 3, thus making four layers instead of three in our simple model. These four layers (or software program groups) provide the basis for the discussions in the remainder of this book. The “bottom” two layers can always be regarded as networking[4]. The “top” two layers can always be regarded as the application environment.

Application and application support (boxes) constitute the “top” of the model; transport network and subnetworking (boxes) constitute the “bottom” of the model.

[4] The sub-equation 'NETG = TPORT + SNETG' causes the original three layers (boxes) to be expanded into four layers (boxes) in our simple model.

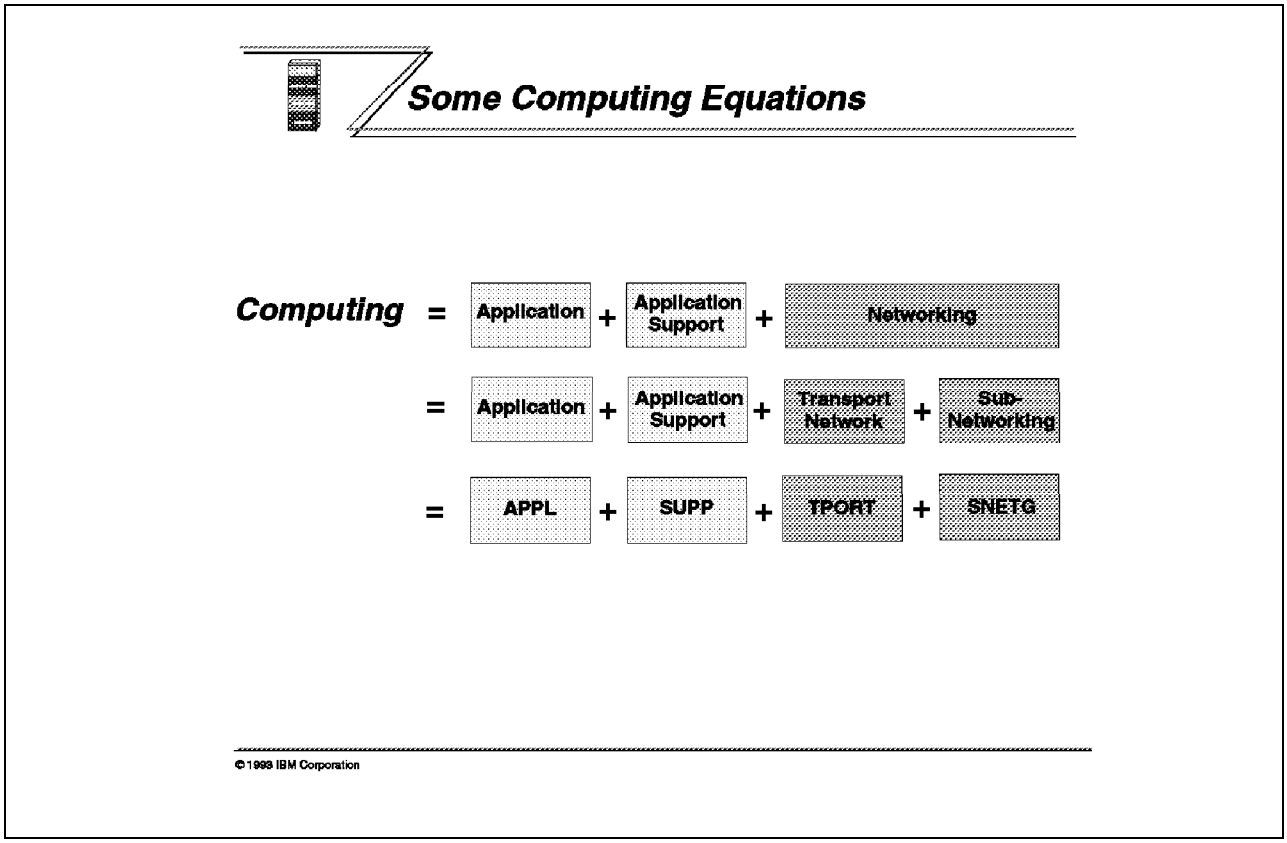



*Figure 4. Some Computing Equations*

Figure 4 identifies the useful equation used throughout the remainder of this book:

COMPUTING = APPL + SUPP + TPORT + SNETG

where each of the four layers (software program groups) interacts only with its adjacent layers (above and below).

The diagrams throughout this book use the short labels APPL, SUPP, TPORT, and SNETG for the four layers of our computing model. Collectively, the four layers constitute computing[5] , as the equation reflects. In our diagrams, the top two layers are closely associated with one another, as are the bottom two layers.

All discussion is from the perspective of a particular application's required application support and communications infrastructure. Briefly, the four major groupings identified above are as follows:

- **Application** is abbreviated *APPL-A* for a particular application 'A' under consideration.

  Note that this application can be user-developed to satisfy a particular business requirement, or it can be a more generic, prepackaged,

[5] Computing can be accomplished at various points in time by using either all of the elements on the right side of the computing equation or just some of them. That is, computing involves, at various times, just applications (APPL) , or applications and application support (APPL and SUPP), or applications and application support and networking (APPL, SUPP, TPORT and SNETG).

off-the-shelf purchased application, such as an X.400 mail application or an X.500 directory server.

- **Application Support** has two divisions, called the *Application Programming Interface* (abbreviated *API-A* for a particular application 'A'), and *Application Support* (abbreviated *SUPP-A*). The API and the application support are closely tied together, and are chosen by the application programmer based upon the requirements of the application.

  Examples of application programming interfaces include CPI-C (Common Programming Interface for Communications), RPC (Remote Procedure Call), and MQI (Message Queue Interface). Depending upon the API selected, the application services may be quite different. For instance, CPI-C utilizes Advanced Program-to-Program Communication (APPC) and SNA logical unit 6.2 (LU 6.2) services, which includes the protocol flows between two applications for establishing a conversation, exchanging data, ensuring commitment of resources, and terminating a conversation. RPC does networking through program stubs that are customized for each application program and then attached (linked). RPC usually operates over TCP/IP protocols. MQI provides queue-to-queue communication, in lieu of direct program-to-program communication over a dedicated logical connection; it is a form of networking middleware with resource commitment and assured message delivery. MQI operates over LU 6.2, TCP/IP, and other networking protocols.

- **Transport Network**, which corresponds to the critical Transport and Network OSI layers, is abbreviated *TPORT-A* for a particular application 'A.' These two layers work closely together to ensure that user data is transmitted with a predictable level of service from the source node to an end node, perhaps through a set of intermediate nodes.

  Depending upon the specific protocol chosen, these layers provide such functions as optimal route determination, error detection and recovery, and congestion control. Examples of transport protocols include TCP/IP and SNA Advanced Peer-to-Peer Networking (APPN*). Each of these protocols utilizes unique addressing structures, protocol flows between peer transport layers in end nodes, and routing protocols between intermediate nodes. Please note that throughout this book the term “transport protocol” will refer to the *combination* of these two OSI layers (unless explicitly identified as OSI layer 4) to match nomenclature commonly used in the industry.

  Also note that, historically, the Application Support and Transport Network have been very closely tied together, and the selection of a particular API forced the selection of a particular network protocol, or, conversely, a programmer was forced to select an API based on the currently supported transport protocol in the network. For instance, Remote Procedure Call (RPC) and the TCP sockets interface are closely associated with the TCP/IP transport protocol, and would be the application programming interface of choice for a TCP/IP-based transport network; however, if a CPI-C-based application might solve a particular business requirement, then SNA transport would have to be added to this TCP/IP network to support the CPI-C-based application, which might involve substantial effort.

- **Subnetworking**, abbreviated *SNETG*, corresponds to the OSI Data Link Control and Physical layers. These layers are concerned with getting data on the physical media of the network, and then getting it reliably and efficiently from one physical node to the next physical node in the network.

Examples of subnetworking include the various Local Area Network choices (such as Token Ring, Ethernet**, and FDDI) and Wide Area Network choices (such as SDLC, X.25, and Frame Relay).

This is a very brief introduction to the four major program groups of our simple model. These groups are discussed in much more detail throughout the remainder of the book.

## 1.3 Fundamental Concepts

The fundamental concepts necessary for an easy discussion of available and emerging techniques are:

- Stacks of software
- Switching points
- Model layers (or software program groups)

    Application (APPL)

    Application Support (SUPP)

    Networking (NETG)

        Transport Network (TPORT)

        Subnetworking (SNETG)

You will recognize these concepts as being related to the computing equation introduced earlier (that is, Computing = APPL + SUPP + TPORT + SNETG). The four layers are stacked, switching points exist between each layer, and each layer serves some purpose. These concepts are discussed in the following sections.

### 1.3.1 Stacks of Software

As evidenced by both the POSIX-OSE computing model and the OSI networking model[6], it is often convenient to think of groups of computer programs as being stacked[7] , one group on top of another, and that one on top of yet another.

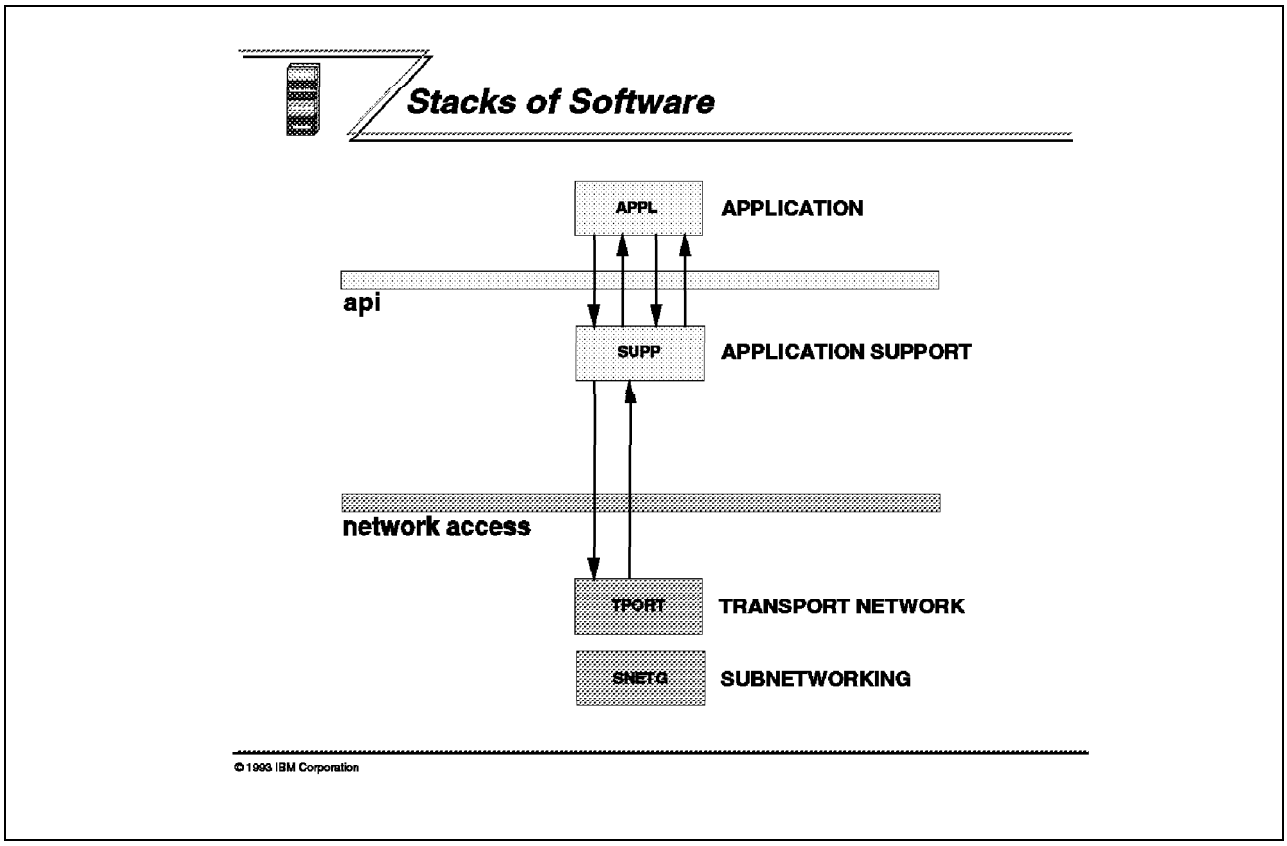



*Figure 5. Stacks of Software*

Figure 5 illustrates this stacks of software concept for the software program groups or layers:

- Application
- Application Support
- Transport Network
- Subnetworking

An application program interface (API) is a mechanism that defines interoperation between the application (APPL) and application support (SUPP) program groups.

---

6 The OSI networking model is mostly concerned with layering of software and (horizontal) interoperations between like layers and is not so much concerned with (vertical) interoperations between adjacent unlike layers, —— that is, OSI is concerned more with protocols and less with interfaces.

7 Each group of programs has its own set (or sets) of formats (information bit patterns) and protocols (information exchange rules); hence, part of or all of this stack is often referred to as a *protocol stack*, whereby a format stack is also implied. Of course, each protocol stack is composed of a particular collection of protocols stacked upon one another; not all possible protocols are included.

The API is a “contract of sorts” whereby the *application* software (above the API) requests particular functions and the *application support* software (beneath the API) supports the interface by interpreting the requests (program calls), executing them, and returning the results across the API to the application software.

The application (APPL) group of programs is often divided into particular subgroups or application sets or “application suites,” according to the purpose of each application. The application support (SUPP) group of programs is often divided into several general types of application support, according to what the support does for the application.

A network access is a mechanism that defines interoperation between the application support (SUPP) and the networking (TPORT and SNETG) program groups.

The network access operates in much the same way as the API but, instead, operates between a networking user (for example, an application support program) and a networking provider (for example, a transport network program).

The networking (TPORT and SNETG) group of programs is often divided into particular types of networking, according to the functions provided and the communications medium being used.

The various subgroups of the above main groups (layers) of programs are discussed in more detail later in this chapter.

Lots of different technologies exist for interoperations between these groups (and between subgroups within these groups). This chapter identifies many of these technologies, particularly those in the networking groups (that is, in the transport network and in the subnetworking program groups).

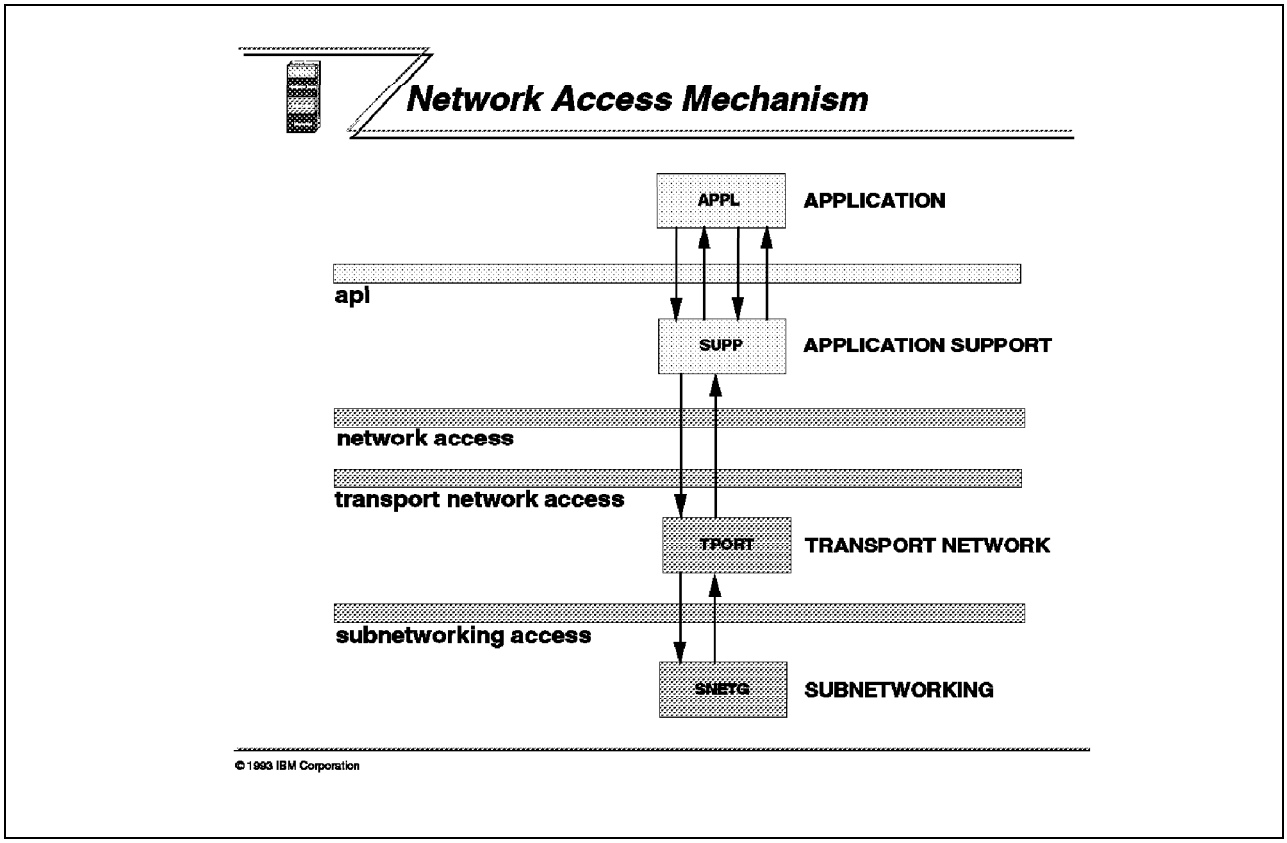



*Figure 6. Network Access Mechanism*

Together, the transport network (TPORT) and subnetworking (SNETG) constitute the networking (NETG) program group with its network access mechanism. Access to networking is almost always through the transport network so that the transport network access and the network access, as shown in the middle of Figure 6, are usually synonymous. Subnetworking is usually accessed only indirectly through a transport network.

In most of the diagrams within this book, you will see the networking group of programs replaced by two groups of programs, the transport network group and the subnetworking group, so that the stack of software will appear as in Figure 6 (that is, with the NETG box expanded into the TPORT and SNETG boxes).

### 1.3.2 Switching Points

The four program groups (or model layers) discussed previously are separated by three dividers between the four groups, as shown in Figure 7. In the best case, these group-separating dividers allow for a selection of software beneath each in such a way that each divider serves as a switching point. The API is the switching point between application and application support layers. The transport network access mechanism is the switching point between application support and transport network layers and, as such, is a system programming interface, often referred to as an SPI, as contrasted with an application programming interface in that it is most often used by those programmers responsible for systems programming.

Each software program group (layer) is a collection of programs. The programs within each group can interact, directly and indirectly, with programs in other groups above and below it. For example, in Figure 7, application program 'A7' is being supported by application support program 'S14', which in turn is supported by transport program 'T5', which in turn is supported by subnetworking program 'W14'. This might, for example, represent an invoice printing application program using the OSI FTAM program to transmit an invoice batch over a TCP/IP transport network using a LAN between the invoice printing application program and the invoice printing device.

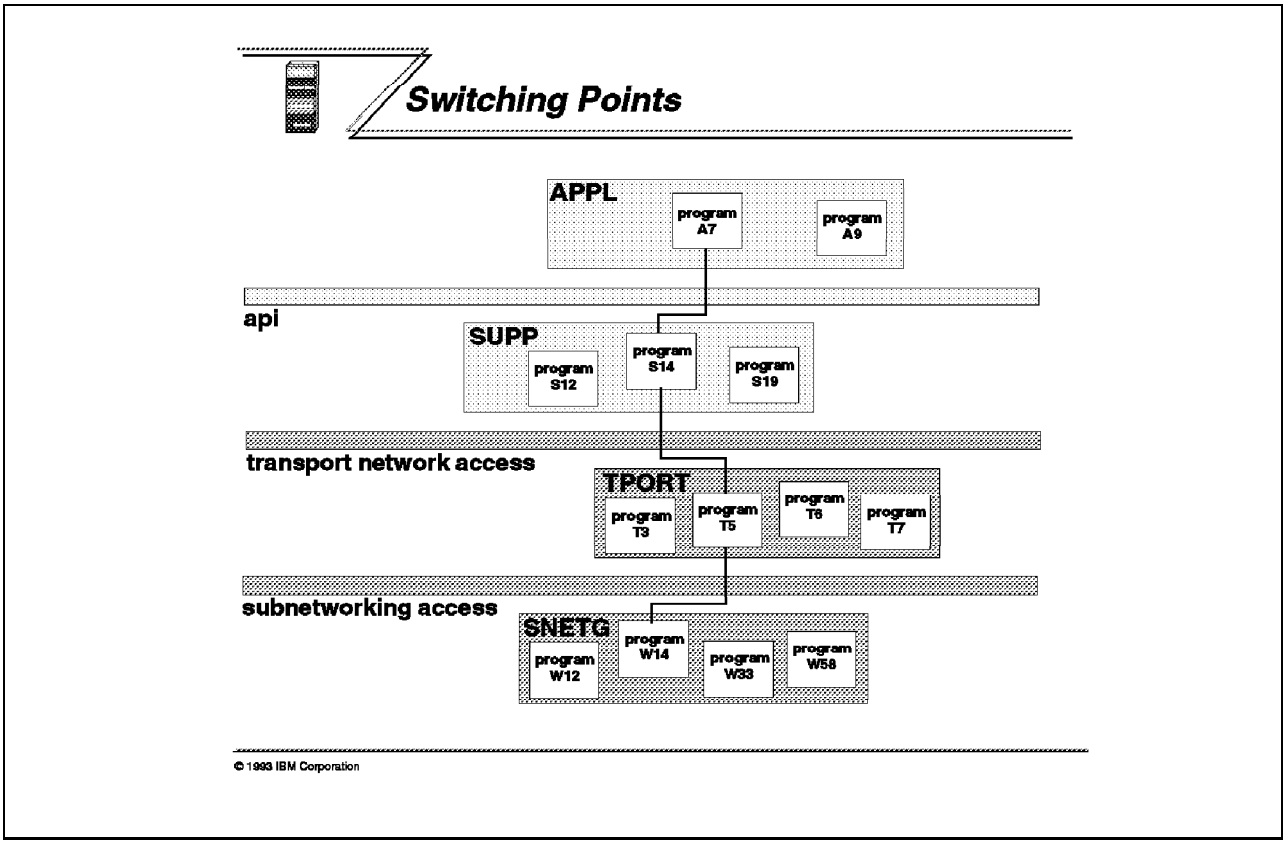



*Figure 7. Switching Points. Program Access To Services*

Because of the manner in which software has been developed (and packaged) for each of these software program groups, there are some affinities between particular types of programs within each layer. For example, a particular type of application support might be available only with a particular type of transport network, which is available with only two types of subnetworking. It is the very

rare case that all layers are found within a single software package; usually the layers are distributed across many software packages where each package may span layers or exist entirely within a layer.

The next sections discuss the four layers (application, application support, transport network, and subnetworking) of our model in more detail. Transport network (TPORT) and subnetworking (SNETG) are sometimes discussed collectively under networking (NETG).

### 1.3.3 Application Definition

A computer Application is best described as " a collection of programs (often intermixed with human and device operations) that serves some useful purpose."

For example, a credit inquiry application and a highway traffic control application are both serving a particular useful purpose (that is, getting information about credit worthiness, and controlling automobiles and trucks on a highway system, respectively).

Figure 8 illustrates the general concept of an application. A collection of computer programs does the information processing while interacting with one or more humans and devices (input devices, data storage devices, printing devices, display devices, communications devices, and such). Humans very often instigate interactions among the parts of an application ——— for example, when a human interacts with a workstation to print a report.

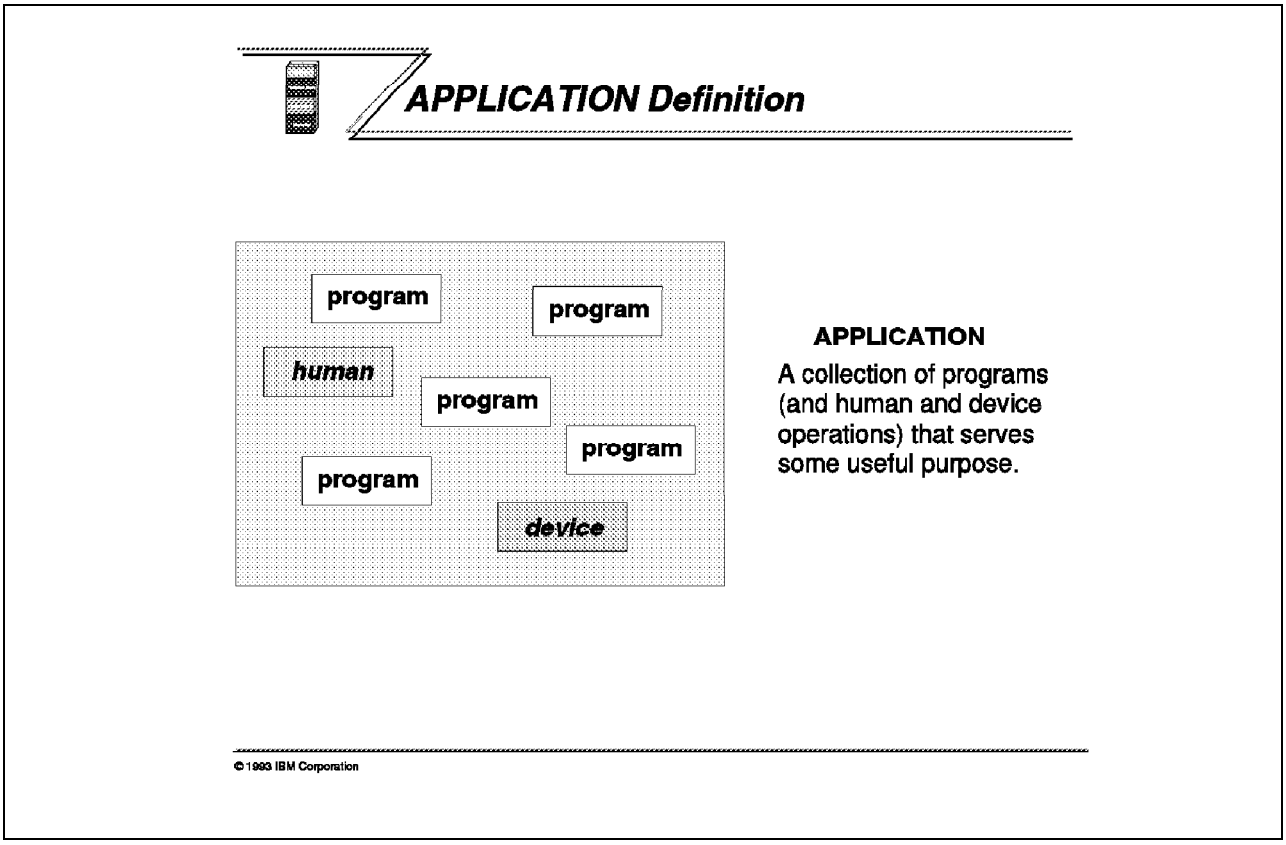



*Figure 8. Application Definition*

For our purposes within this book, an application is simply *a collection of application programs*.

### 1.3.3.1 Non-Distributed Application

A Non-Distributed Application is best described as “an application existing entirely within one location.”

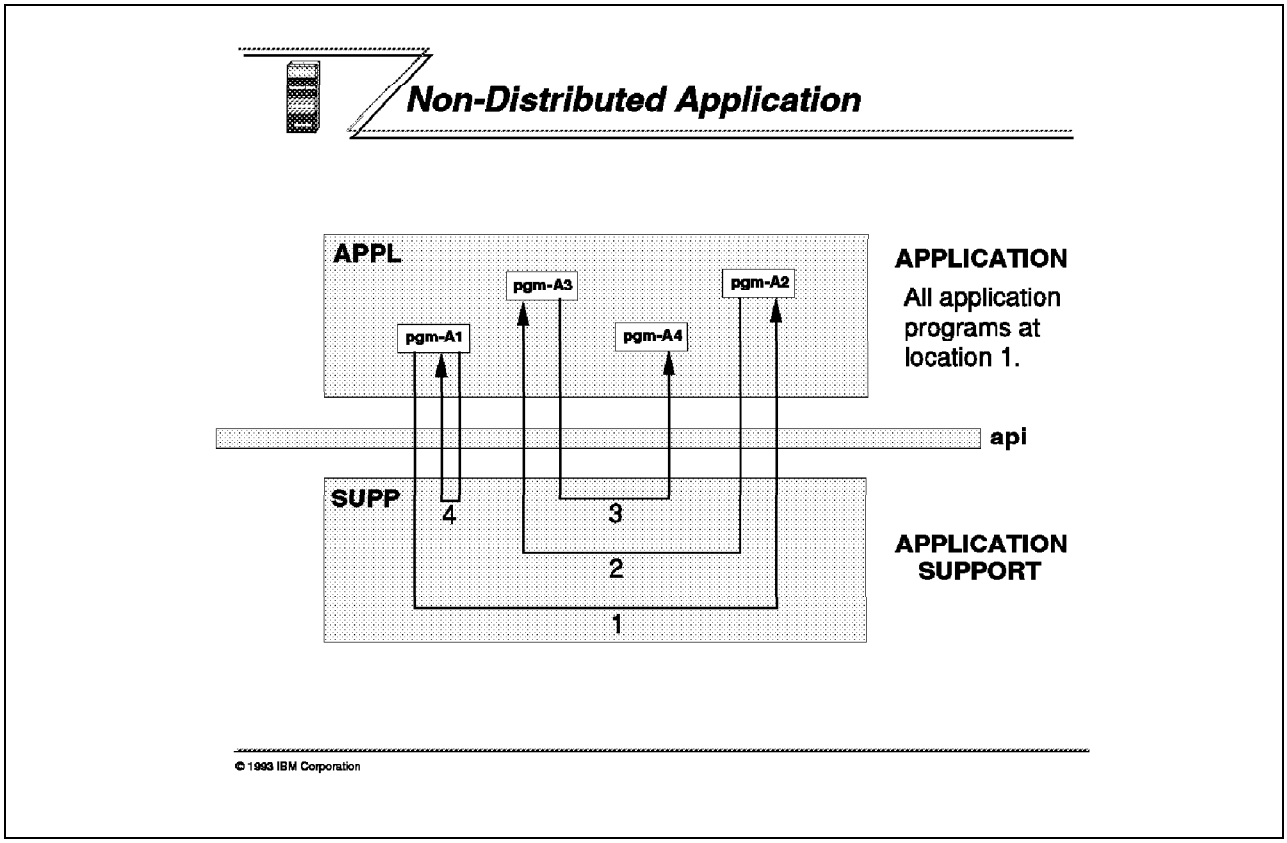



*Figure 9. Non-Distributed Application*

As shown in Figure 9, all of the programs of a non-distributed application are at the same location (for example, within one computer processor at location '1'). Similarly, all of the application support is at one location (for example, within one computer processor's software collection). No networking is required.

Interoperations can be application-to-application through application support (see flows 1, 2, and 3 at the middle left part of Figure 9), or, by contrast, simply application-to-(application support) and back (see flow 4 at the bottom of Figure 9). Of course, much of the application processing can be executed within each program.

### 1.3.3.2 Distributed Application

A Distributed Application is best described as “an application existing within two or more locations.”

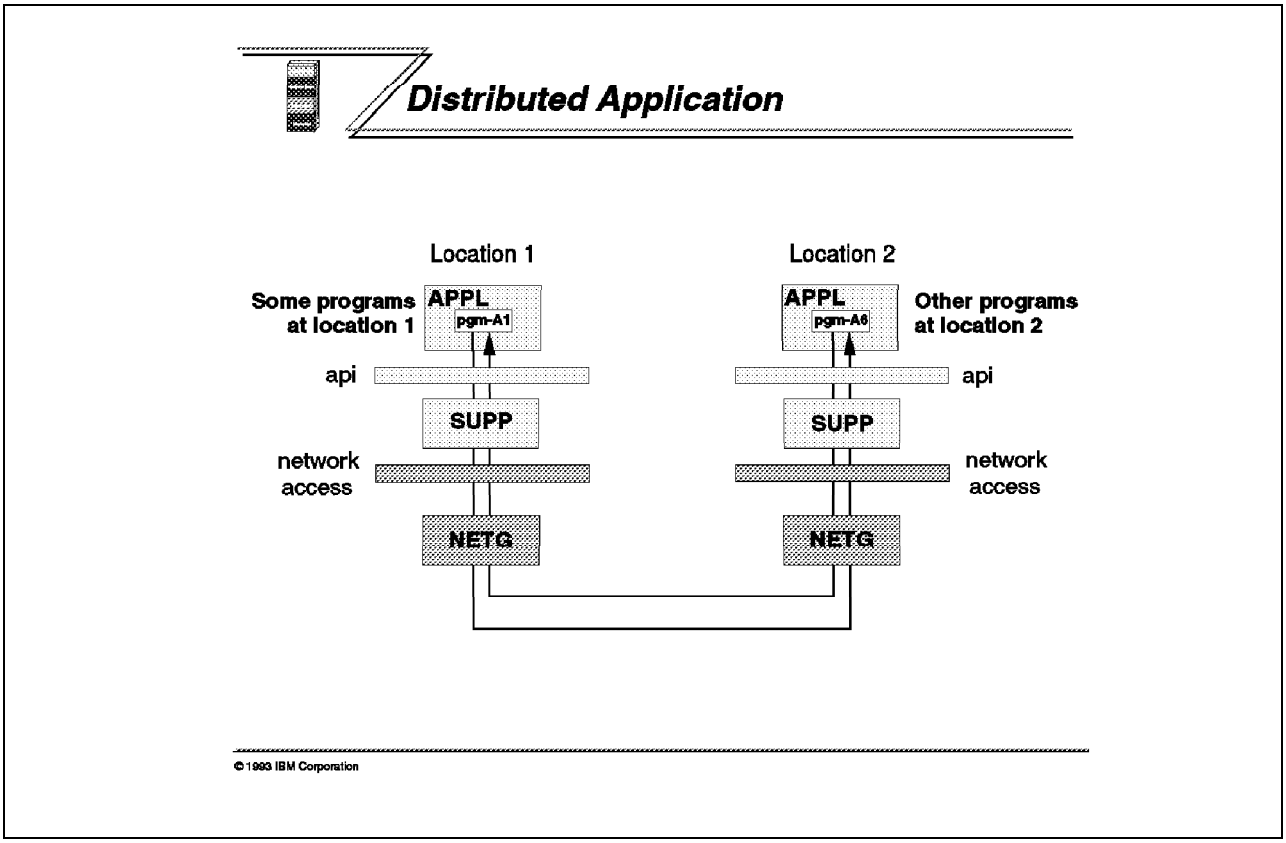



*Figure 10. Distributed Application*

As shown in Figure 10, some of the programs of a distributed application are at one location (for example, within one computer processor at location 1) while others are at another location (for example, within one computer processor at location 2). Networking is required between the locations and the network access mechanism must be used by the application support programs at each location. In this example, program 'A1' (within application 'A') is communicating with program 'A6' (also within application 'A' but at a different location). The application is distributed across locations, or is a distributed application.

In the remainder of this book, the label 'APPL-Ax' is often used in diagrams and in text as a short label for 'Application *Program* Ax'. For example, 'APPL-A4' should be read as 'application program name A4'. Similarly, 'APPL-A (without the number)' is usually meant to denote the entire application collection of programs. For example, 'APPL-A' should be read as 'Application A, consisting of many programs'.

### 1.3.3.3 Information Flow Patterns

The patterns by which information flows among application programs is interesting to study and vital to understand in order to optimize application

designs[8] . It is not just the flow of information *between* exactly two programs that is important but also the flow *among* all of the programs.

Networking technologies support information flow patterns of all sorts among application programs (and, indeed, among all kinds of programs) in different application and system environments.

In general, information is generated (created) at some point (or points) and absorbed (processed) at another point (or points). Information can take the form of a request, a response, a one-way signal, a one-way tidbit of data, a sequence of messages, and many other forms.

Patterns of flow among application programs are of basic and composite types, as shown in Figure 11.

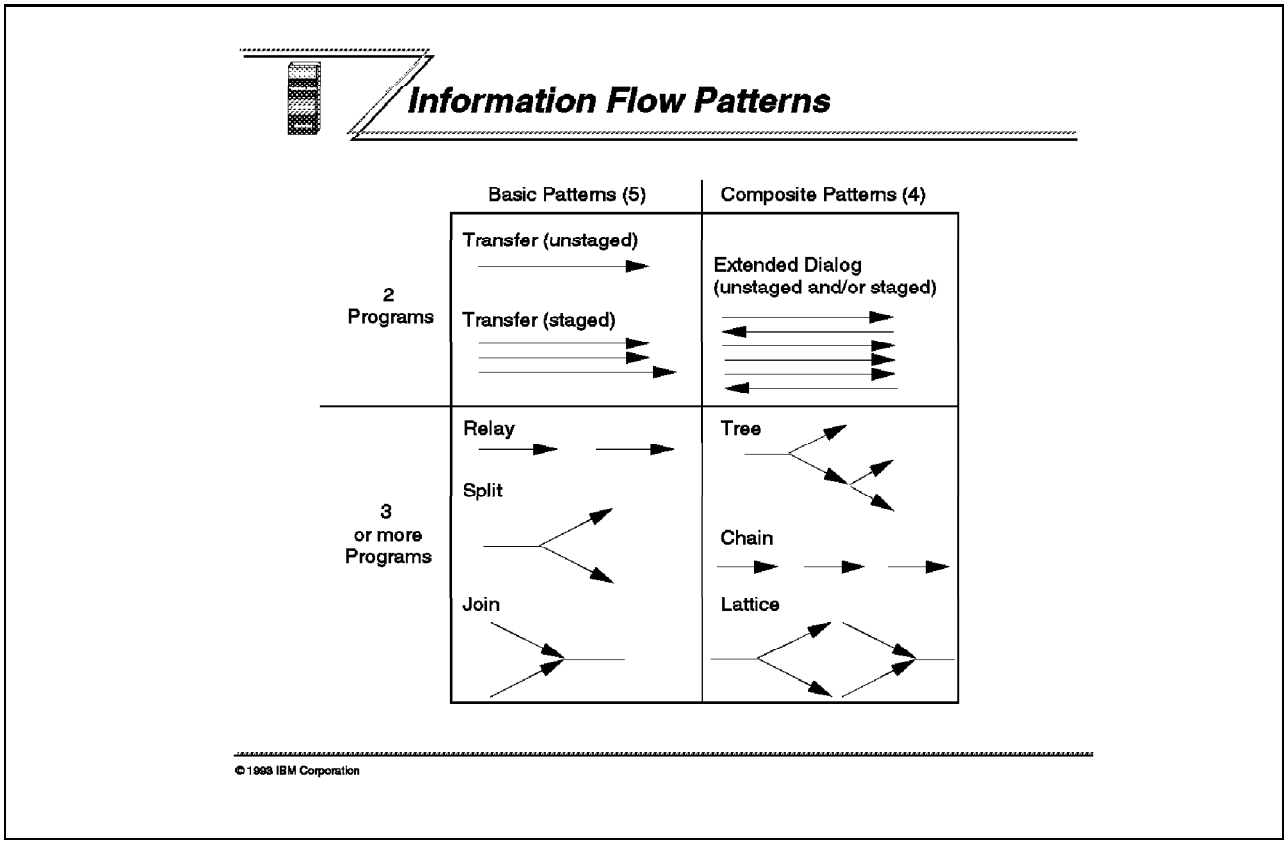



*Figure 11. Basic and Composite Information Flow Patterns*

Basic flow patterns are the simplest building blocks; composite flow patterns are aggregates of simple patterns.

In the above diagram, the five basic flows (unstaged transfer, staged transfer, relay, split, and join) are in the left column and the four composite flows (extended dialog, tree, chain, and lattice) are in the right column. For simplicity in this diagram, only one-way (non-returning) flows are shown. In the more general case, two-way (returning) flows occur in a nested fashion.

[8] A complete discussion of information flow patterns can be found in Messaging and Queuing Using the MQI, B. Blakeley, et al., McGraw-Hill Publishing Company, 1994, ISBN# 07-005730-3.

### 1.3.3.4 Basic Information Flow Patterns

Basic flow patterns involve exactly two or exactly three programs. Composite flow patterns involve two or many more programs.

The basic information flows are discussed in the following sections.

***Transfer Flow:*** A transfer of information involves exactly two programs and is accomplished either in an unstaged manner or in a staged manner. (See the upper left corner of Figure 12 on page 20.)

*Unstaged Transfer:* The simplest flow pattern is an unstaged transfer of information whereby all of the information is transferred with a single operation (request from a program). Unstaged means one-staged.

A one-way message is an example of an unstaged transfer.

*Staged Transfer:* The next simplest flow pattern is a staged transfer of information whereby all of the information is transferred only by multiple operations (requests from a program). Staged means multi-staged.

A file transfer is an example of a staged transfer.

***Relay Flow:*** A serial, unstaged transfer among three programs is a relay whenever one program transfers to a second, which transfers to a third. (See the middle left side of Figure 12 on page 20.)

***Split Flow:*** A split of information involves exactly three programs whereby information is simultaneously sent from one program to two others. (See the lower left corner of Figure 12 on page 20.)

***Join Flow:*** A join of information involves exactly three programs whereby information is simultaneously received by one program from two others. (See the lower left corner of Figure 12 on page 20.)

### 1.3.3.5 Composite Information Flow Patterns

Composite flow patterns involve two or more programs and are composed of multiple basic flow patterns in all sorts of permutations and combinations.

The three most interesting composite information flow patterns are the tree, chain, and lattice, as shown in Figure 12 on page 20.

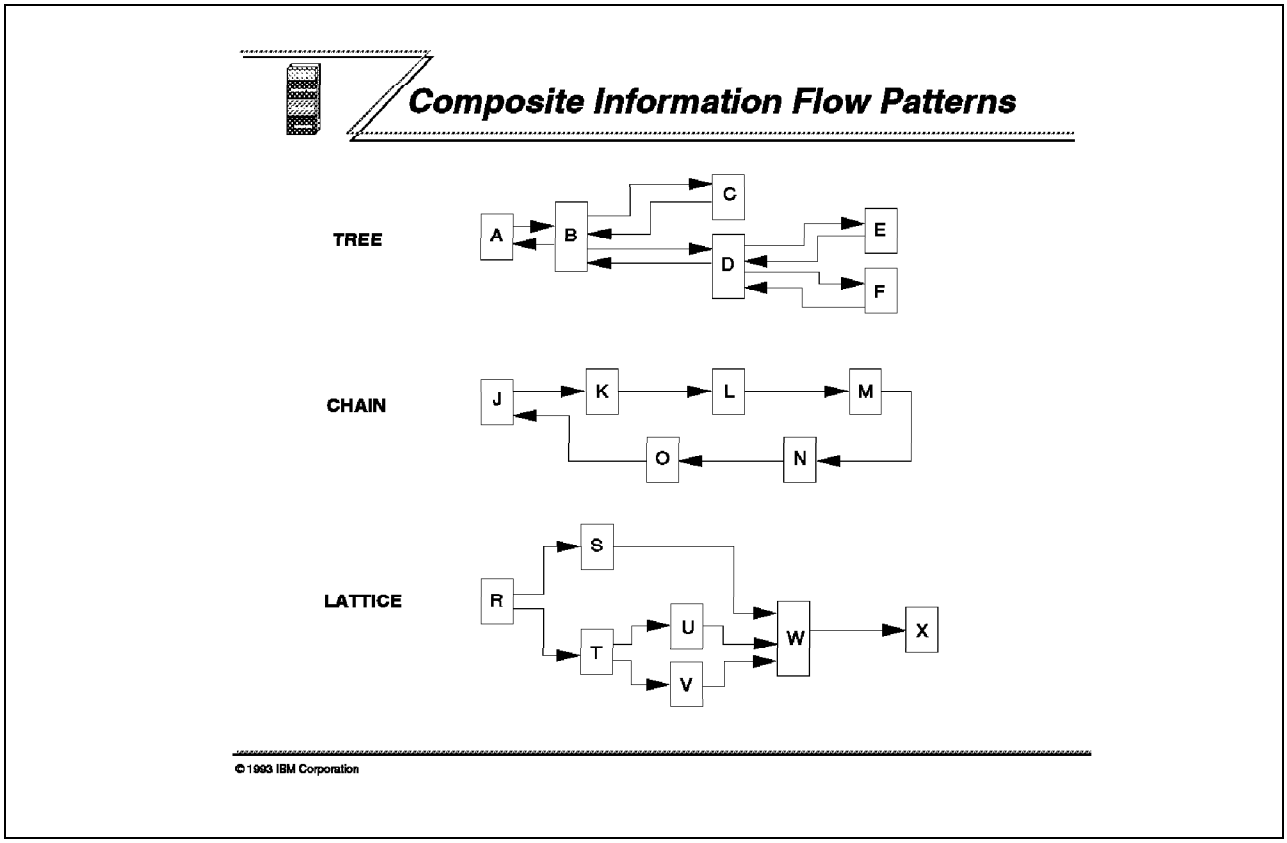



*Figure 12. Composite Information Flow Patterns*

The extended dialog is a special case of the tree or chain where only two programs are involved in an exchange of information.

The above diagram shows a returning (2-way) closed flow for the tree structure (flow pattern), a non-returning (1-way) closed flow for the chain structure, and a non-returning (1-way) open flow for the lattice. Closed flows return to the origin either by a reversal of flows or by another means (as in the chain structure shown). Open flows do not return to the origin.

The composite information flows are discussed in the following sections. Composite flows quite often involve unlike application and communication environments and depend heavily upon networking technologies to support interoperating applications.

***Dialog Flow:*** Dialog is an ISO-defined term for combinations of staged and unstaged, closed (2-way) flows between a pair of programs.

***Tree Flow:*** A **tree** structure consists of a series of split basic flows (and constitutes the classical “nested” programming structure).

***Chain Flow:*** A **chain** structure consists of a series of relay basic flows. A chain structure is much like a degenerate tree where every split has only one branch.

***Lattice Flow:*** A **lattice** structure consists of a series of split basic flows followed by join basic flows.

### 1.3.3.6 Client-Server Information Flow Patterns

The client-server flow pattern is a special case of a closed chain with exactly two programs. There are lots of implications associated with this flow pattern —— for example, configurations of small client machine and a larger server machine and configurations of multiple clients and a single server.

The special case of a 2-element closed chain is the basis for all client-server (server-requestor, requestor-responder) flow patterns.

Superimposed over a client-server configuration can be many 2-element, closed chains consisting of the client and the server where the server is a single server.

A server can play both the role of server and the role of client to other servers, in a nested fashion. In one direction the server is a server; in the other direction the server is a client. By reading the tree structure of Figure 12 on page 20 in a reverse direction from right to left, you can detect program 'D' playing the role of server to programs 'E' and 'F' while simultaneously playing the role of client to program 'B'.

One or more networking technologies are often vital in accomplishing client-server information flow patterns, especially whenever either the client or the server or both are well-established programs but in different computing environments.

### 1.3.4 Application Support Definition

Application Support is best described as “a set of software services useful to an application.”

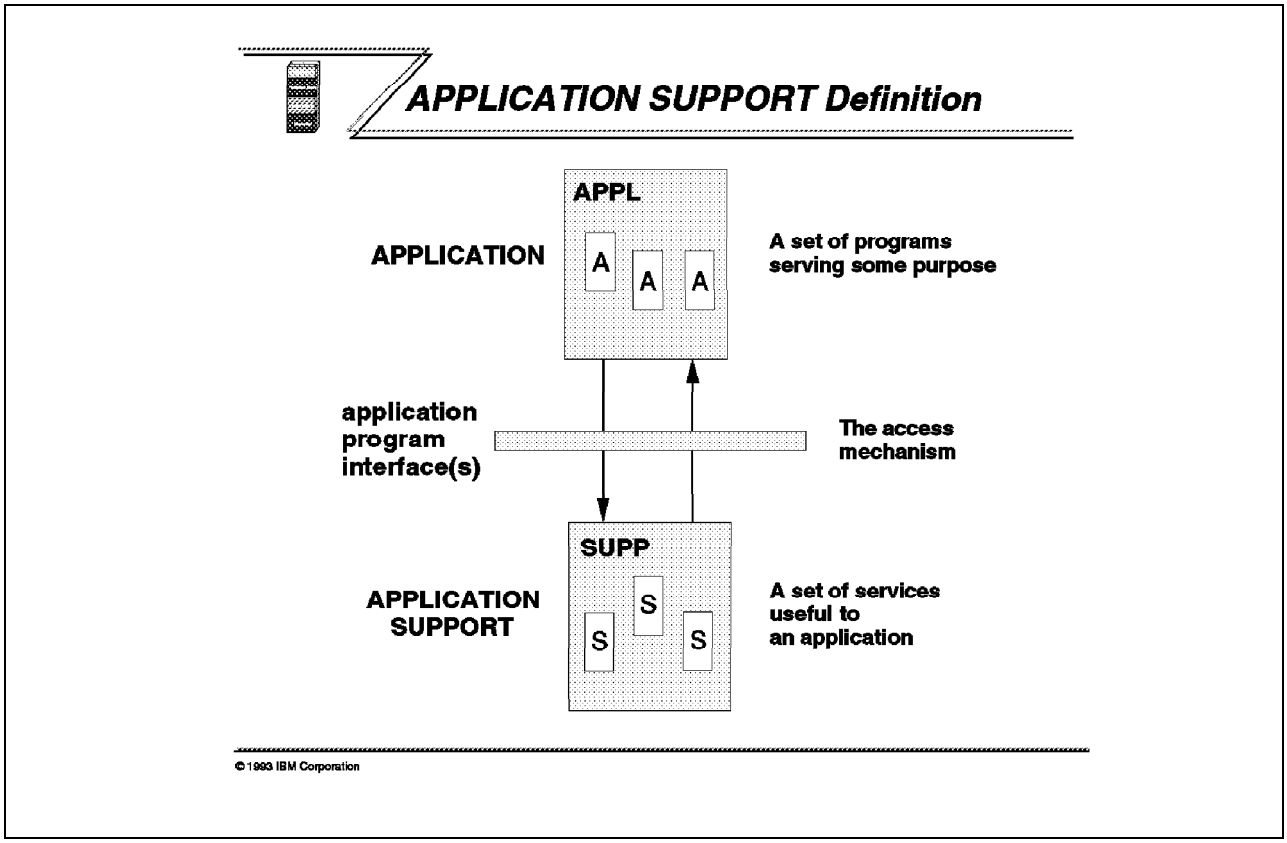



*Figure 13. Application Support Definition*

As shown in Figure 13, application support is generally composed of multiple software packages, each offering a particular set of helpful functions. Application support comprises *helper programs* containing functions (grouped together into services) that can be used by ordinary application programs. Application support is simultaneously available to multiple application programs and saves each program from having to support itself. Individual application programs access functions within services within application support through one or more application program interfaces (that is, interface to application support), commonly called APIs.[9] Application support is usually packaged so that multiple APIs, one or more for each package, are required to access all of the functions available within the total aggregate of application support.

[9] In a reciprocal manner, application support programs are accessing individual application programs through the same API.

### 1.3.4.1 Types of Application Support

Among the many types of application support (SUPP) software are those types of a more-global, less-local nature, such as:

- Communications (that is, networking),
- Standard Applications (that is, industry utilities), and
- Distributed Services (that is, sets of common application functions needed for distributed applications),

which are shown in Figure 14 and discussed in this chapter, and those of a less-global, more-local[10] nature, such as:

- Presentation Services (that is, human/device interfaces),
- Data Access Services (that is, information storage/retrieval methods), and
- Application Services (that is, application control functions),

which are not discussed in this chapter.

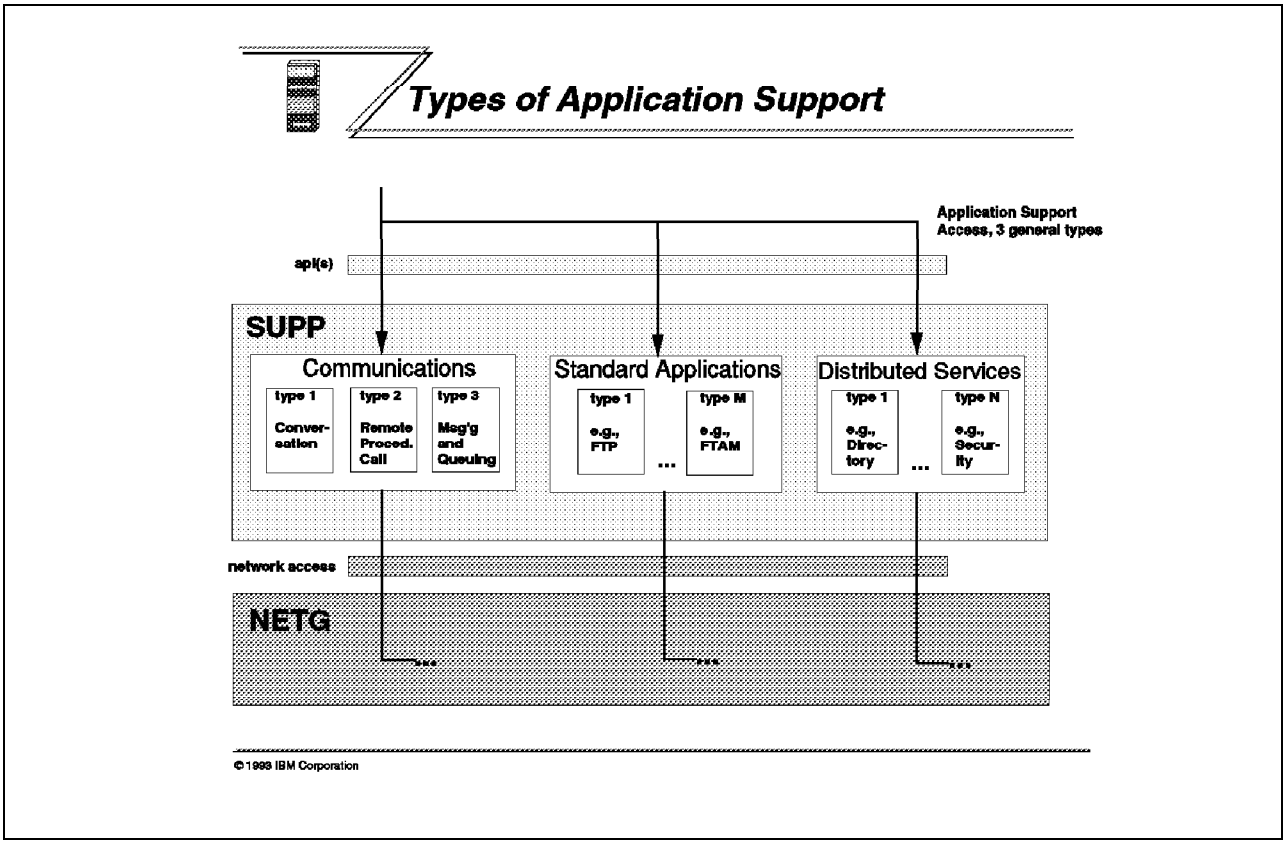



*Figure 14. Some Types of Application Support*

Individual application programs almost never need all of the application support available; most application programs use application support selectively, and repetitively; and some application programs never need any of it. The following sections describe the more-global (more network-oriented) application support identified above.

[10] Many of these "local" services can be modified by way of special techniques (for example, the interception of API Calls) to transform them into actual distributed services (for example, remote file access).

### 1.3.4.2 Communications Support

Communications support programs support communication within and between applications[11] .

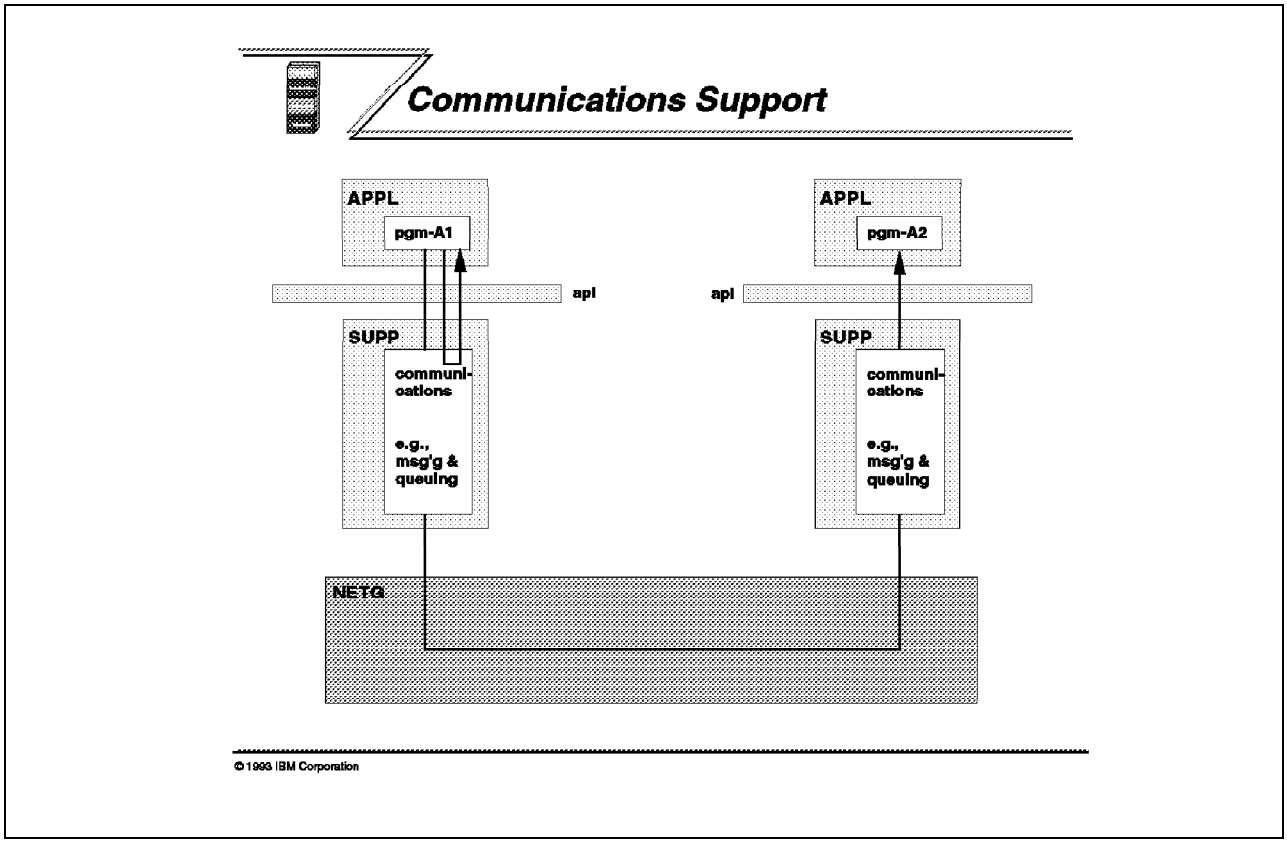



*Figure 15. Communications Support*

The communications support programs provide application-to-application (commonly called “program-to-program” or “queue-to-queue”) communication. There are many styles of communications support providing differing functions and differing degrees of support. The example above shows queue-based application support, which differs from connection-based application support. Both queue-based and connection-based application support, however, use networking for intercommunication.

[11] Any program of any type can use the communications support programs to provide its services in a distributed fashion. Distributed file access and distributed directory support are examples of non-applications that also need this communications support.

### 1.3.4.3 Standard Applications Support

Standard application programs support communication within and between applications or between end users; for example, e-mail between humans (X.400 or SMTP) or file transfer (FTAM or FTP) between systems.

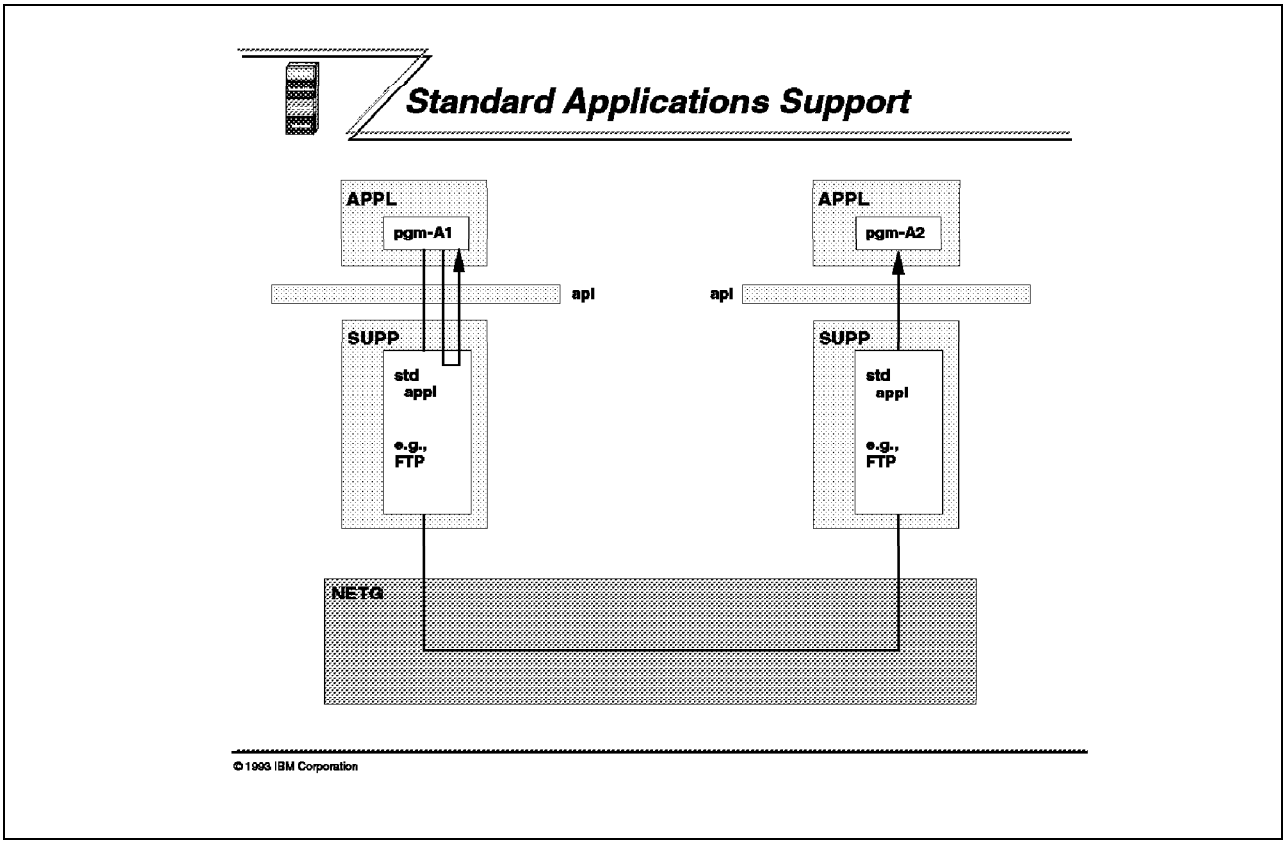



*Figure 16. Standard-Applications Support*

These are applications that have become “standardized” through constant and heavy usage over a long period of time by large numbers of organizations. This program group essentially constitutes a tried-and-tested set of industry utility programs for moving information (for example, large files) across networks.

### 1.3.4.4 Distributed Services Support

Distributed services programs provide ancillary support for communication within and between applications. These services are typically used selectively (for example, directory or security) or as an enhancement to normal application processing (for example, transaction management or resource recovery).

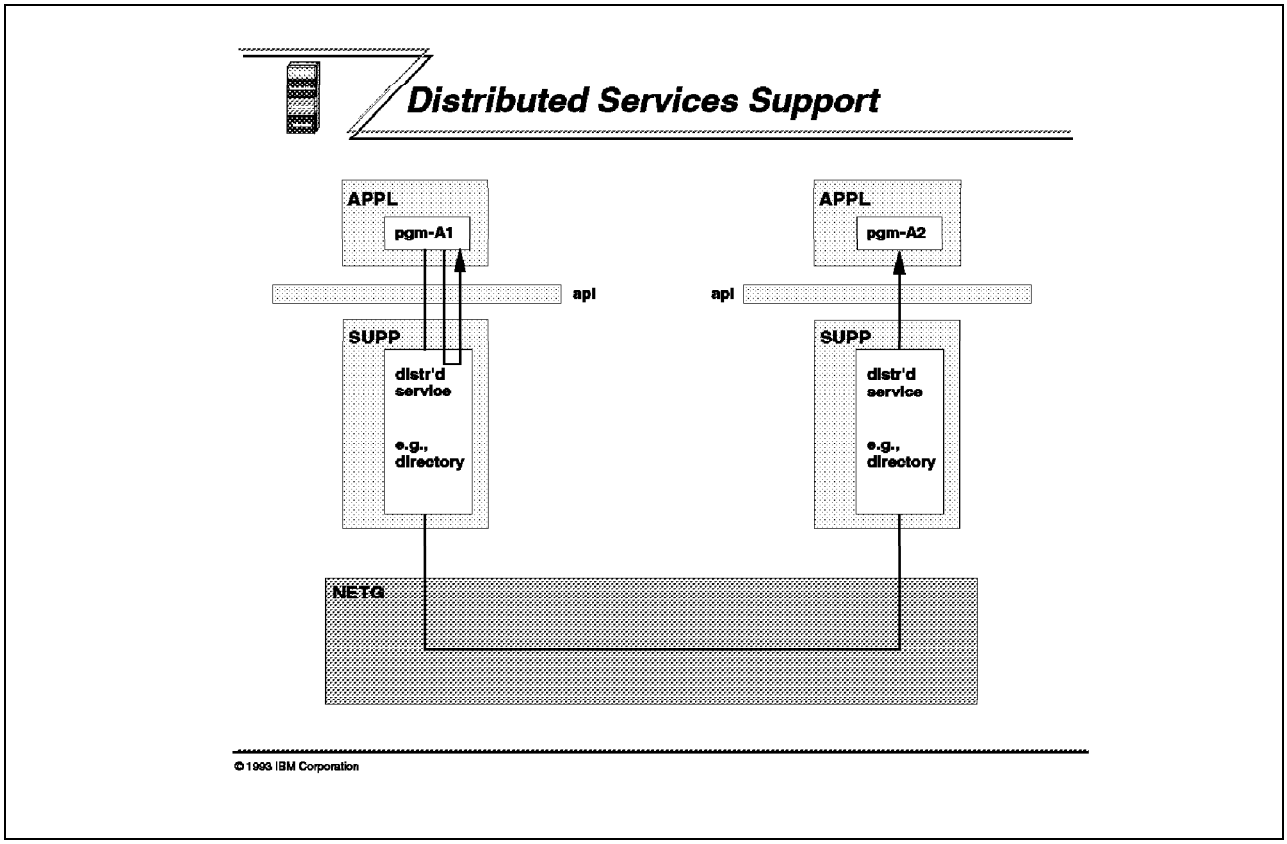



*Figure 17. Distributed-Services Support*

Distributed services are, for the most part, in a constant state of evolution in the computer industry. There are no uniform sets of distributed services available yet, except for such consortium offerings as OSF-DCE.

## 1.3.5 Networking Definition

Networking is best described as “a set of software services accomplishing communication between computer systems.”

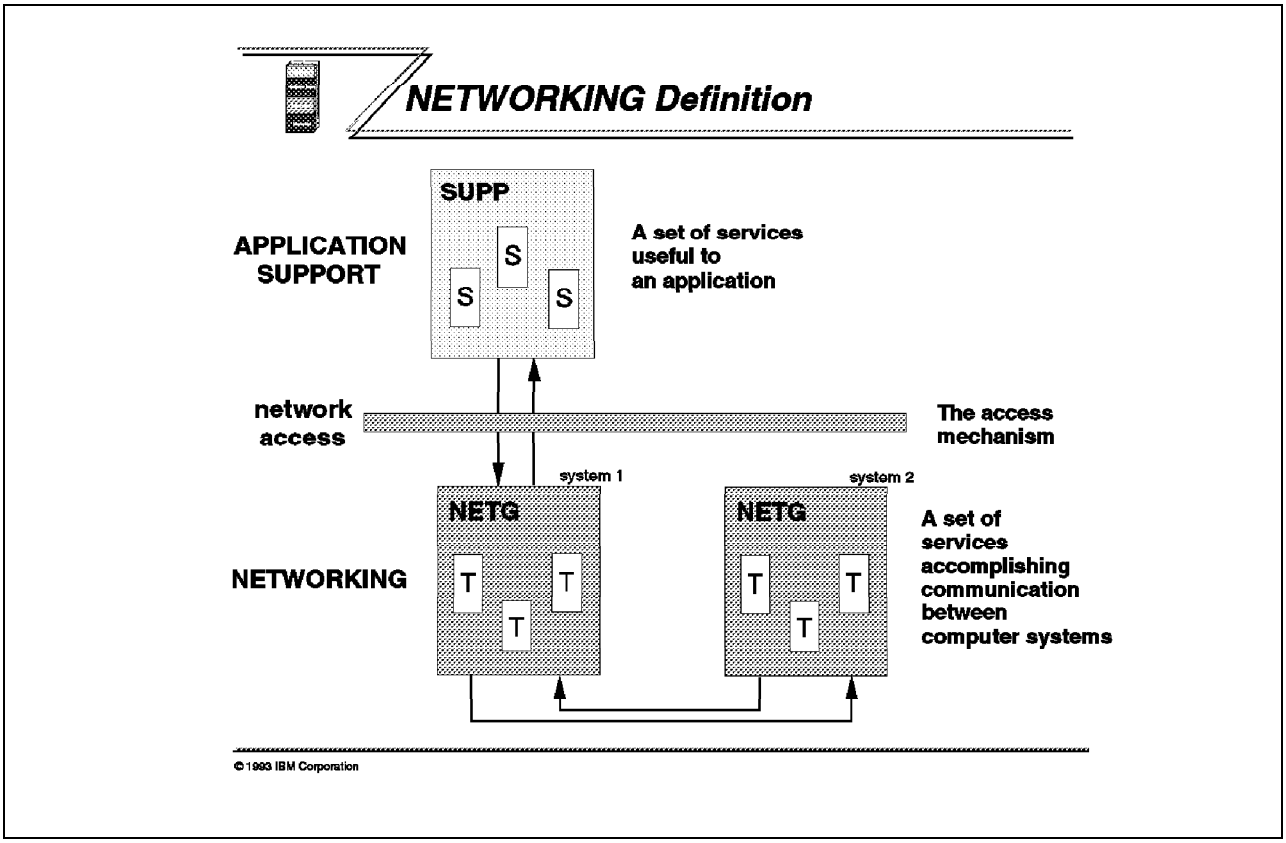



*Figure 18. Networking Definition*

Individual application support programs access functions within services within networking through the network access mechanism or set of mechanisms. A particular program 'S' from within the application support group selects a particular program 'T' from within the networking group. Program 'T' *provides* the networking; program 'S' *uses* the networking.

### 1.3.5.1 Basic Networking Componentry

As shown in Figure 19, Networking (NETG) is composed of two parts:

- Transport Network (TPORT)
- Subnetworking (SNETG)

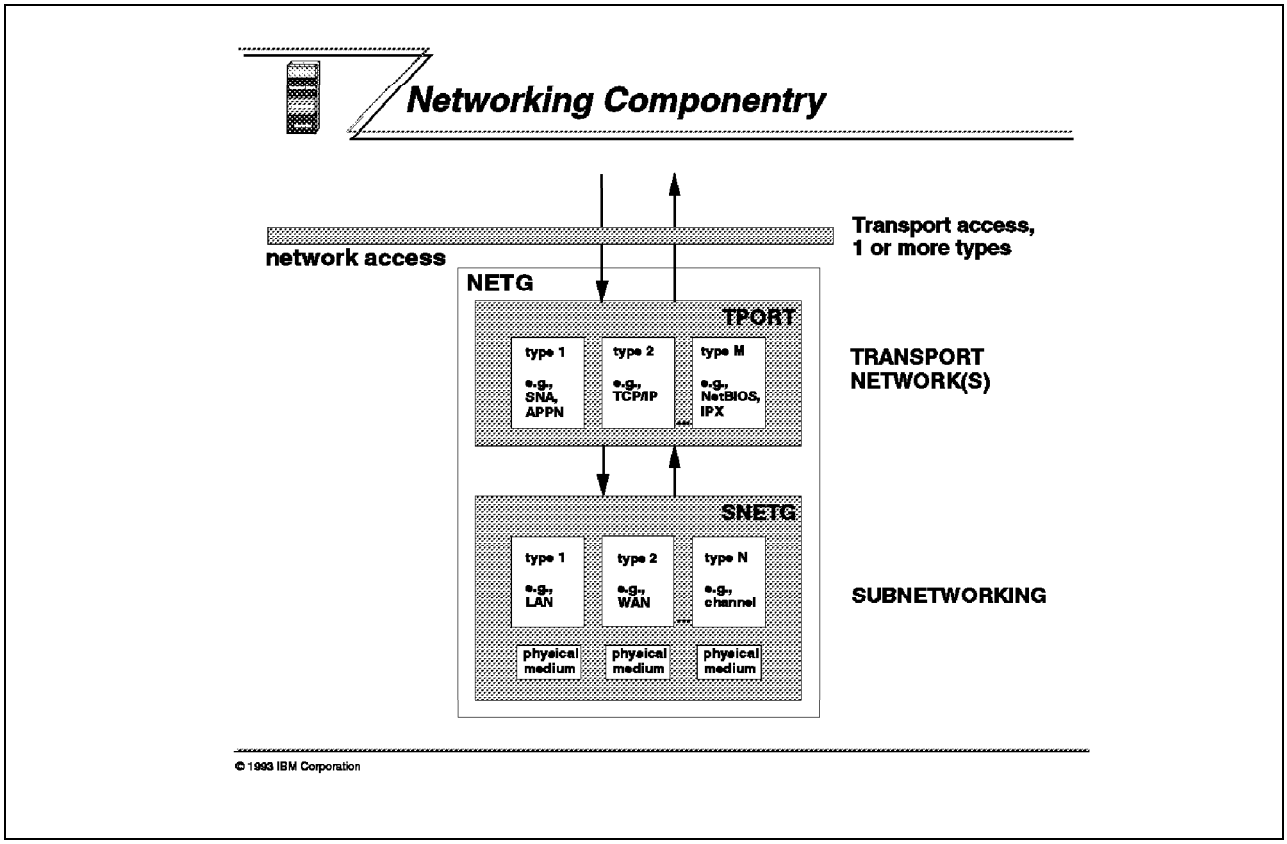



*Figure 19. Networking Componentry*

That is, NETG = TPORT + SNETG.

The TPORT box is composed of several selectable transport network types with the implication that only one is used at a time. Likewise, the SNETG box is composed of several selectable subnetworking types with the implication that only one is used at a time.

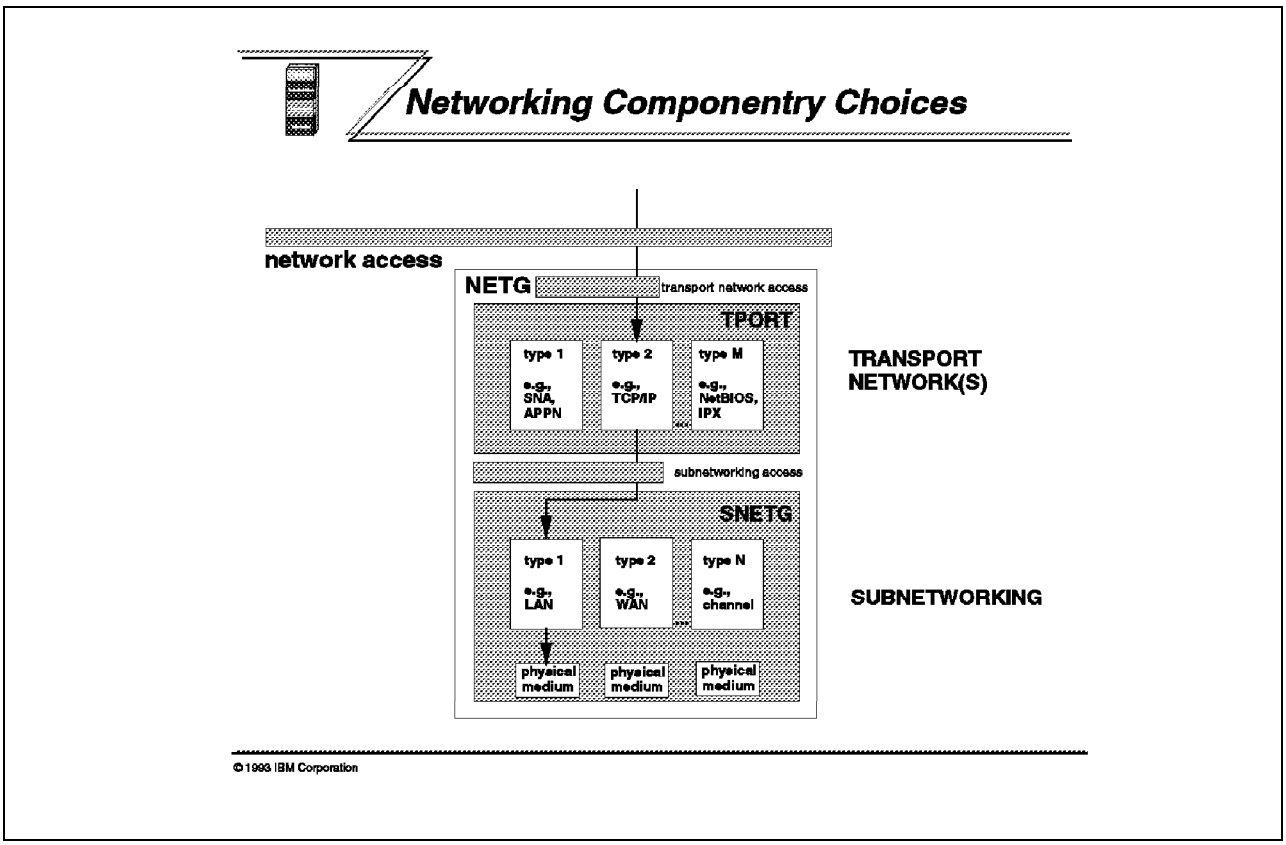



*Figure 20. Networking Componentry Choices*

In general (but based on product availabilities), a network-using (SUPP) program can “mix-and-match” types of transport networks and types of subnetworking, as shown in Figure 20. In this example, TCP/IP was chosen for the transport network and LAN was chosen for the subnetworking. It is as if a giant “rotary switch” exists between the SUPP and TPORT layers such that, by rotating the dial, the SUPP can choose exactly one of the TPORT choices. Similarly, the TPORT can choose exactly one of the SNETG choices. Of course, this rotary-switch choosing is based solely upon available (and announced) networking products.

The number of mix-and-match combinations between SUPP, TPORT, and SNETG available today is limited but this number is destined to grow with more choices becoming available.

***Transport Network Definition:*** A Transport Network is best described as "a collection of networking programs that exchange information between and among adjacent and non-adjacent computer systems using a variety of available communications media."

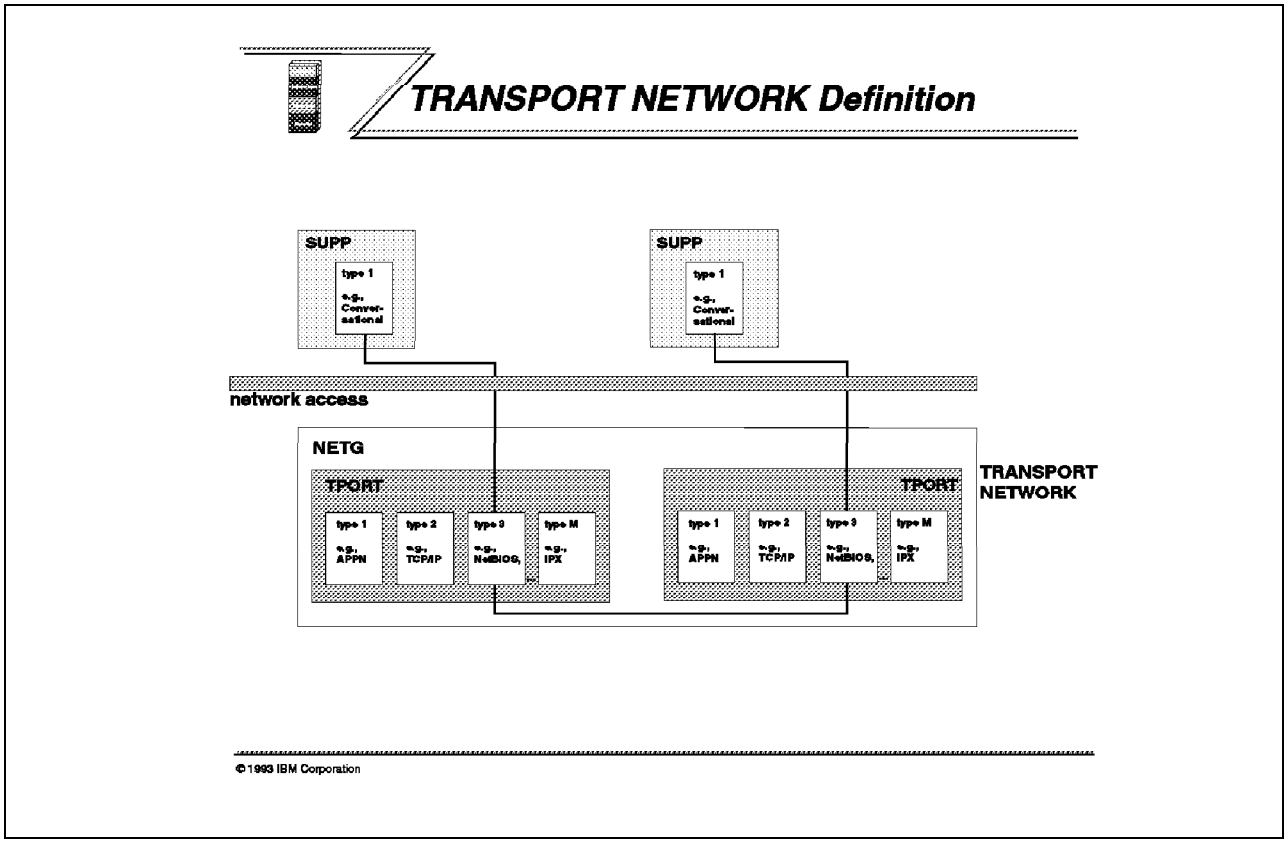



*Figure 21. Transport Network Definition*

The transport network (TPORT in Figure 21) is composed of selectable transport network types, including:

- SNA transport
- APPN transport
- TCP/IP transport
- NetBIOS transport
- IPX** transport
- others

In this example diagram, a NetBIOS transport network is being used to support conversational communication between applications. In most cases, both users of the transport network must select the same type of transport network. In the above diagram, both application support programs have selected a NetBIOS transport network.

For simplicity in the above diagram, subnetworking (SNETG) is purposely not shown. In a similar manner, many of today's networking diagrams purposely exclude subnetworking for simplicity.

***Subnetworking Definition:*** Subnetworking is best described as "a collection of networking programs that exchange information between immediately adjacent (or logically adjacent) physical communications/computing devices."

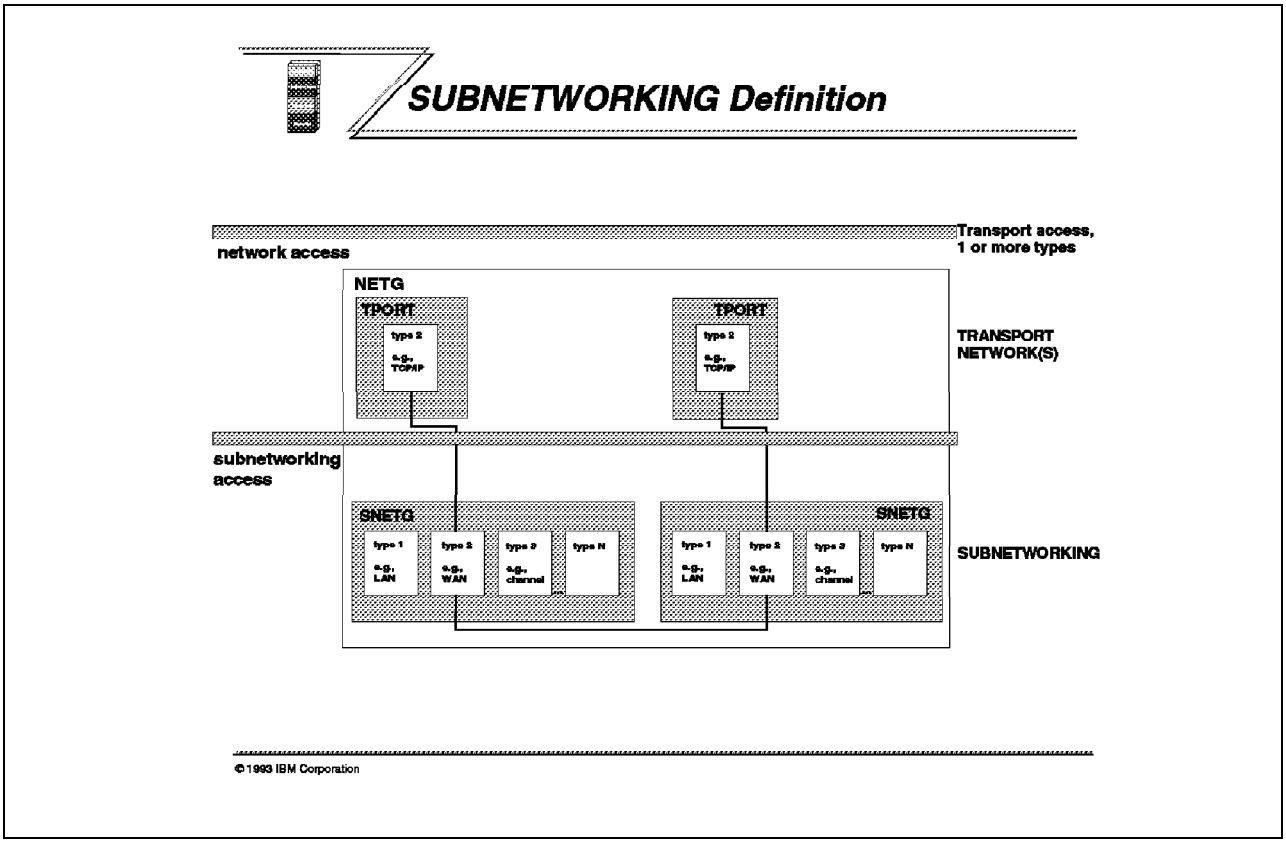



*Figure 22. Subnetworking Definition*

Subnetworking (SNETG in Figure 22) is composed of selectable subnetworking types, including:

- LAN
- WAN
- Channel
- others

In this example diagram, LAN (local area network) subnetworking is being used to support a TCP/IP transport network.

***Networking Cloud:*** A "network cloud" is often substituted for subnetworking across a particular communications medium, as shown in Figure 23 on page 32.

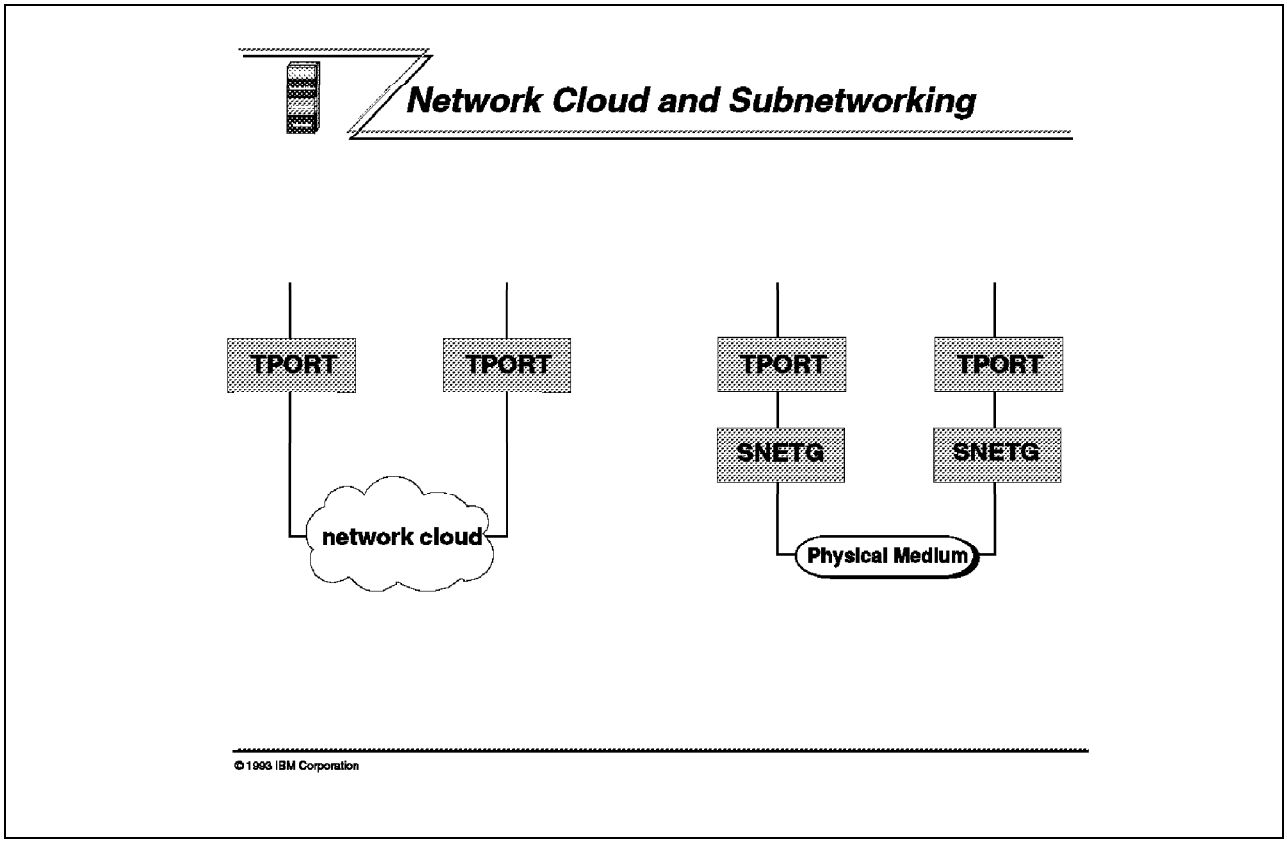



*Figure 23. Transport Network Cloud and Subnetworking Medium*

Thereby, subnetworking (SNETG) layers are interconnected with a particular physical medium and transport network layers are interconnected with a "network cloud," which is composed of subnetworking layers and a physical medium.

Network "clouds" are used in a quite universal manner in the computer industry to reduce the complexity of networking diagrams. Clouds can represent a single subnetwork, a series of subnetworks, or even (in many cases) one or more transport networks.

***Physical Media and "Short" Stacks:*** Adjacent subnetworking (SNETG) layers are interconnected across a physical medium of some sort (for example, a twisted pair of wires), as is shown in Figure 24 on page 33.

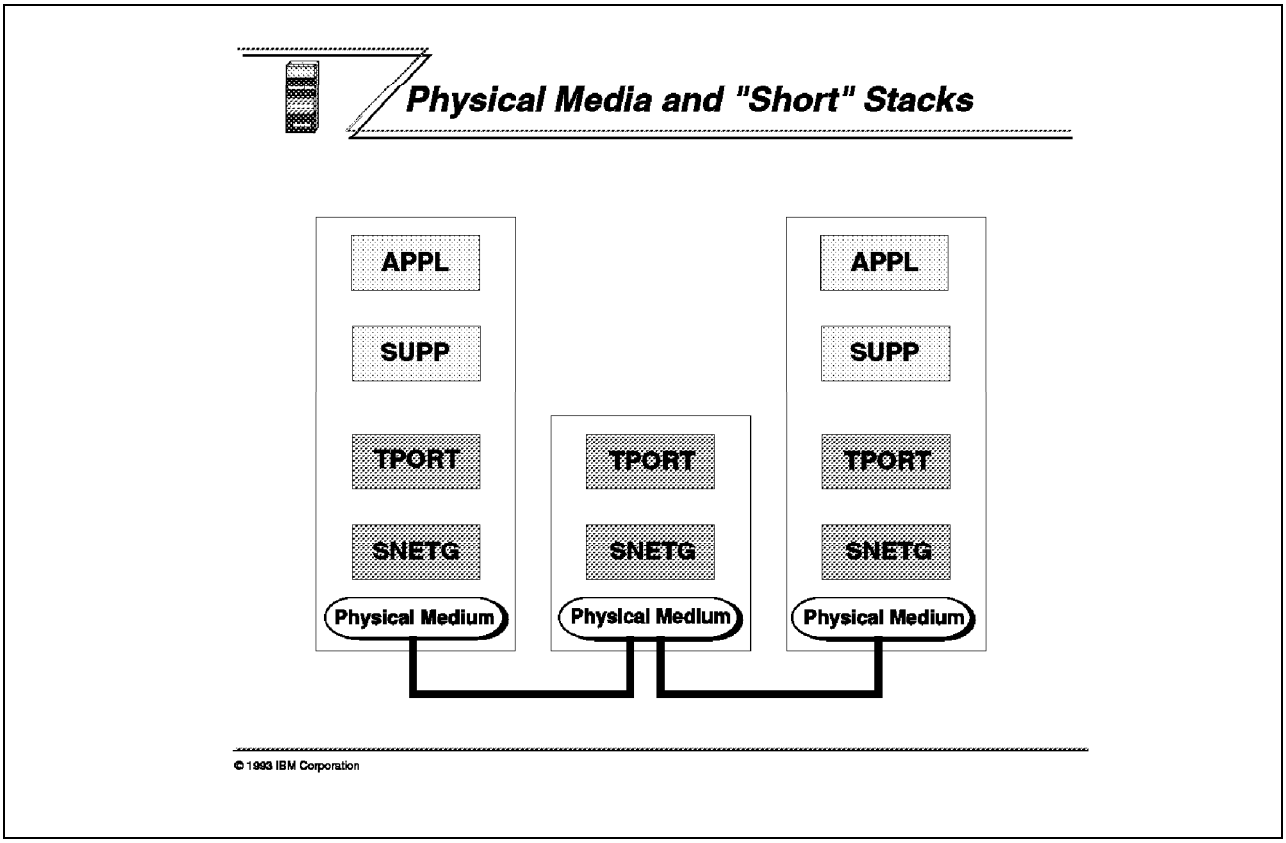



*Figure 24. Physical Media and "Short" Stacks*

In the middle of a communications network, the application (APPL) and application support (SUPP) layers are often absent, as is shown in the middle column of Figure 24, since there is usually no demand for application programming in the middle of a communications network. This creates a “short stack” of software, which is involved only with networking.

### 1.3.5.2 Repeaters and Multiplexors

Repeaters and multiplexors span subnetworking (SNETG) stacks “on the bottom side,” as shown in Figure 25 .

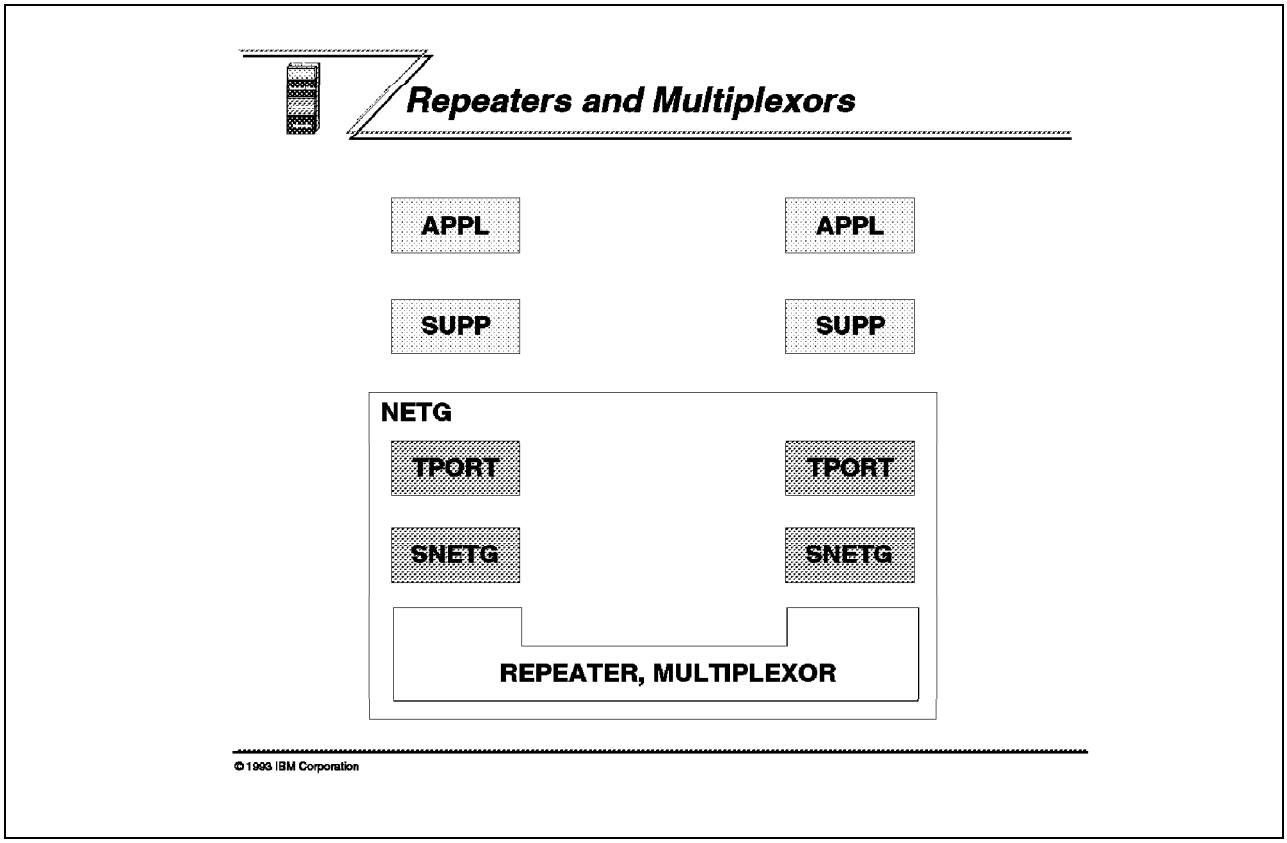



*Figure 25. Repeaters and Multiplexors*

In this way, subnetworking stacks are connected to one another through a physical medium, which is most often not discussed in subnetworking descriptions.

Also, for the sake of consistency throughout this book, we will use the following “formal” definitions. Please be aware that actual product implementations may not match these definitions. These terms are explained briefly below.

- **Repeaters**

  Repeaters operate at the physical layer (OSI layer one), simply extending the physical characteristics of the network by regenerating signals so that the optimum performance in terms of signal quality and distance can be achieved. Repeaters can sometimes also provide media conversion from one type to another, for example, from fiber optics to copper.

- **Multiplexors**

  Multiplexors operate at the physical layer (OSI layer one), taking data bits from several devices, and interleaving this data onto a single physical link. Such methods as Time Division Multiplexing (TDM) or Frequency Division Multiplexing (FDM) may be used to maximize the number of devices that can share a single link. Multiplexors “manage the bandwidth” available on the serial link, and, hence, are often called *bandwidth managers*.

### 1.3.5.3 Bridges, Routers, and Gateways

Aside from the reference model, some basic concepts and terminology must be defined so that particular technologies can be better understood. For instance, terms such as bridge, router, and gateway, keep reappearing in various types of literature relating to networking, but the definitions of these terms may not be consistent. Figure 26 illustrates the “formal” definition of these terms with regard to the OSI model.

In general, bridges span subnetworking (SNETG) layers, routers span transport network (TPORT) layers, and gateways span application support (SUPP) layers. However, these terms are not always used in this strict manner (for example, routers are described as existing at layer 3 and gateways are described as existing at layer 4 and above).

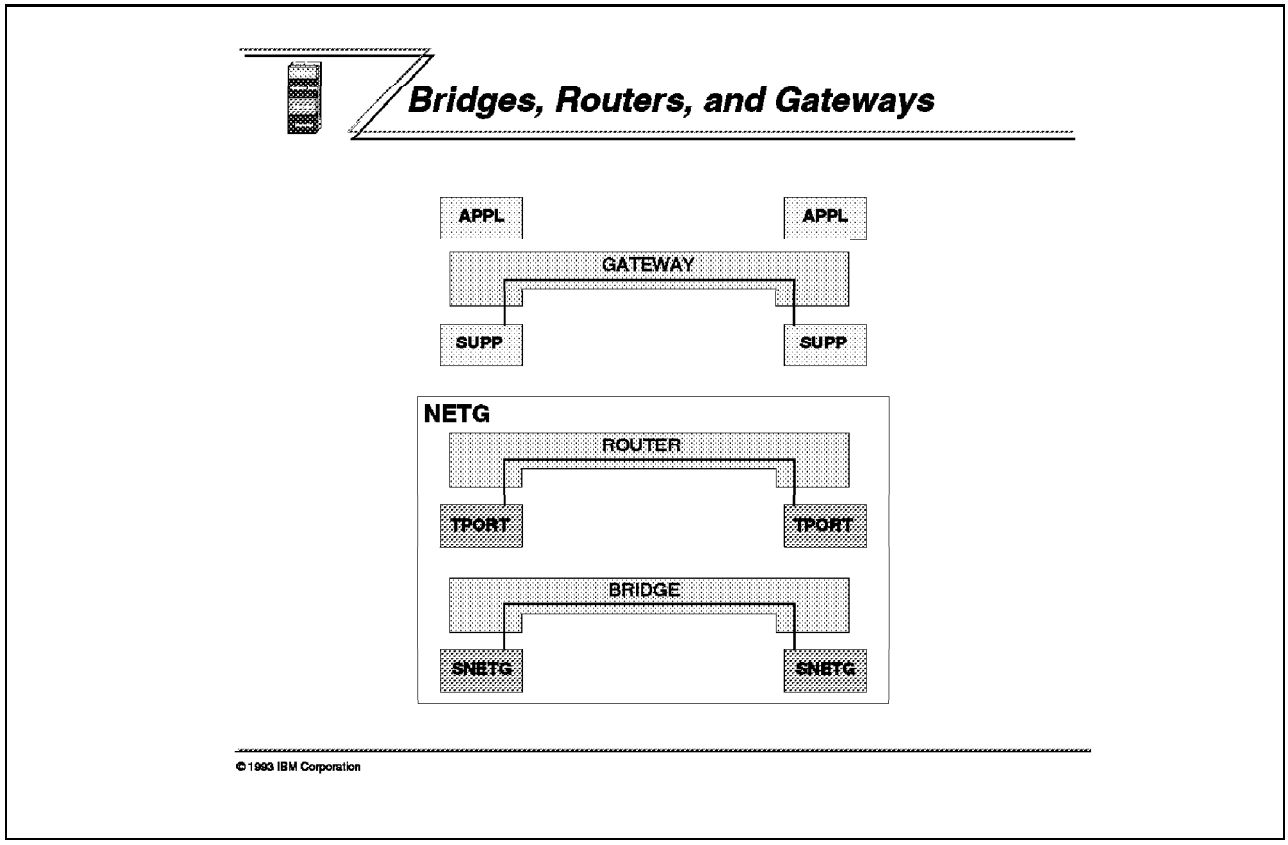



*Figure 26. Bridges, Routers, and Gateways*

Bridges, routers, and gateways all serve the same general purpose of interconnection of software.

***Bridges:*** Bridges effectively "melt" two LANs of like[12] types together.

Bridges operate at the Media Access Control (MAC) sublayer of the Data Link Control (DLC), OSI layer 2. They connect two LAN segments, and forward frames from one LAN segment to the other. Minimal processing is needed for this processing, and bridges can be quite efficient. Since all the processing takes place at OSI layer 2, and does not involve the higher layers, bridges are often referred to as being "protocol-independent." A bridge does not care if it transports TCP/IP, DECnet**, or OSI protocols, which all operate at OSI layers 3 and above.

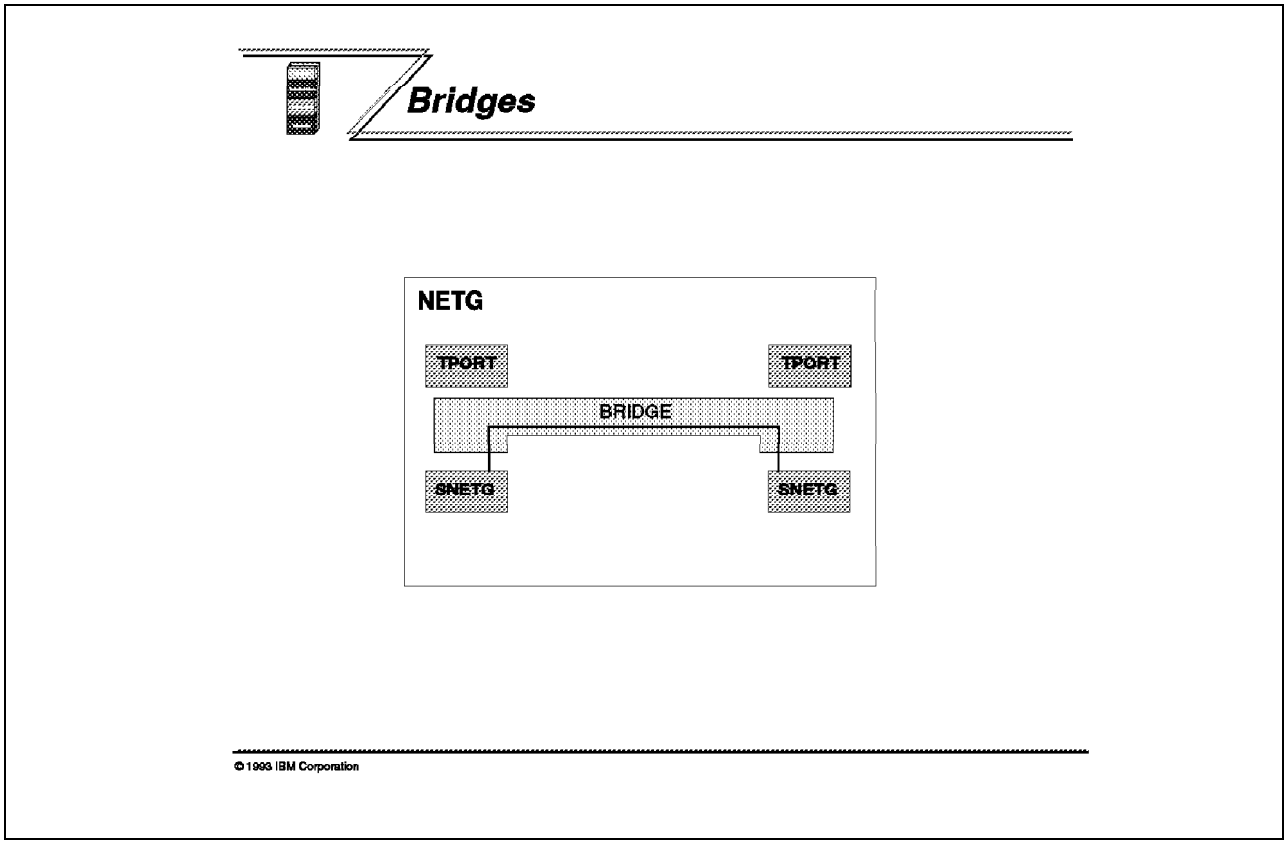



*Figure 27. Bridges*

*Local and Remote Bridges:* Bridges can exist between adjacent LANs, or they can exist between non-adjacent LANS that are separated by a WAN (or other non-LAN subnetworking technology). In the first case, the single bridge is called a local bridge. In the second case, one bridge is called a local bridge and the other a remote bridge. The two bridges (one local and one remote) are sometimes collectively called a split bridge (one part in one LAN and another in different, non-adjacent LAN).

*Protocol Converters:* A "special-case bridge" can handle not only layer 2 protocols like-to-like, but also layer 2 protocols like-to-unlike (involving protocol conversion). Some level of lower-layer protocol conversion is often a feature of a bridge, router, or gateway without the product being so advertised. Between

[12] In general, bridges operate with lower-layer (for example, token-ring or Ethernet) protocols like-to-like, not like-to-unlike.

token-ring and Ethernet LANs there exist many special types of bridges that involve hidden (or at least proprietary) protocol conversion.

***Routers:*** Routers interconnect different networks, and determine the optimal path for a packet of data to traverse, through an interconnected series of networks, to reach its destination. Routers operate through OSI layer 3 (the network layer), and are *transport network protocol-specific*. At the network layer, the network address varies according to the choice of protocol. Thus, the addressing scheme for SNA will be different from DECnet, and a router must be purchased for each of the desired protocols (except for the case where a multi-protocol router — to be discussed in Chapter 2, “Positioning and Usage of Technologies” on page 47 — is available and can suffice). Routers enable the connection of different subnetwork types to perform this routing function, for instance, connecting a LAN to a Frame Relay wide area network for a specific OSI layer 3 protocol.

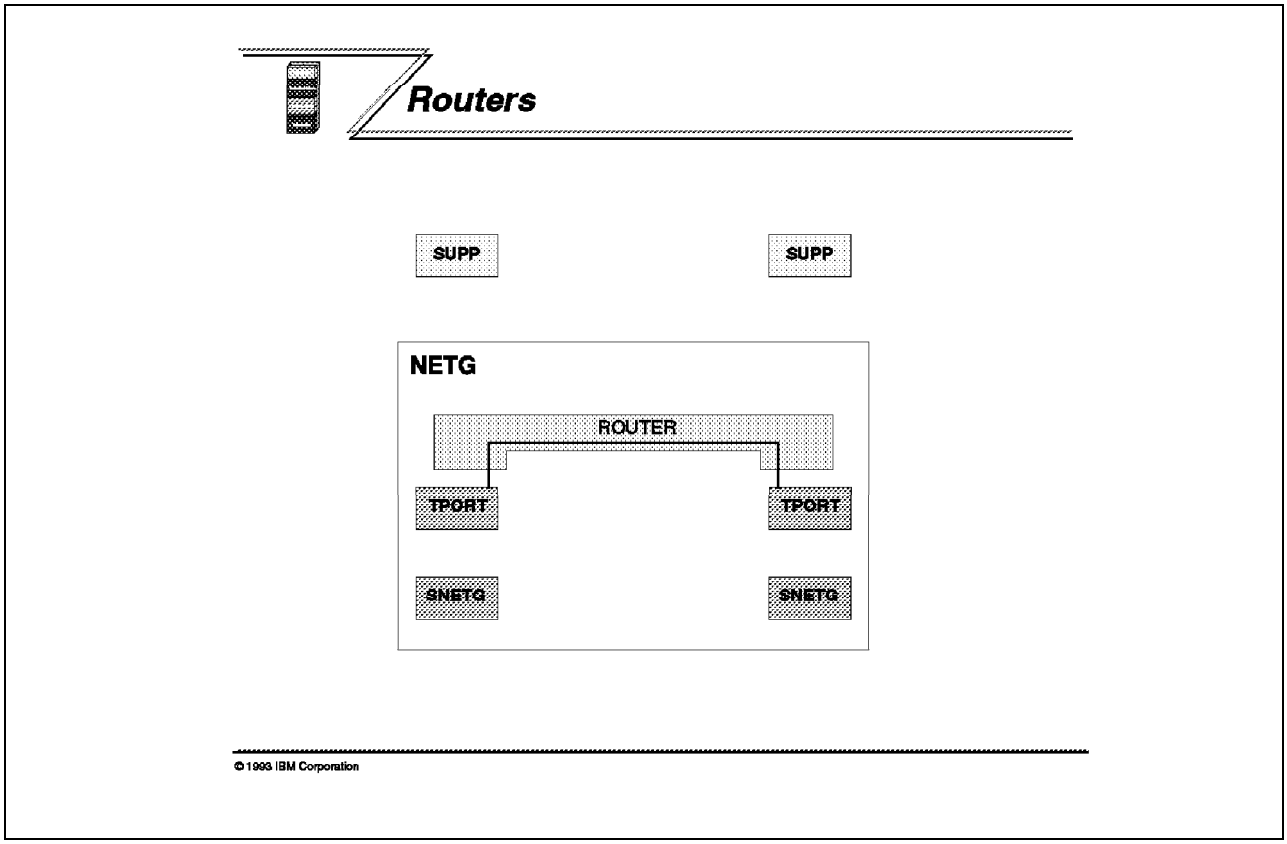



*Figure 28. Routers*

A Brouter is a combination of a remote bridge and a router, performing the functions of both. Often it will route the protocols that it knows how to route, and bridge all other protocols.

***Gateways:*** The most basic (and earliest) concept of a gateway is that entity (software and hardware, combined) which "funnels" traffic from one place to another. Gateways exist in many forms: between LANS, between other types of networks, and between applications. Gateways exist at that point where LANs, networks, or applications *touch* one another. Gateways exist within networks and between networks. In the case of network gateways, the networks are usually of different types; for example, a collection of devices (that is, a device network) attached to a communications network. In the middle 1970s (and even today), devices in a subscriber network could be hidden (address-wise) behind the gateway where they were collectively attached to a service network.

Gateways usually operate above the network layer, layer three, and may involve all layers. The key to a gateway is *conversion*. A gateway will convert all layers up through the layer at which it operates. A common type of gateway is the *application gateway*, which is very application-specific, and which converts all seven layers as needed. An electronic mail gateway, which takes office mail from one type of electronic mail system (such as, IBM PROFS) and converts it to another mail format (such as, DEC All-In-One**), is an example of this type of gateway.

There are gateways which provide protocol conversion from one session and datastream type to another, such as VT100 to 3270. These gateways usually operate at the higher OSI layers.

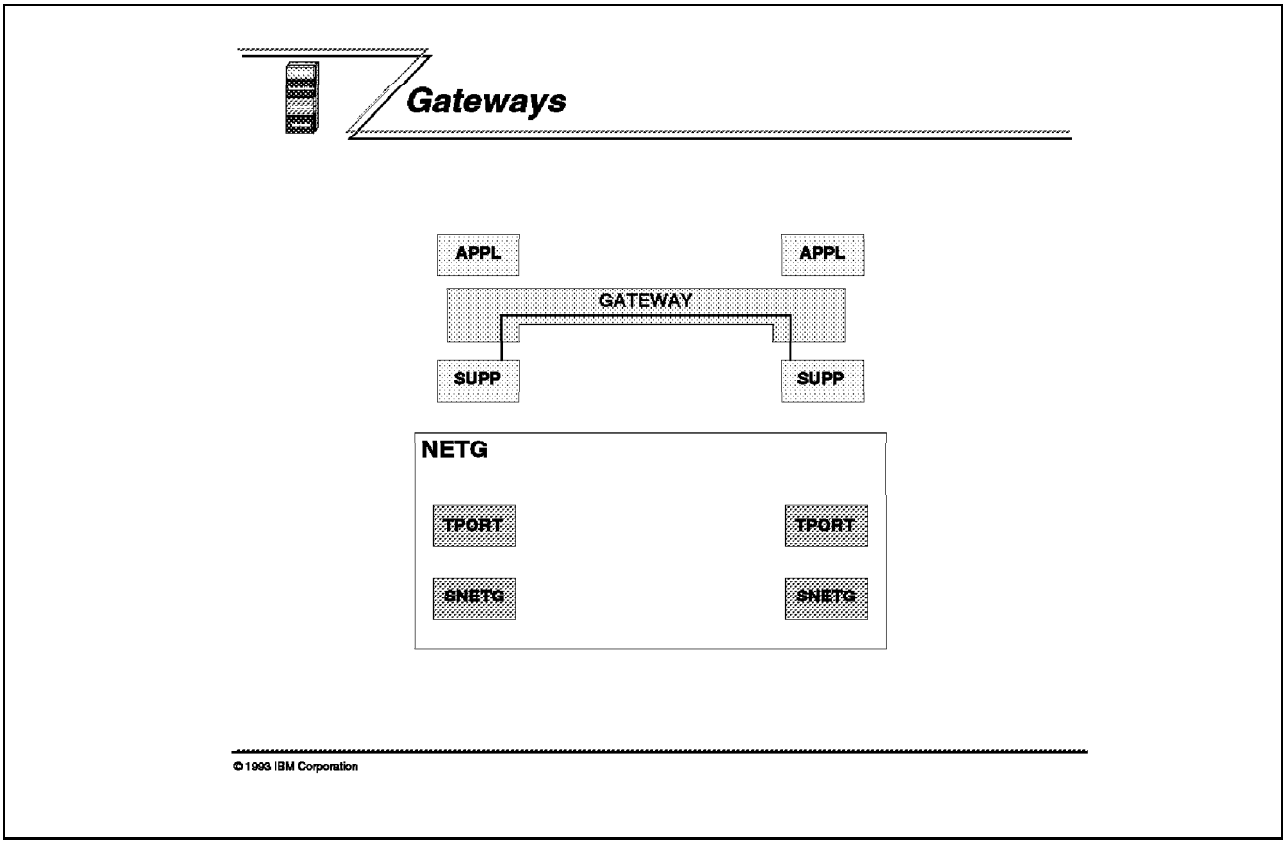



*Figure 29. Gateways*

Although not shown in the above diagram, another special type of a gateway is a *transport gateway* spanning transport network (TPORT) layers, where different transport facilities are matched to each other. Examples of transport gateways include a NetBIOS application transported over a TCP/IP network, or a TCP/IP

application run over an SNA network. These gateways perform a *relay function*, where the original application information is relayed over one or more transport protocols to its final destination. There is usually extra code provided at the transport level to *compensate* for the change in transport protocol, and to permit the relaying of the application information through the network, without impacting the application program itself.

## 1.4 Importance of Technologies

As shown in Figure 30, technologies can be grouped by layers, from the lowest to the highest:

- Subnetworking
- Transport network
- Application support

The remaining sections of this chapter discuss technologies, by layers. In many cases, more than one set of these technologies can be useful, and even vital to implementing good computing solutions.

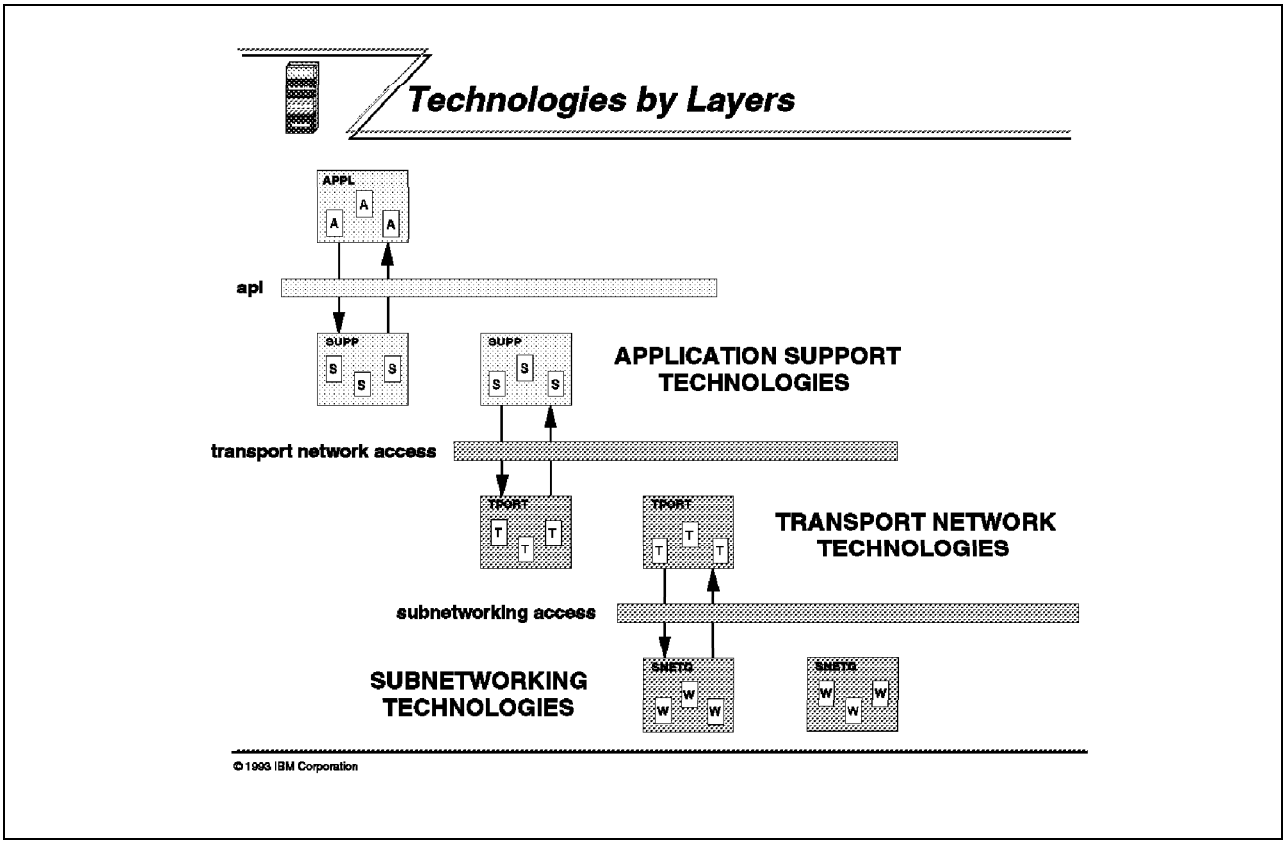



*Figure 30. Technologies by Layers (SNETG, TPORT, SUPP)*

In the above figure, programs within the application program group (layer) are labeled *program 'A'*, indicating an application program. Similarly, programs within the application program group are labeled *program 'S'*, indicating a support program. In the transport network program group, programs are labeled *program 'T'*, indicating a transport network program. In the subnetworking program group, programs are labeled *program 'W'*, indicating the “wire” (or equivalent physical communication medium).

### 1.4.1 Direct and Indirect Networking

Ideally, only application support programs, and not ordinary application programs, are involved in networking, as shown in Figure 18 on page 27. There are cases, as shown in Figure 31, where ordinary application programs deal directly with networking instead of indirectly with networking through application support. These cases are often to effect extremely high efficiency and performance at the expense of application program simplicity, portability, and maintainability.

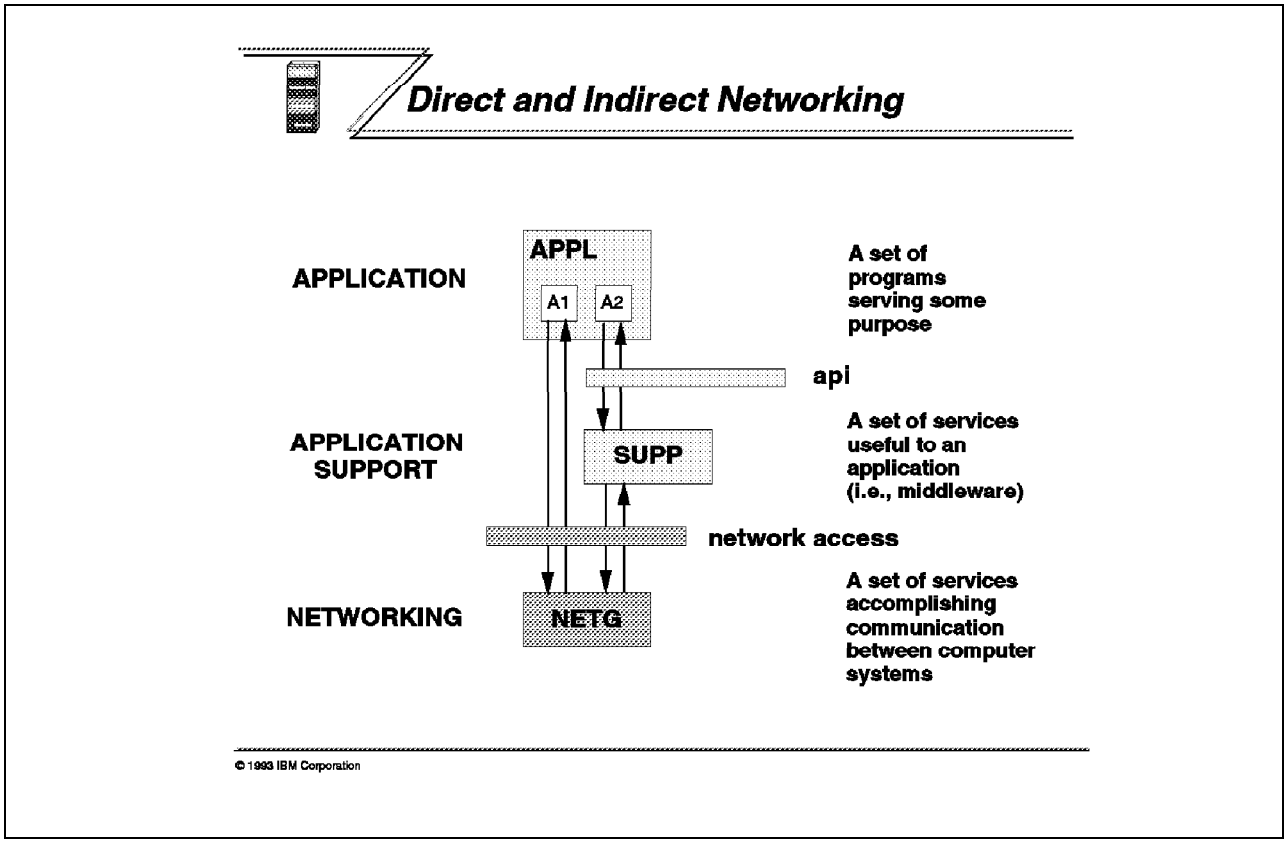



*Figure 31. Direct and Indirect Networking*

Indirect networking saves enormous amounts of application development time and complexity. It shields the developer from some of the many complexities of networking.

## 1.4.2 Simple Networking

Simple networking involves just a single network interconnecting two systems, within each of which are application programs.

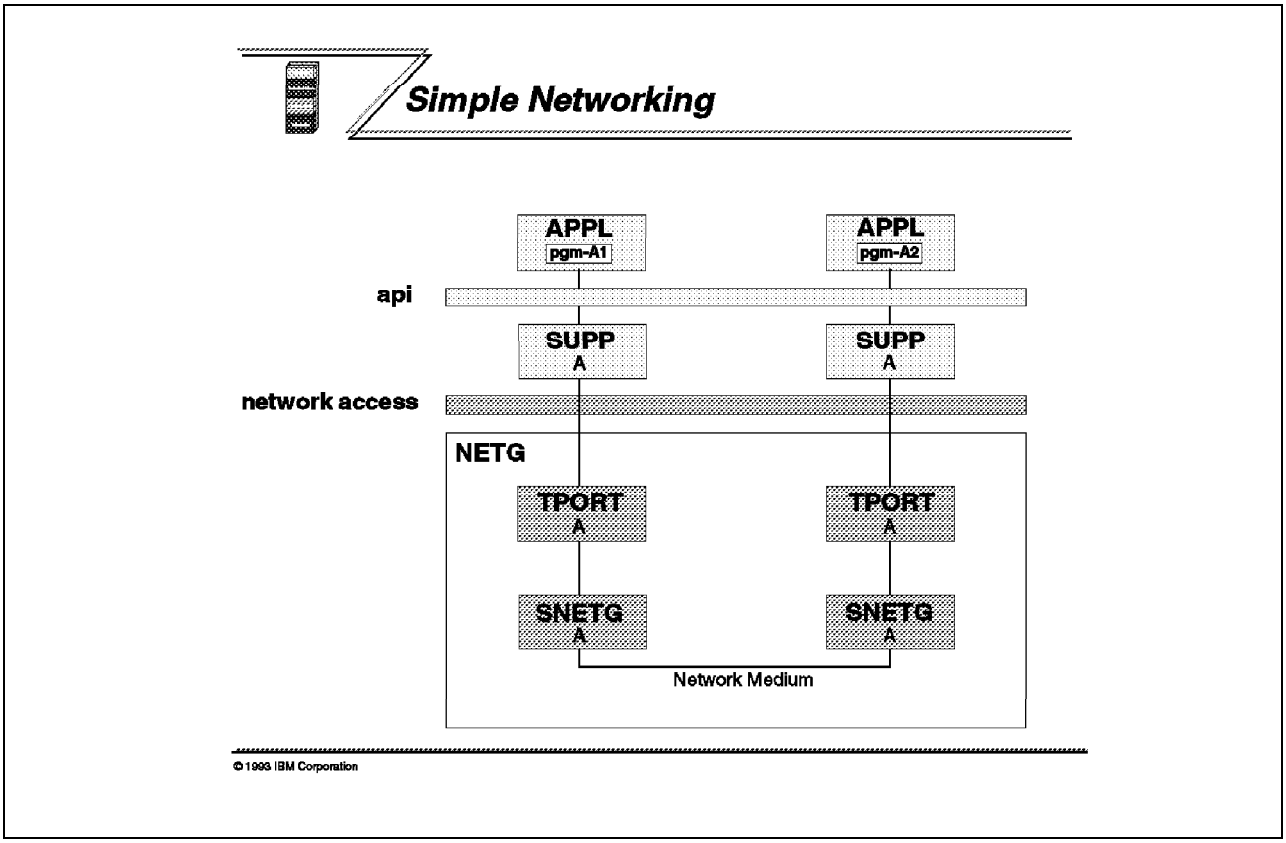



*Figure 32. Simple Networking*

With simple networking, program 'A1' and program 'A2' within the same application intercommunicate (indirectly through the SUPP layer) with great simplicity and efficiency across a single network of some particular type. The networking is used by the SUPP software and provided by the TPORT software. The application support is used by the APPL software and provided by the SUPP software.

### 1.4.3 Separated (Private) Networking

Separated networking involves two or more disjoint networks each interconnecting two (or more) systems. Each separate network has its own “private” or separate connections, even if several of the networks are of the same type (for example, two separate NetBIOS networks). Nothing is shared, logically or physically.

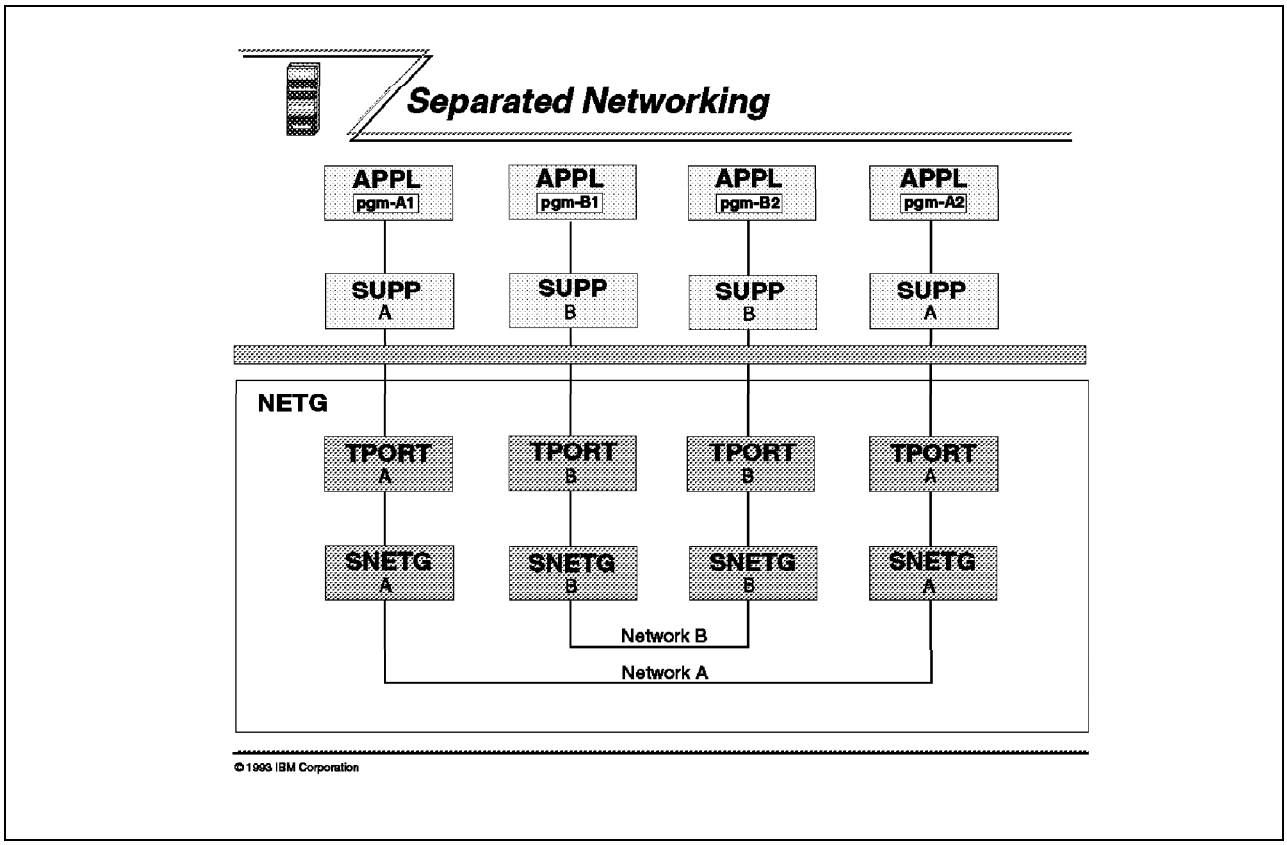



*Figure 33. Separated Networking. Disjoint (or Private) Networks*

In the above diagram, program-A1 and program-A2 are intercommunicating across a network-A while program-B1 and program-B2 are intercommunicating across a *different* network-B.

## 1.4.4 Combined (Shared) Networking

Combined networking involves two (or more) separate networks operating as one combined network.

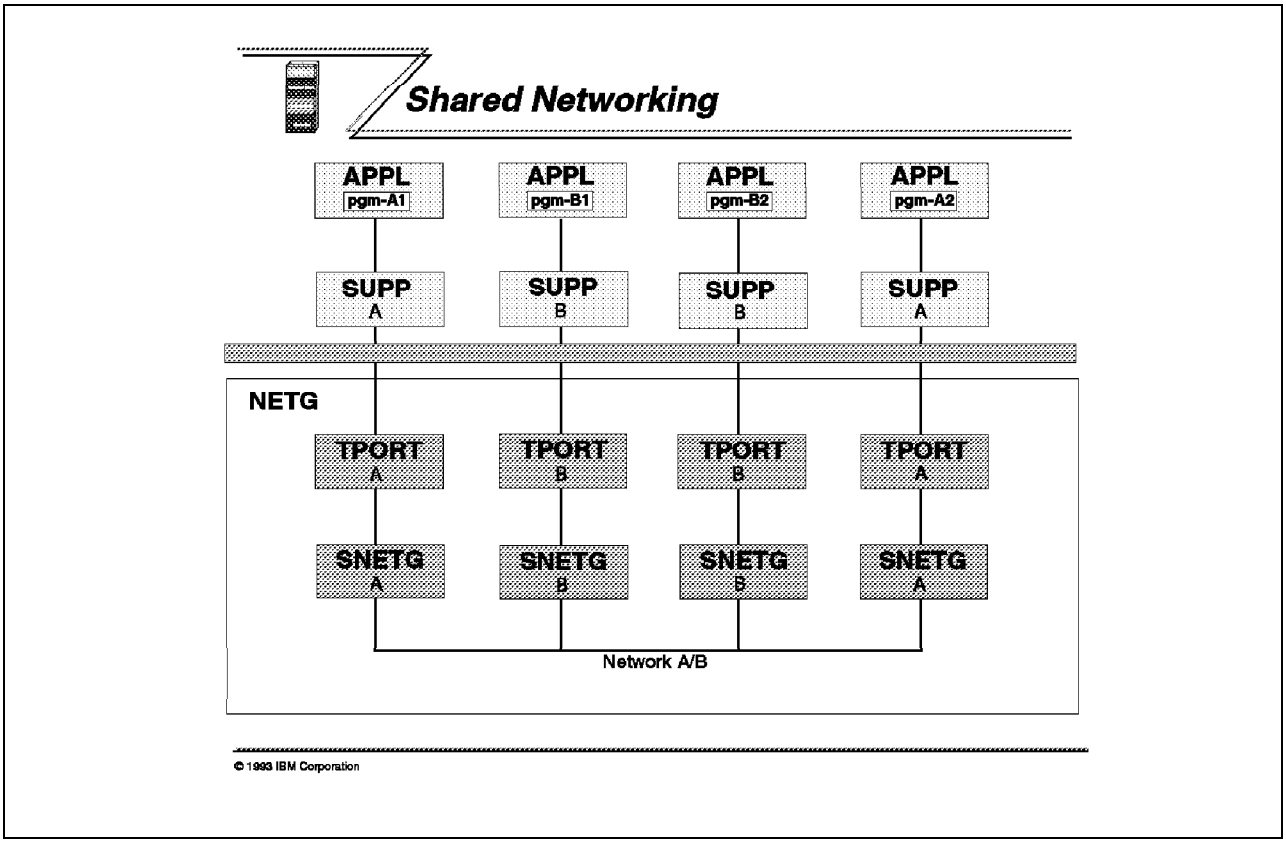



*Figure 34. Combined Networking. Common (or Shared) Network*

In the above diagram, program-A1 and program-A2 are intercommunicating across a network-A/B while program-B1 and program-B2 are intercommunicating across the same *shared* network-A/B.

# Chapter 2. Positioning and Usage of Technologies

The objective of this chapter is to describe today's technologies and point out some of their inherent advantages and disadvantages[13] .

## 2.1 Introduction

There are a variety of considerations that must be kept in mind as one evaluates alternative solutions to complex connectivity and interoperability problems. This chapter discusses the currently available solutions that exist to solve a wide variety of these problems. Each solution is described, and its inherent advantages and disadvantages are listed. However, the selection of any particular technology must be done with regard to a specific situation. In Chapter 3, “Typical Customer Situations” on page 95, the applicability of these technologies within a particular situation is illustrated using a set of typical customer situations.

Although the terms bridge, router, and gateway are widely used throughout the industry, it is often difficult to classify a particular technology with this hierarchy, especially as the technology evolves. For instance, consider bridges that originally connected two adjacent local LAN segments. Now bridges can connect two LAN segments over a wide area network, and some higher-layer functionality is involved. This “remote bridge” is no longer a bridge in the formal definition sense of operating strictly at OSI layer 2, yet lacking a better set of definitions, we still call it a “bridge.”

To avoid this definitional dilemma, we will examine these technologies in terms of the general levels of functionality at which they primarily operate: subnetworking, transport network, or application support. We will examine these technologies in a sequential fashion, starting with techniques that are concerned with the lowest levels of connectivity at the subnetworking OSI layers 1 and 2 (physical and data link control layers), and then evaluate techniques which gradually increase the scope of connectivity and interoperability. The techniques that are examined include the following:

1. Techniques that operate primarily at the subnetworking (SNETG) level:
   - Separate Wide Area Networks
   - Bandwidth Management
   - Shared Subnetworks - LANs
   - Local Bridges
   - Encapsulation
   - Extended Subnetworking Approaches
     - Remote/Split Bridges
     - Data Link Switching (DLSW)
     - Packet Interfaces - Frame Relay, X.25, SMDS, ATM
2. Techniques that operate primarily at the transport network (TPORT) level
   - Native Multi-Protocol Routers

[13] Because circumstances where these technologies apply will differ and because more than one technology may be applicable, and because implementation and packaging of these technologies varies widely, there can be no simple and definitive choices.

© Copyright IBM Corp. 1994

- Mixed-Protocol Standards (RFCs, etc).
- Multi-Protocol Transport Networking (MPTN)

3. Techniques that operate primarily at the application support (SUPP) level, including applications services and application programming interface
   - XTI
   - Middleware
   - Remote APIs
   - Application Gateways
   - Multi-Protocol Servers

For each of these possible connectivity and interoperability solutions, the technology is described in general terms. Inherent technical and business advantages and disadvantages of this solution are then described. Assume that each node[14] represented in the diagrams is separate and distinct from other nodes indicated, and that these nodes are not currently physically connected or in any way currently communicating. The nodes where the applications exist will be referred to as "end nodes," and any devices in between performing key functions are referred to as "intermediate nodes." Sometimes the solutions will impact the end nodes in terms of changing the application or adding additional equipment (such as adapter cards) to the end node; other times the end node will be untouched, but additional hardware and software must be procured for one or more intermediate nodes. These modifications will be noted for each solution. The network "clouds" shown in these diagrams represent *transport networks* and *not* physical networks (that is, they are running a single transport protocol). In fact, several transport networks may run on top of a single physical network. Nor does the "cloud" indicate anything about the configuration of the physical network itself, which may have many intermediate nodes.

Again, please note that these are *technology concepts*, and real products will probably implement several of these technologies. For instance, the IBM 6611 routers implement native multi-protocol routing, local bridging, remote bridging, and data link switching. Other router vendors also include much more than just native multi-protocol routing. However, once these technologies are understood, it is much easier to evaluate particular product implementations.

[14] The terms system and node are often used interchangeably.

## 2.2 Subnetworking Technologies

Subnetworking technologies are implemented at the lowest layers of a network.

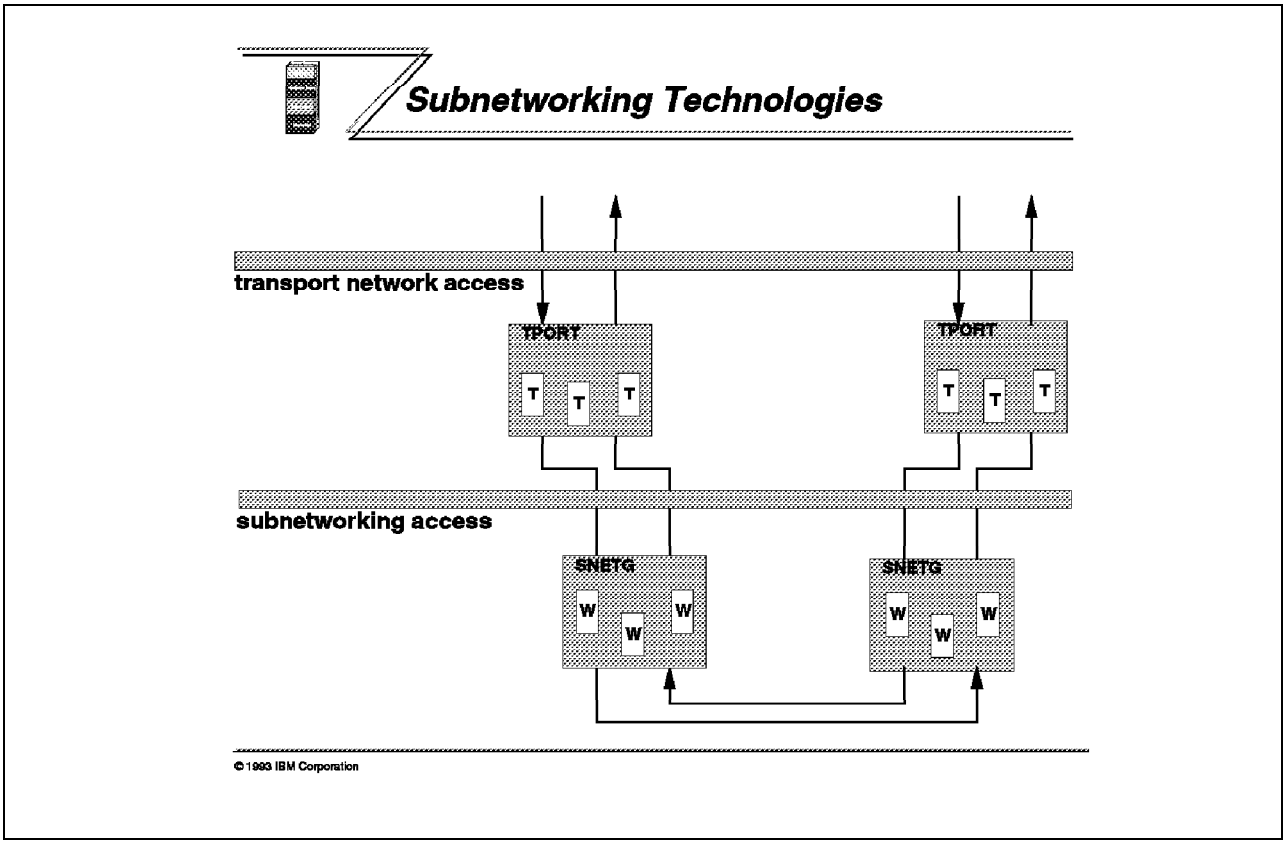



*Figure 35. Subnetworking (SNETG) Technologies*

Some of the technologies implemented within the subnetworking layer are:

- Separate Wide Area Networks
- Shared Subnetworking
  - Shared WAN (Multiplexor, Bandwidth Manager)
  - Shared LAN (Multi-Protocols)
  - Shared LAN (Local Bridge)
- Encapsulation
- Extended Subnetworking
  - Remote/Split Bridge
  - Data Link Switching (DLSW)
  - Packet Interfaces (for example, Frame Relay)

There are a variety of techniques available that concentrate on providing connectivity solutions at the lowest OSI layers for wide area networking. These technologies provide a shared subnetwork environment for a variety of transport protocols. One might refer to these technologies as examples of *extended subnetworking* because they attempt to make the wide area network look like a single large subnetwork which is transport protocol-independent. In fact, with these extended subnetworking approaches, the end nodes believe that they are

in direct connection with each other and are unaware of the intermediate nodes which transmit the traffic over the wide area network[15] .

"Extended subnetworking" technologies include:

- The *Remote (Split) Bridge* technique

  See 2.2.5, "Shared LAN (Remote/Split Bridge) Technique" on page 59

- The *Data Link Switching (DLSW)* technique

  See 2.2.7, "Data Link Switching (DLSW) Technique" on page 67

- The *Packet Interface* technique

  See 2.2.8, "Packet Interface (Using Frame Relay) Technique" on page 69.

[15] This is in contrast to the situation where there is explicit (layer 3) routing used to traverse multiple types of links/subnetworks (for example, LAN-WAN-LAN); in that case, the end node/system is aware of the (adjacent/first) intermediate node/system.

## 2.2.1 Separate WANs Technique

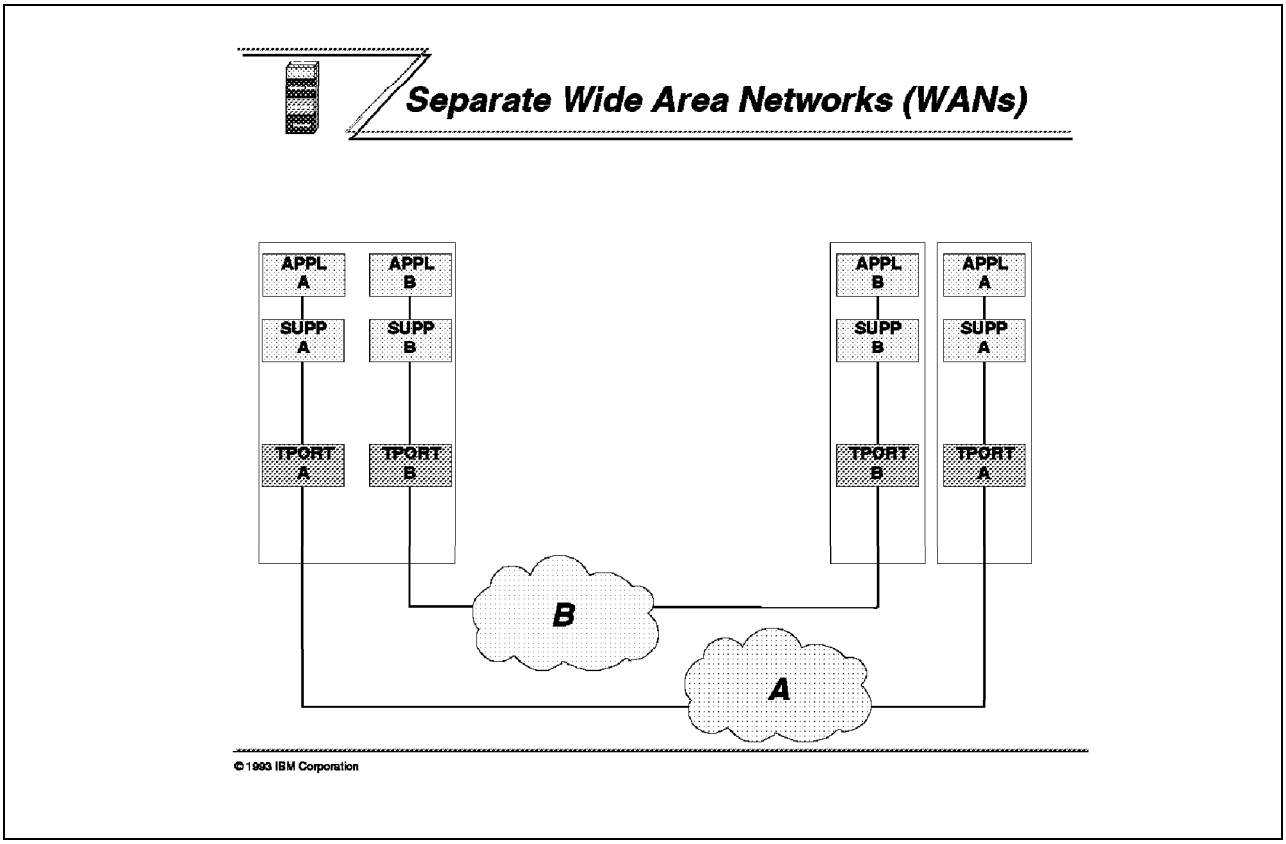



*Figure 36. Separate Wide Area Networks (WANs) Technique*

As shown in Figure 36, the applications in the node on the left side of the diagram need to communicate with identical applications in specific nodes on the right side of the diagram. Assume that the two nodes on the right side of the diagram are in the same physical location, which is remote from the node on the left. Appl-A uses transport protocol A, which is different than Appl-B's transport protocol. A common approach is to use separate serial links to connect the three nodes (one on the left and two on the right of the diagram).

By using separate facilities for each application, separate networks have been created. Thus, Appl-A would use only the network that utilized transport protocol A, and Appl-B could communicate only over the second network. Appl-A cannot communicate with Appl-B.

Separate networks have no impact on the applications, but require that multiple copies of all hardware, software, and facilities necessary for communications be purchased.

**Advantages of Separate Wide Area Networks**

- From a conceptual viewpoint, it is simple to visualize. Each transport protocol requires its own network.
- There is no multiplexing required, and each network can run its own Data Link Control protocol as well as a different transport protocol. For example, Network A might be using HDLC, while Network B might be using X.25.

- With no multiplexing of the different protocols required, potentially better throughput for each protocol might be realized. Depending upon throughput requirements and traffic volume, different line speeds might be chosen.
- Management might be easier, since each network has its own transport protocol and thus only one type of network manager. People can specialize in the administration, configuration, and problem resolution for this single protocol.
- Organizational control might be easier as each separate group has its own network. Agreements must be reached as how to handle shared servers, such as in the left-hand node.
- Security might be easier to administer over separate networks, particularly if one of the applications requires a higher degree of security, such as encryption.

**Disadvantages of Separate Wide Area Networks**

- Physical network costs might be prohibitive, such as separate lines, wiring, network interface cards, modems, etc., especially if there are many transport protocols, locations, or nodes involved.
- Personnel costs can get very high, if each network has its own staff to support administration, configuration, and problem resolution.

## 2.2.2 Shared WAN (Multiplexor) Technique

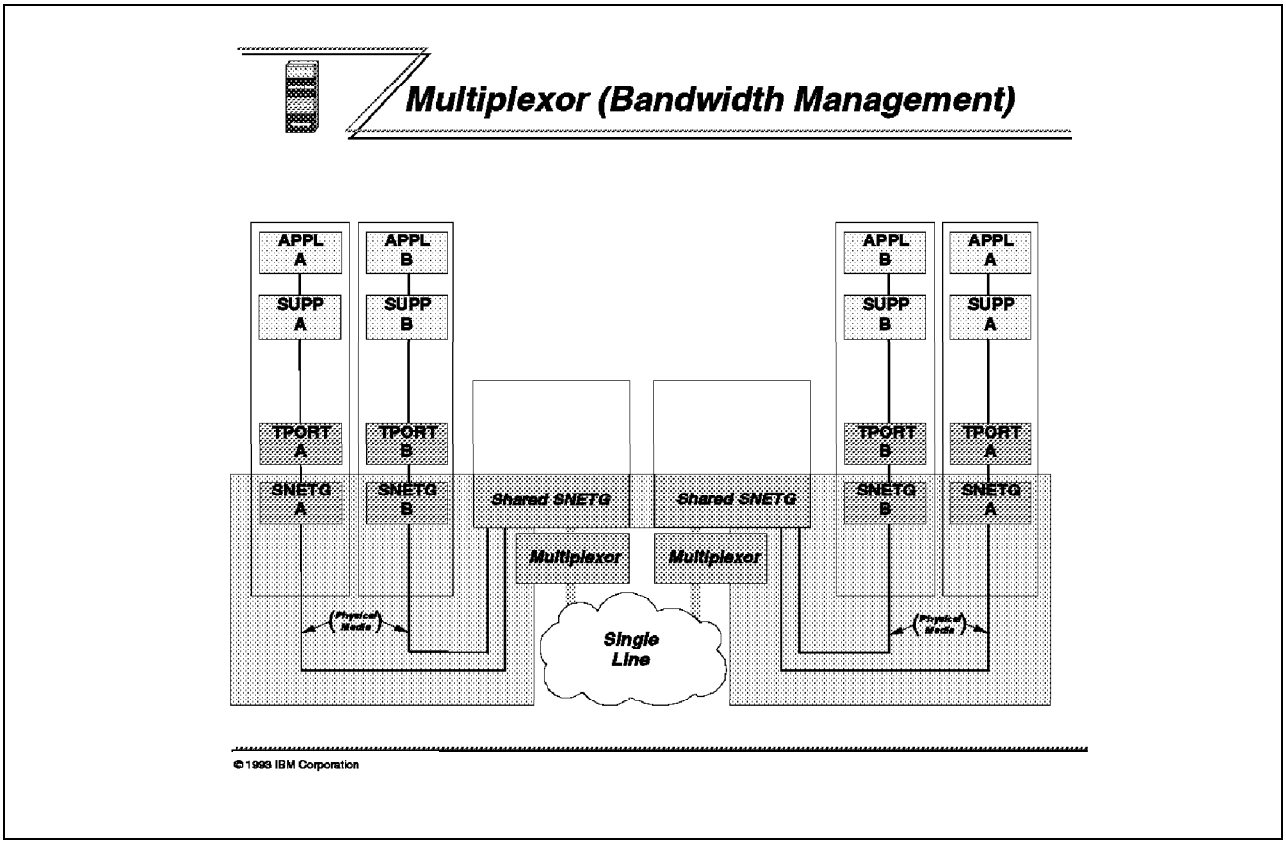



*Figure 37. Shared WAN (Multiplexor, Bandwidth Management) Technique*

If leased circuit costs are prohibitive for separate wide area networks, then a possible solution is to use a **multiplexor** or **bandwidth manager** which allows the sharing of leased facilities. These devices basically interleave bits of data over shared serial lines, using such techniques as frequency division multiplexing (FDM) or time division multiplexing (TDM). As shown in Figure 37, the end nodes in each location will interface to their individual bandwidth manager through a subnetwork interface. This subnetwork between the end nodes and the bandwidth manager is often a serial interface, although LAN interfaces are also possible. The bandwidth manager will take data from multiple local links/subnetworks and multiplex this traffic over a single line to the recipient bandwidth manager. The type of multiplexing and the exact algorithms used between the bandwidth managers vary by vendor, and thus are often proprietary. Based on these algorithms, the recipient bandwidth manager will know which local subnetwork should receive the data. Definitions for bandwidth managers are based on a port-to-port basis; for instance, data from port A in Location #1 (a serial link to End Node A) should always go to port B in Location #2 (a serial link to End Node B).

Since the multiplexing function is taking place at the lowest OSI layer, bandwidth managers are insensitive to the transport network, and are thus *protocol-independent*. However, problems might arise if certain timing-sensitive Data Link Control protocols are not properly handled (such as SDLC), so actual products may provide termination of the DLC at the local segment before the data is multiplexed over the serial link. To “terminate” the DLC means to handle these timed messages by emulating the responses at the originating

location, and not to depend upon the messages being returned across the slower-speed wide area network. Actual products may also include remote bridges and routers to be more efficient and provide some transport layer sensitivity. For instance, NET IDNX** and Newbridge bandwidth managers include these capabilities in their products.

Each geographical location that wishes to participate in using this shared bandwidth must have a bandwidth manager. In order to utilize the bandwidth manager, the end nodes must have the appropriate physical attachments, such as serial ports and modems for a serial link connection, or a LAN adapter if a LAN connection is possible. There will be no impact to the applications.

**Advantages of Multiplexors (Bandwidth Managers)**

- Multiplexing permits the sharing of higher-speed leased circuits.
- Statistical multiplexing techniques allow for some “over-committing” of WAN bandwidth, yielding extra savings for bandwidth managers.
- The multiplexor or bandwidth manager can accomodate data, voice, and video traffic.
- Depending upon the number of leased circuits between sites, the topology of the bandwidth manager network, and the capabilities of the bandwidth manager product, redundancy and automatic re-routing of traffic may be possible.

**Disadvantages of Multiplexors (Bandwidth Managers)**

- The static nature of traffic definitions inhibits expandability. For instance, if End Node A in Location #1 wishes to communicate with End Node B in Location #2 and End Node C in Location #3, then End Node A must have two serial interfaces into two separate ports for the bandwidth manager in Location #1, one for each of the other locations. End Node A must be aware of which serial interface to direct the traffic to depending upon the traffic need to go to End Node B or End Node C. Adding more End Nodes or locations can be very difficult, and obtaining a “mesh” topology where any node can communicate with any other node might be prohibitively expensive.
- Timing-sensitive data link control protocols, such as SDLC, may not be handled properly causing sessions to be dropped.

## 2.2.3 Shared LAN (Multi-Protocols) Technique

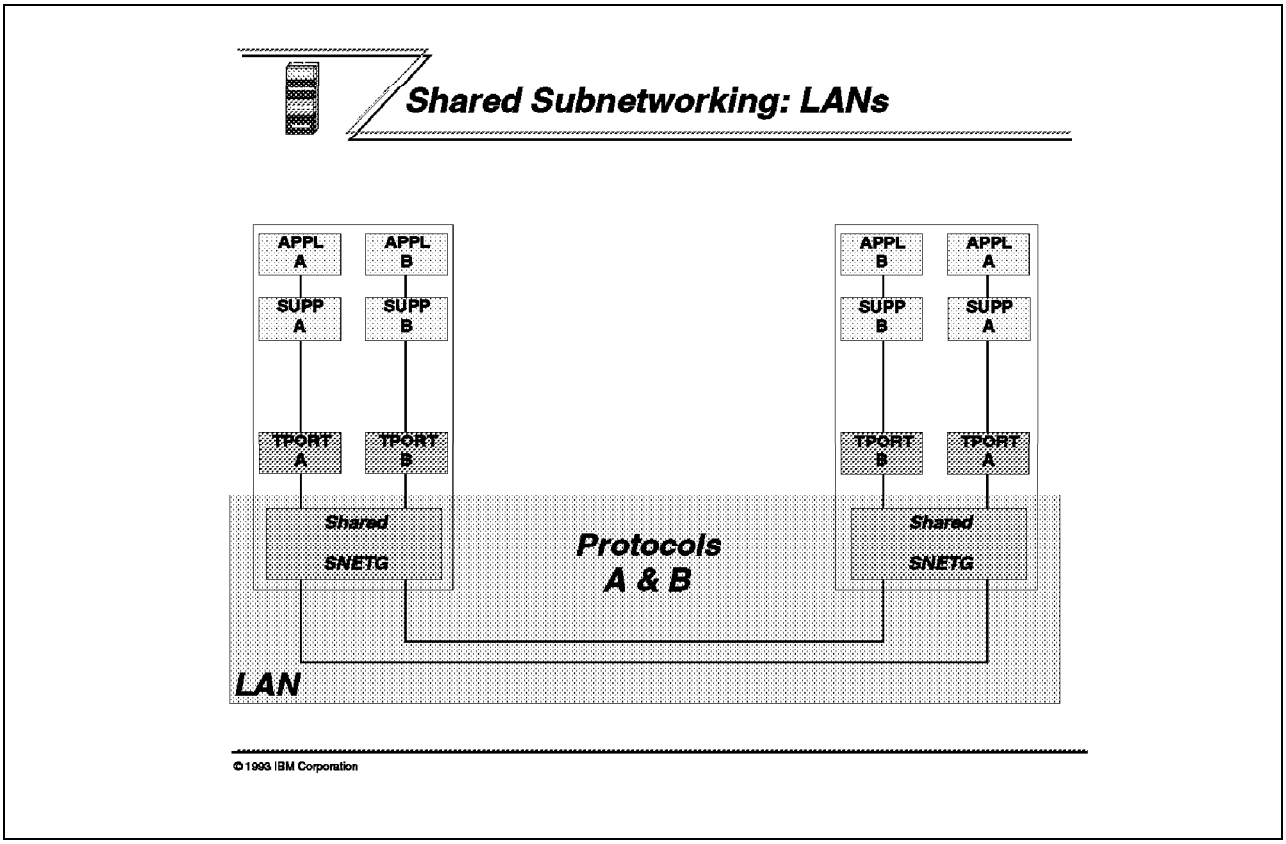



*Figure 38. Shared LAN (Multi-Protocols) Technique*

LANs, such as Token Ring (IEEE 802.5), Ethernet V2, IEEE 802.3, and FDDI, permit the sharing of a subnetwork. All participants on a LAN share the LAN's high bandwidth by following the access protocol for that particular LAN type. For token ring and FDDI, this access protocol is based upon a token-passing algorithm; for both Ethernet V2 and IEEE 802.3, a collision-detection scheme called CSMA/CD is used to permit access to the LAN. Since LANs are only defined through OSI layer 2 (Data Link Control), any higher-level transport protocol can utilize this subnetwork. As the term local area network implies, the nodes must be in close physical proximity to each other, usually within the same building or campus not requiring the crossing of a right-of-way. In Figure 38, Appl-A in the left node must communicate using Transport A with its partner in the right node, and likewise for Appl-B. Note that Appl-A does not interoperate with Appl-B.

LANs have historically been used to connect the workstations of people within an organization who need to communicate frequently together and share data and resources, such as high-speed printers. Usually these LANs are created along departmental lines, in the belief that people within a single department need primiarly to communicate with one another. This is often referred to as “workgroup computing.”

Although the applications will probably not be directly affected by adding a LAN, additional hardware and facilities must be purchased to set up the LAN. Wiring between nodes must be installed, each workstation must have a network interface adapter card for the specific LAN protocol, additional LAN hardware

may need to be purchased (such as multistation access units for token ring or transceivers for Ethernet), and software might also need to be purchased to enable particular transport protocol(s) to utilize the adapter card.

**Advantages of Shared Subnetworking: LANs**

- Permits the sharing of a single subnetwork by multiple protocols, therefore reducing physical network costs, such as wiring, adapter cards, and other LAN equipment (for example, transceivers for Ethernet, multistation access units for token ring).
- Can utilize shared personnel resources, since a single group of people can handle the administration and network management for all applications on the network.
- Does not require special devices to establish logical connections between end nodes; no intermediate routing nodes are required. The MAC-level protocol determines how to locate particular nodes, and how to transmit the data to that node and this MAC protocol is built into each adapter card.

**Disadvantages of Shared Subnetworking: LANs**

- LANs are restricted to limited distances.
- May require the sharing of a single network interface card for multiple transport protocols, which might impact the performance for a particular application.
- The use of a shared medium sometimes has an impact on application throughput. If one application is currently utilizing the network (for example, for a TCP/IP FTP file transfer), then poor performance might be noted for another application (for example, a 3270 emulation session).
- If the LAN is not operable for some reason, users for both Appl-A and Appl-B might be unable to function.

## 2.2.4 Shared LAN (Local Bridge) Technique

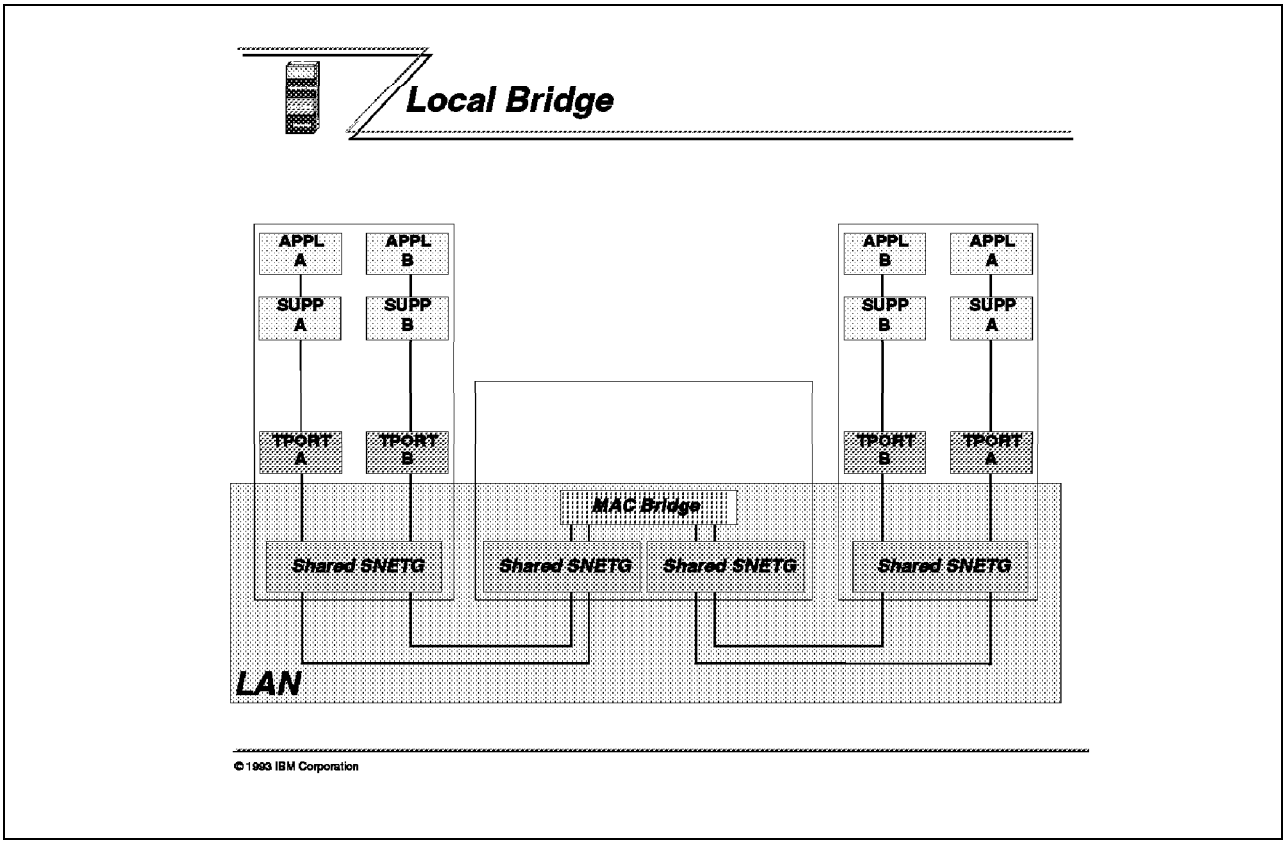



*Figure 39. Shared LAN (Local Bridge) Technique. Two LANs, with One Bridge*

As mentioned in the previous LAN description, all LANs operate at OSI layer 2, the Data Link Control layer. The Media Access Control sublayer of the DLC defines the frame formats, addressing, and access protocols for each particular LAN type. If two local LANs need to be connected to enable communication between similar applications, then a local bridge can be used. A local bridge operates at the MAC level, and is often referred to as a **MAC bridge**.

In Figure 39, if the two nodes shown are located on two separate LAN segments (perhaps these LANs belong to two different departments of a company within a building), and Appl-A on the left node needs to communicate with Appl-A on the right node, physical connectivity must be arranged. As shown in this diagram, a MAC bridge has been chosen to provide this connectivity. The MAC bridge will be able to detect that frames need to be forwarded from the left-hand segment to the right-hand segment. The bridge makes this determination via its “bridging protocol,” such as source route bridging for token-ring LANs, and transparent bridging for Ethernet LANs. If the two LAN segments are of the same type, such as two token rings, then this forwarding can be extremely fast since the frame formats and the bridging protocol are identical on both sides.

However, some local bridges also perform some amount of conversion so that different segment types might be able to be connected, such as a token ring to a Ethernet V2. In this case, the frame formats are different and conversion needs to take place. Also, the bridging protocols might differ. As long as the conversion takes place strictly at OSI layer 2, one can consider this function to

be mainly a “bridge” and not a “router” function. The IBM 8209 bridge performs this function for token ring and Ethernet LANs.

Although a local bridge will need to be added to provide this functionality, there will be no impact to the end nodes. The bridge will need to attach to each LAN segment as a regular device on the LAN segment, and thus there needs to be planning to ensure that it can be attached to both LAN segments. For example, if the local bridge were connecting two token rings, it must be positioned in the building so that the wiring for both LAN segments can reach it, plus it must attach to the multistation access unit (MAU) of Token Ring #1 and the MAU of Token Ring #2.

**Advantages of Local Bridge**

- Since bridges operate at OSI layer 2, they are not sensitive to the choice of transport protocols. Thus, bridges can forward frames for all transport protocols.
- Bridges can forward frames very fast since little, if any, conversion needs to take place.
- Bridges can be used to isolate segments of a large “LAN” in order to improve overall performance, where two segments define their own logical workgroups.

**Disadvantages of Local Bridge**

- Since bridges are not transport protocol-sensitive, they are unable to detect problems. For instance, with TCP/IP networks, it is quite possible to get “broadcast storms” where a single user might flood the network with extraneous messages. Routers, which are sensitive to the transport protocol, can prevent such problems from occurring.
- Inherent limitations of the bridging protocol might prohibit extensive connectivity. For instance, source route bridging is limited by the implementation of only seven sequential bridges in the network (seven “hops”); transparent bridging is limited by a single active bridge at any time between two segments. These limitations will affect the design and extensibility of any large LANs.
- Since all traffic can flow over the bridge, there is no organizational separation by addresses or subnets.

## 2.2.5 Shared LAN (Remote/Split Bridge) Technique

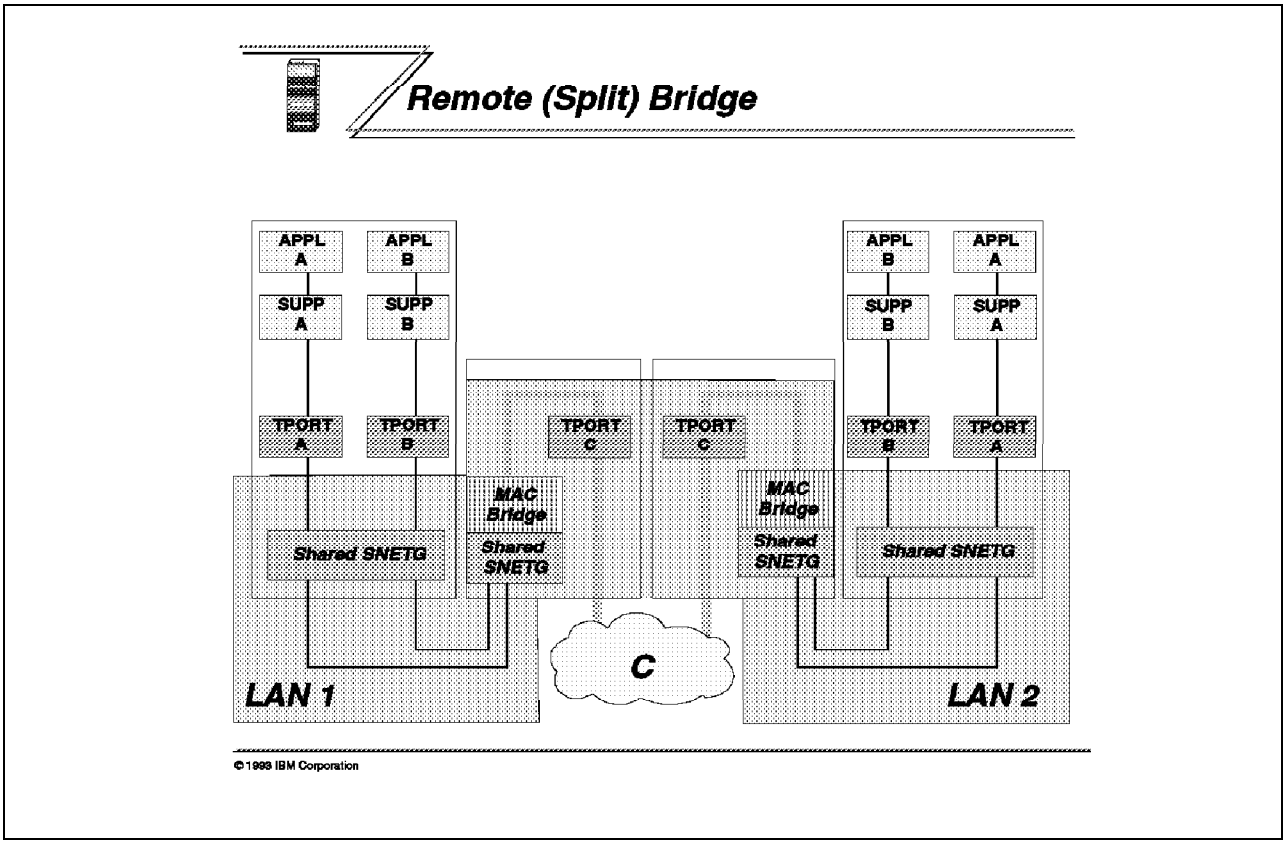



*Figure 40. Shared LAN (Remote/Split Bridge) Technique. Two LANs, with Two Bridge Parts*

This technique is a special case of extending LANs with bridges over a wide area network. These bridges are referred to as **remote** or **split** bridges. The essential feature of this technique is that the frames are taken from the LAN at the MAC sublayer of OSI layer 2; these frames are then sent as data across the wide area network, where they are reconstituted as MAC frames on the target LAN. An encapsulation approach is typically used by the bridges to send these frames across the wide area network. A connection is established between the two remote bridge partners, which multiplexes all of the traffic between the two LANs. In fact, a single remote bridge may connect multiple LANs, and can multiplex for all these LANs over a single connection to a partner bridge; this is referred to as multi-port bridging.

As with other encapsulation techniques, the connection protocol between the partner bridges is usually proprietary. As shown in Figure 40, these bridges are shown to be true MAC bridges for taking traffic off of the attached LAN; then an extra protocol stack is involved to provide the encapsulation function across the wide area network utilizing transport protocol C. Depending upon the implementation of Appl-C, which performs the encapsulation/de-encapsulation, there may be some filtering of LAN traffic to prevent the flooding of the intermediate wide area network with unnecessary broadcasts (such as the broadcasts for locating a session partner with token ring). This type of filtering is performed only at the Data Link Control layer (OSI layer 2), and rarely do these remote bridges filter at the upper layers (layer 3 and above); thus they are transport protocol-independent. Therefore, both Tport-A and Tport-B in the diagram can be sent over the remote bridges.

Remote bridges occur in pairs (hence the name *split bridge* to indicate the "split" between the two partners). Point-to-point connections are required between the bridges. Some implementations permit a single physical bridge to have multiple connections, but it is always in a point-to-point manner with partners. The IBM Remote Token-Ring Bridge/DOS is an example of such a split bridge.

There are usually no changes to the end nodes or the applications. Additional hardware and software will be needed to provide the remote bridge function; also, these devices must be able to attach to their respective local LANs. Between the partner bridges there is usually a leased line.

**Advantages of Remote (Split) Bridge**

- Transport protocol is independent.
- Some filtering of LAN broadcast messages may be done, therefore cutting down on wide area network traffic.
- These remote bridges are usually inexpensive, and show good performance for the given link speed between the remote bridges.

**Disadvantages of Remote (Split) Bridge**

- Adequate filtering may not be done by the bridges at the Data Link Control layer or higher layers, with the result that extraneous traffic may flood the intermediate wide area network. This extraneous traffic may impact critical session timers for certain sensitive transport protocols (such as SNA and NetBIOS), causing sessions to be dropped.
- Since the encapsulation approach and session protocol between the partner bridges is often proprietary, all equipment must be procured from a single vendor. This may change, however, as new standards such as PPP become more popular.
- It may be expensive to connect every bridge physically to every other bridge in a point-to-point manner over leased line facilities.
- There may be limited, if any, flow control over the lower-speed link between partner bridges.
- There are limitations to extensibility and/or scalability of these networks, due to hop-count limitations (as for token ring) and/or problems caused by the sheer number of potential partners (end stations and/or bridges).

## 2.2.6 Encapsulation Techniques

The general principle of encapsulation is to “mis-use” a program-to-program logical connection as a physical link, as shown in Figure 41.

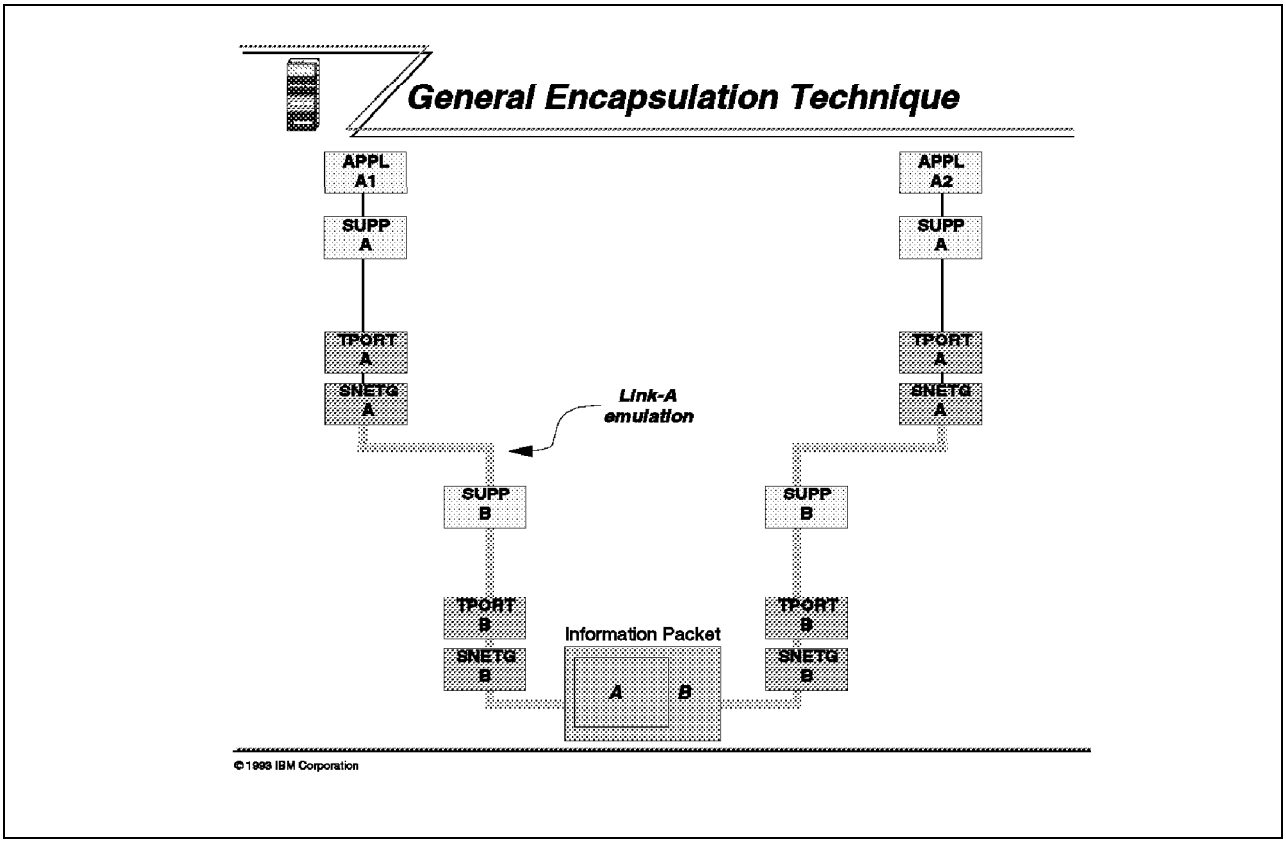



*Figure 41. General Encapsulation Technique*

In the above diagram, the facilities of B are used as an emulation of link-A; the 3-group stack, SUPP-TPORT-SNETG, has been used twice, once in its normal manner at the top of the diagram and once as a substitute for a physical link at the bottom of the diagram. In the encapsulation diagrams that follow, you can mentally “stretch the picture from top to bottom” following the darker line and see an uncollapsed flow similar to that in Figure 41. We have merely collapsed each diagram (vertically) to conserve space and line up the layers.

The information packet identified at the bottom of the above diagram reflects the fact formats are encapsulated along with information specific to the protocol. The networking control headers for the packet-A are encapsulated within the networking control headers for packet-B. That is, one packet-B usually contains one packet-A (or more).

### 2.2.6.1 End-to-End Encapsulation Technique

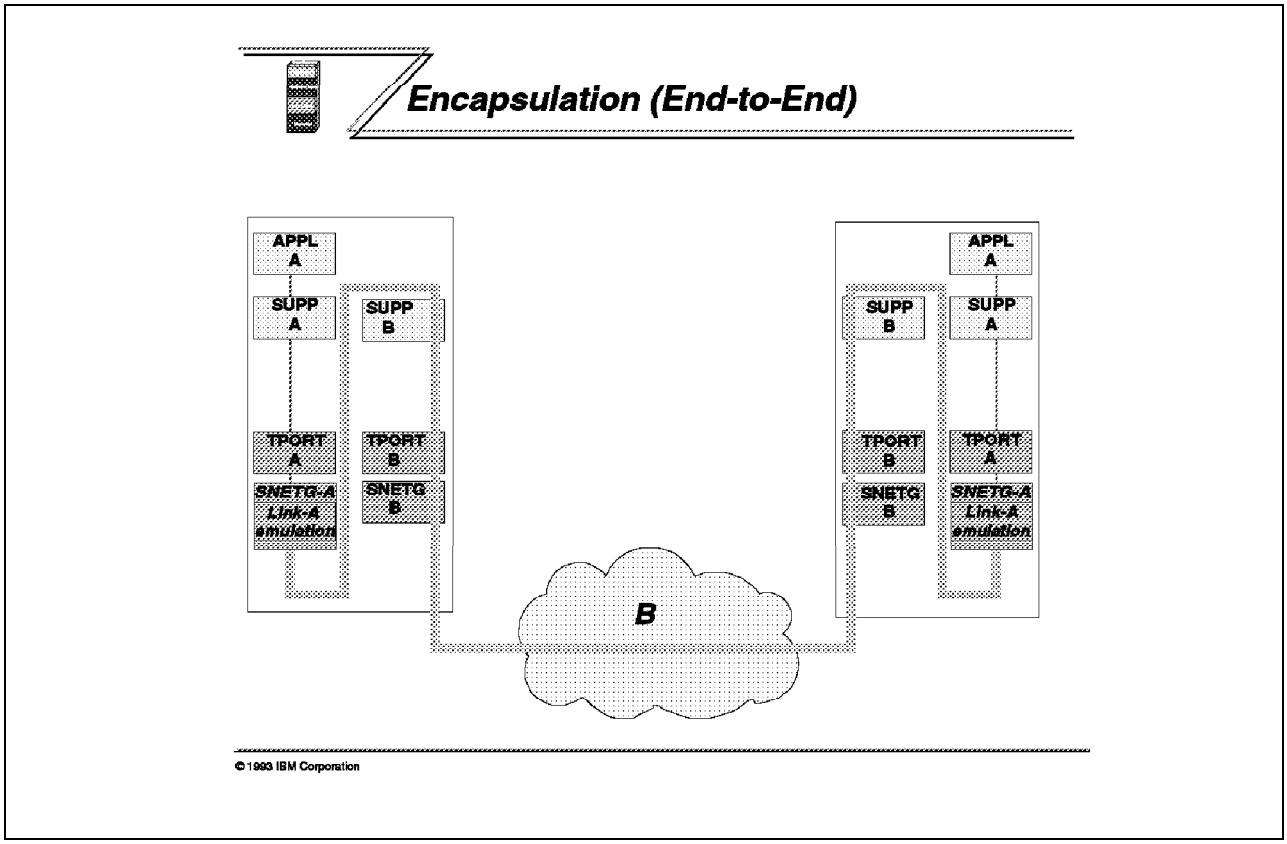



*Figure 42. End-To-End Encapsulation Technique*

In the above diagram, services[16] of type 'A' are found in each of the bottom three program groups (layers). That is, service of type A is found in the application support program group (that is, communication support). Likewise, service of type A is found in the transport network program group (for example, TCP/IP transport network service). In other words, the entire protocol stack is labeled with 'A' (APPL-A, SUPP-A, TPORT-A, and SNETG-A). These labeling conventions are followed in later diagrams in this book.

This technique encompasses a whole range of possibilities, as it can be used to describe many aspects or methods of mixed-protocol networking. In current implementations it is limited to those techniques used for “tunneling” through one transport network. From the viewpoint of the application and its protocol stack, the second transport network is being used as subnetwork. This subnetwork is often referred to as a “link,” since it is usually a physical line into a wide area network. From the viewpoint of the intermediate network, the originating network's “link” is just another user session. If Appl-A in the left node of Figure 42 wished to send data to Appl-A in the right node, encapsulation is necessary to traverse the intermediate network, which supports only transport protocol B. Appl-A in the left node would generate its data as normal, and it would proceed through the usual protocol stack for Appl-A. However, instead of encountering a real subnetwork layer, the subnetwork would be emulated by a special program before the entire packet (Appl-A's data,

[16] The terms services and application support (SUPP) are often used interchangeably.

transport protocol information, and emulated link information) is sent to the application programming interface (api-B) of the second protocol stack in the same node.

How this second protocol stack treats the Appl-A packet can vary quite widely in implementation. Sometimes the Appl-A packet is treated simply as data, and no attention is paid to the particular needs of the Appl-A transport or subnetwork levels. In other cases, the encapsulating protocol may be very sensitive to these requirements. In either case, the Appl-A packet will be "enveloped" or "encapsulated" with the appropriate transport and subnetwork headers and trailers necessary to traverse Network B. A session will be established between the encapsulating protocol stack and its partner de-encapsulating protocol stack across the network to ensure reliable delivery of these packets.

Once this packet reaches the destination in the node on the right side of the diagram, these headers and trailers are removed, thereby *de-enveloping* or *de-encapsulating* the original Appl-A packet. The "link" A emulation functions are completed, and then the packet is returned to the native Appl-A protocol stack. The partner Appl-A applications believe that they have just communicated over their "native" transport network. The encapsulating/de-encapsulating session partners can often multiplex traffic from many user sessions.

Encapsulation always requires a pair of nodes to cooperatively perform the enveloping and de-enveloping steps. These can be end nodes, a combination of an end node and a gateway, or two gateways. Figure 42 on page 62 shows a scenario where the encapsulation and de-encapsulation are performed in end nodes over a single transport network.

In Figure 43 on page 65, the target applications are in different networks. Appl-A in the left node is a TCP/IP-based application, which must traverse Network B, which is SNA, to reach its partner application in the right node, which resides in Network A. Connecting Networks A and B is a gateway, which utilizes an encapsulating methodology to transport the TCP/IP traffic over the SNA network. If Appl-A in the left node sent data to Appl-A in the right node, the encapsulation would take place in the left node, and de-encapsulation would take place in the gateway. The right node would only have the native application. The IBM SNALINK function utilizes this approach, where the left node and the gateway are hosts in Figure 43 on page 65.

A double-gateway approach is illustrated in Figure 44 on page 66. This type of approach is very common with routers that are attached to LANs, where the router is acting as the gateway. The routers will communicate over their own network, which may use a different transport protocol than the applications. The applications use their native transport protocol on their local network, and the router detects that the traffic is destined for a remote location. The local router will encapsulate this packet to transmit it over the router backbone network, and the remote router will de-encapsulate the packet before sending it to the destination node. This "tunneling" approach has been used for transmitting SNA traffic between IBM 6611 or cisco** routers. IBM's Non-SNA Interconnect (XI) product, which runs in 37XX Front-End Processors running NCP, routes X.25 traffic over the SNA backbone network using this technique.

Since most encapsulation techniques utilize a full protocol stack to perform this function, there is a private application-to-application protocol being used between the encapsulating and de-encapsulating nodes. This private protocol is often proprietary to a given vendor. Depending upon which of the three

configurations is utilized, additional software may need to be added to the end nodes, or gateways may need to be purchased, but the applications should not need to be changed.

**Advantages of Encapsulation**

- No modification is required to the application (Appl-A in the examples) or the native networks.
- Only one network protocol will be in use. In large networks this substantially simplifies management of the network, reducing cost in management tools and personnel.
- Encapsulation is usually a software-based solution, so the cost may be kept low.

**Disadvantages of Encapsulation**

- Encapsulation techniques are often proprietary, limiting the user to a single vendor for equipment or software.
- Full protocol stacks are required in the end nodes. If encapsulation takes place in the end nodes, as in Figure 42 on page 62, then two protocol stacks are present in this end node: the stack for the application, and the stack for encapsulation/de-encapsulation. This can be costly or even impossible in terms of system resources, such as memory, processor utilization, and storage costs. For example, this may not even be feasible on a PC DOS machine, where there is a 640KB memory limitation. These same concerns apply for the end node in the single-gateway configuration in Figure 43 on page 65. If a gateway is involved, then additional hardware or software may be required to provide this encapsulation function.
- Encapsulation implementations vary widely; sometimes there is no attempt to filter any of the traffic from the emulated link; all packets are sent *unaltered*. This can cause problems if neither the application's subnetwork nor transport network requirements are met. For instance, if Appl-A is running on a SDLC link, then SDLC has certain requirements in terms of acknowledgement protocols and the timing of these acknowledgements, which the encapsulation method may not realize, and thus dropped sessions may result.
- There is increased overhead in each transmission in terms of headers and trailers. For instance, in Figure 43 on page 65, in the left node at point #1, the headers and trailers necessary for transmission through an SNA network are added to the TCP/IP packet. This extra overhead can increase network load requirements, and potentially affect the performance of the traversed network.
- From an application viewpoint, performance may suffer due to the additional work that needs to be done on each transmission to provide the encapsulation and de-encapsulation.
- The characteristics of the network used to carry the encapsulated traffic may not match the requirements of the application. For example, encapsulating SNA traffic on a TCP/IP network may result in degrading or defeating the SNA priority scheme (class of service).

### 2.2.6.2 Single-Gateway Encapsulation Technique

Encapsulation can be accomplished through a single gateway, one side only, as shown in Figure 43.

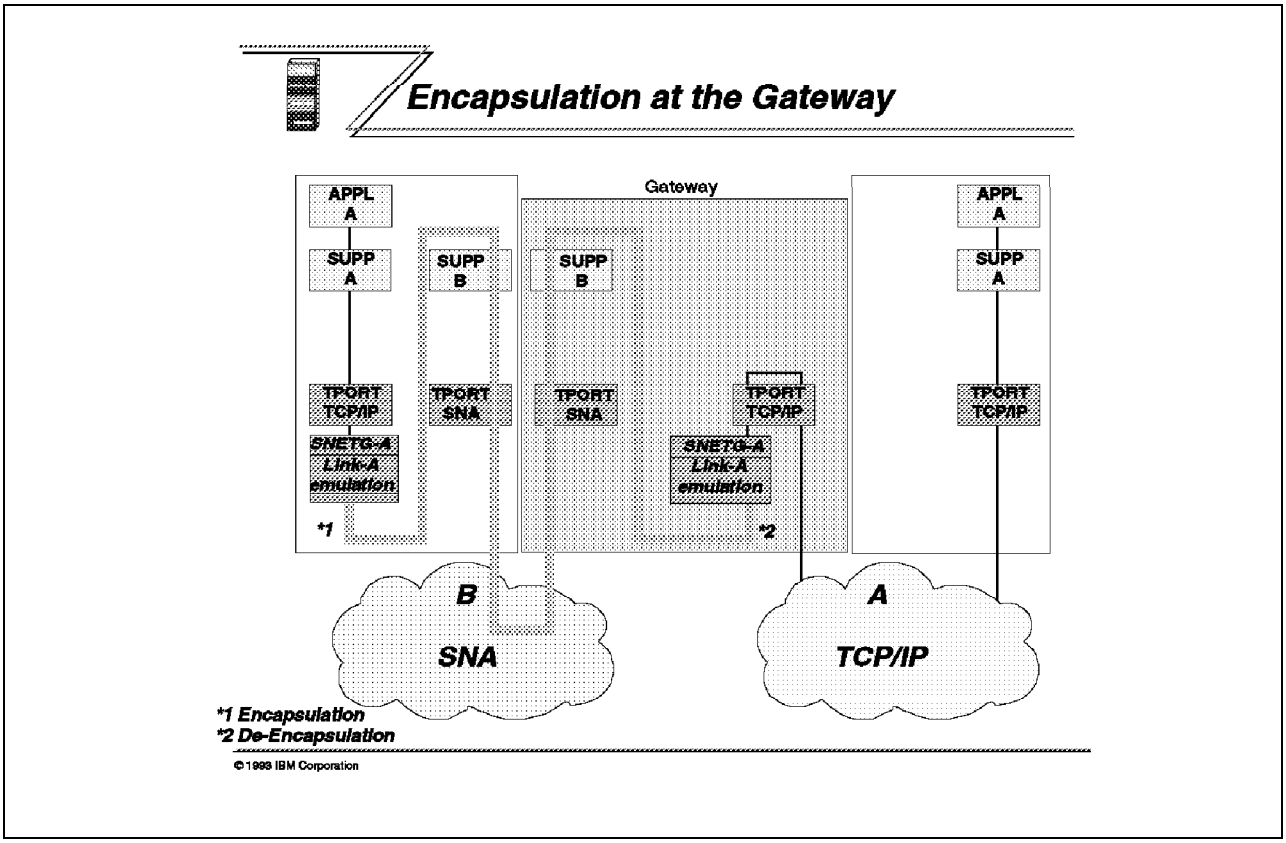



*Figure 43. Single-Gateway Encapsulation Technique*

In the above diagram, Application-A at the upper left-hand corner is connected to the gateway in the middle by an emulation of a type-A (IP) link over a type-B (SNA) transport network. Application-A is connected through the single gateway in the middle to its partner Application-A at the upper right-hand corner.

### 2.2.6.3 Double-Gateway Encapsulation Technique

Encapsulation can be accomplished through two (or even more) gateways, in a series, as shown in Figure 44.

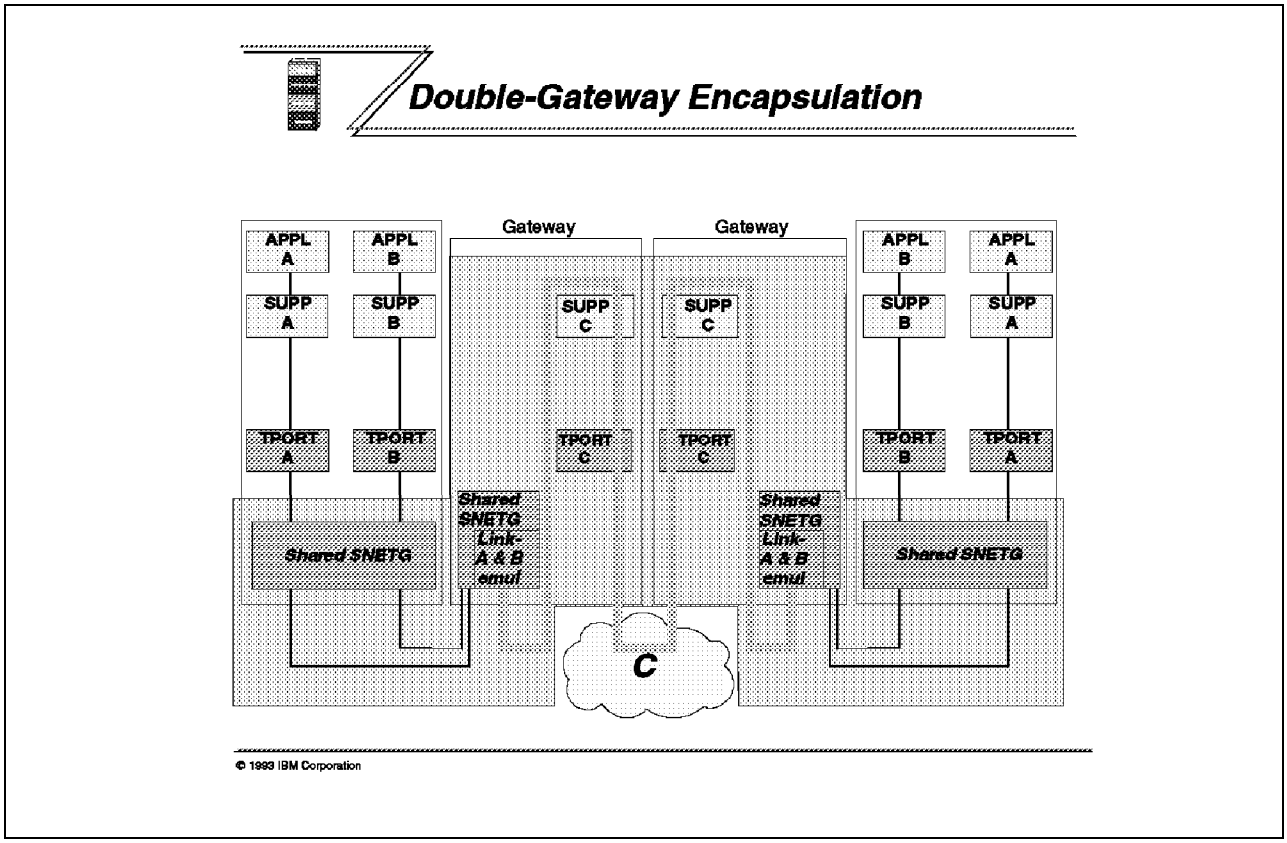



*Figure 44. Double-Gateway Encapsulation Technique*

In the above diagram, Application-A and Application-B at the upper left-hand corner are both connected through the two gateways in the middle to their partners at the upper right-hand corner. The type-A link and the type-B link are both emulated over the type-C transport network. An example of this is the encapsulation of SNA and NetBIOS over a TCP/IP network.

## 2.2.7 Data Link Switching (DLSW) Technique

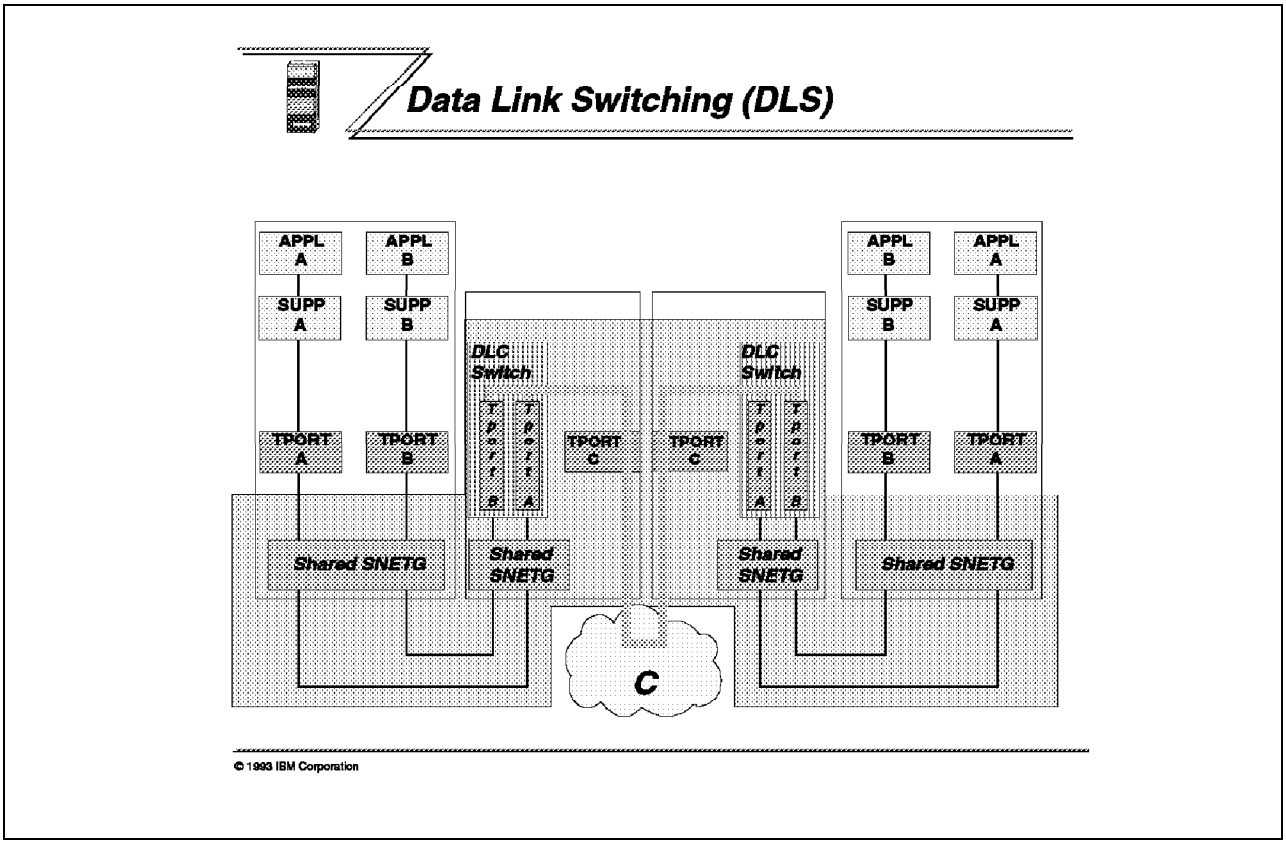



*Figure 45. Data Link Switching (DLSW) Technique*

Data Link Switching (DLSW) is an example of *filtered encapsulation at the transport level*. It is fundamentally an encapsulation technique as described previously, but for specific transport protocols, a great number of protocol-specific functions are handled to reduce unnecessary traffic over the intermediate wide area network, and to handle specific timing situations. In fact, many packets are suppressed over the wide area network, and compensations are made locally to provide necessary functions.

DLSW is defined for carrying SNA and NetBIOS traffic over TCP/IP networks. At the transport level, NetBIOS broadcasts to locate session partners are intercepted and suppressed. Data Link Switching thus must capture and maintain data about the requests, and the location of partners, so *session connections can be emulated across the wide area network*. At the Data Link Control level, there is some traffic which is timing-dependent. Data Link Switching *terminates* this DLC; that is, it locally emulates a response to a timing-dependent message that an end node is expecting. Examples of this DLC termination are the handling of polling-type messages or “keep-alive” messages. If these messages are sent over a wide area network without this type of interception and emulation, the slower-speed wide area network will often cause these timing windows to be exceeded, resulting in hung or dropped sessions.

As shown in Figure 45, the routers that implement Data Link Switching utilize an application to provide the encapsulation plus the specific transport and data link control functions. There must be partner Data Link Switching nodes to provide these functions, and a session is established between these partner applications.

Unlike other proprietary encapsulation techniques, the Data Link Switching technique has been submitted as an informational RFC, and thus is available to the vendor community for implementations. Currently the IBM 6611 router provides this capability, and many other router vendors have indicated that they will be providing Data Link Switching in the future.

There is no change to either the end nodes or the applications for this technique. Routers that implement this data link switching technique must be purchased.

**Advantages of Data Link Switching**

- Data Link Switching filters most of the unnecessary transport and data link control messages, thereby reducing wide area network traffic.
- It compensates for transport- and data link control-level timing dependencies, thus keeping SNA and NetBIOS sessions up in these timeout situations.
- A published specification detailing this technique has been submitted as an informational Request For Comment (RFC 1434) and thus, this technique can be used by multiple vendors in router products.
- DLSW has techniques to accomplish congestion control at both the local and global levels, thereby easing the load on the backbone network.
- DLSW will “bundle” packets together for a single destination, thereby saving on transmission headers and network overhead.

**Disadvantages of Data Link Switching**

- Data Link Switching is transport- and data link control protocol-specific. Current implementations exist for NetBIOS and SNA over token ring, Ethernet, and SDLC connections, using TCP/IP as the backbone transport.
- The DLSW specification does not define some advanced functions, such as priorities, adaptive pacing, and expedited flows.
- Since there is no prioritization in the DLSW specification, response times may be inconsistent even if sessions are not dropped.

## 2.2.8 Packet Interface (Using Frame Relay) Technique

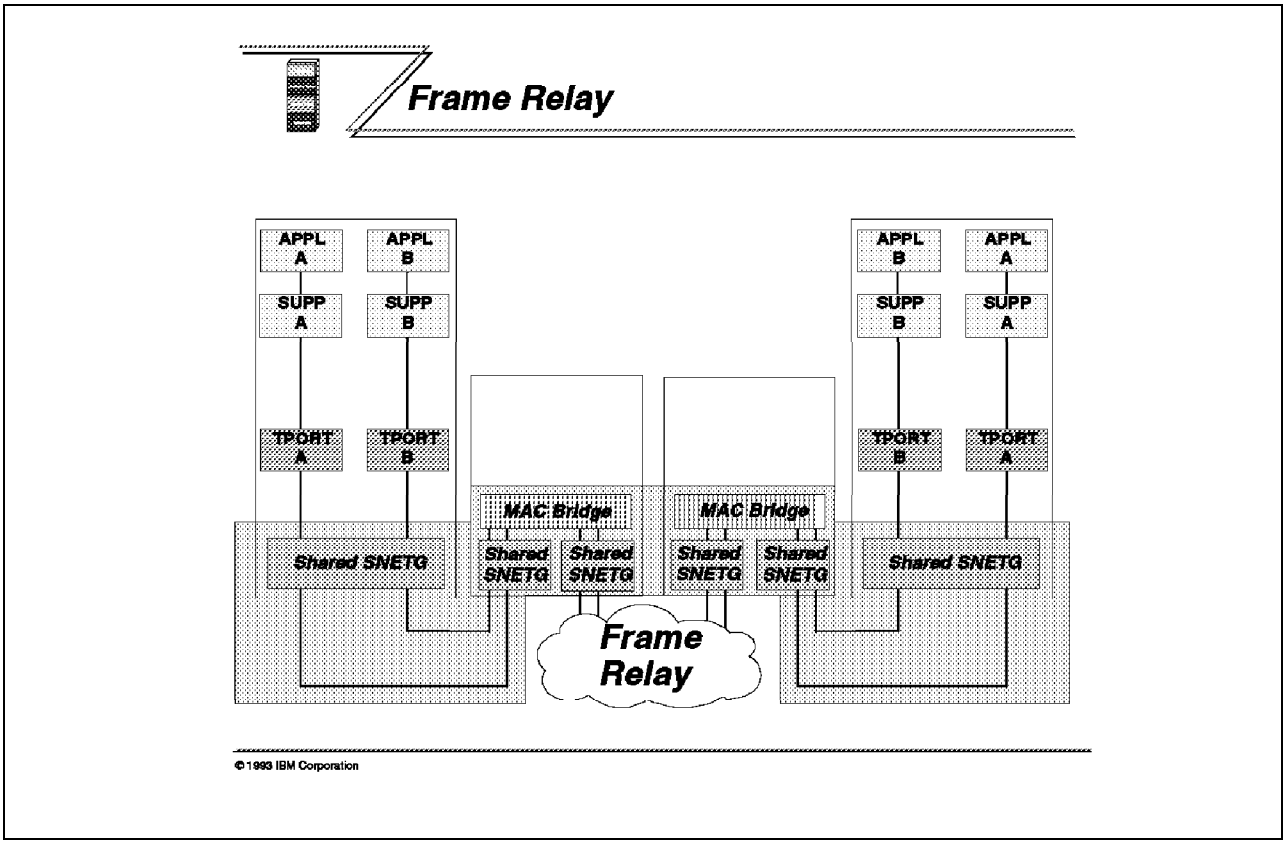



*Figure 46. Packet Interface Technique. Using Frame Relay*

Many of these technologies operate at OSI layer 2. Examples of these technologies are Frame Relay (shown in Figure 46), SMDS, and ATM/Cell Relay. Although it operates through OSI layer 3, X.25 is also in this category. For all of these technologies, there is always a device that will take the traffic from the local subnetworks, such as LANs or serial lines, and convert it to a format understood by the wide area network (an interface protocol). The data will be taken from the subnetwork, and packaged into *packets* before being sent over the wide area network. The size of these packets may be variable (as with Frame Relay or X.25) or it may be a fixed size (as with SMDS or ATM). The wide area network, such as the Frame Relay network illustrated in Figure 46, consists of a set of switches that understand this interface protocol, and routes the data to the appropriate final location. The routing algorithm between the switches may not be defined by standards and therefore may be proprietary to the switch vendors (for example, Frame Relay) or may be completely defined by standards (for example, ATM). At the final location, there must also be a device that understands the same interface protocol, which will take the data from the wide area network and deliver it to the end node.

As shown in this diagram, the devices that implement the Frame Relay interface act like a MAC bridge, taking the data from the shared local subnetwork, and forwarding these frames onto the Frame Relay wide area network. Both transport protocol A and transport protocol B can share the same Frame Relay network, but Appl-A can still only communicate with Appl-A, and Appl-B can only communicate with Appl-B.

The equipment at the customer locations that understand the interface protocol are often routers. The end nodes and the applications are not usually impacted. The customer must decide on whether to utilize a public network service for the wide area network, or to build a private network by procuring switches and using leased lines.

**Advantages of Frame Relay**

- The wide area network becomes transport protocol-independent.
- It can also provide a “mesh” network, permitting any location to communicate to any other location.

**Disadvantages of Frame Relay**

- The wide area network may not have a prioritization scheme to allow more important traffic to take precedence over less important traffic.
- Traditional packet switching, using the X.25 interface standard, often had severe performance impediments. Fast packet switching, which includes Frame Relay and cell relay (ATM), has much better performance, but at the cost of error checking and recovery in the network. For Frame Relay, error recovery must be performed at the devices that interface to the network. How much error recovery is actually performed in these nodes can vary greatly depending on implementation.

## 2.3 Transport Network Technologies

Transport network technologies are implemented at the higher layers (OSI layers 3 and 4) of a network.

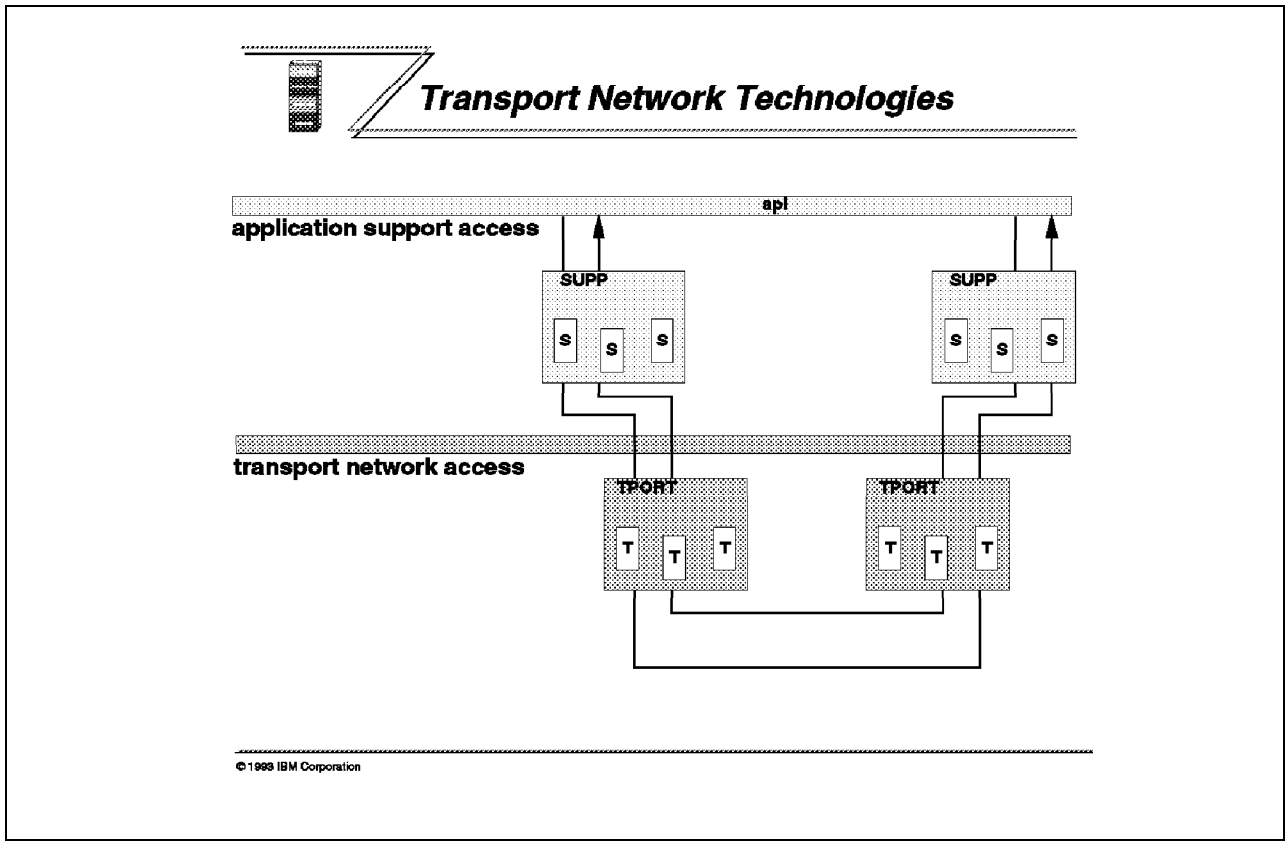



*Figure 47. Transport Network (TPORT) Technologies*

Some of the technologies implemented within the transport network layer (program group) are:

- Multi-Protocol Router
- Mixed-Protocol RFCs
- Multi-Protocol Transport Network (MPTN)

## 2.3.1 Multi-Protocol Router Technique

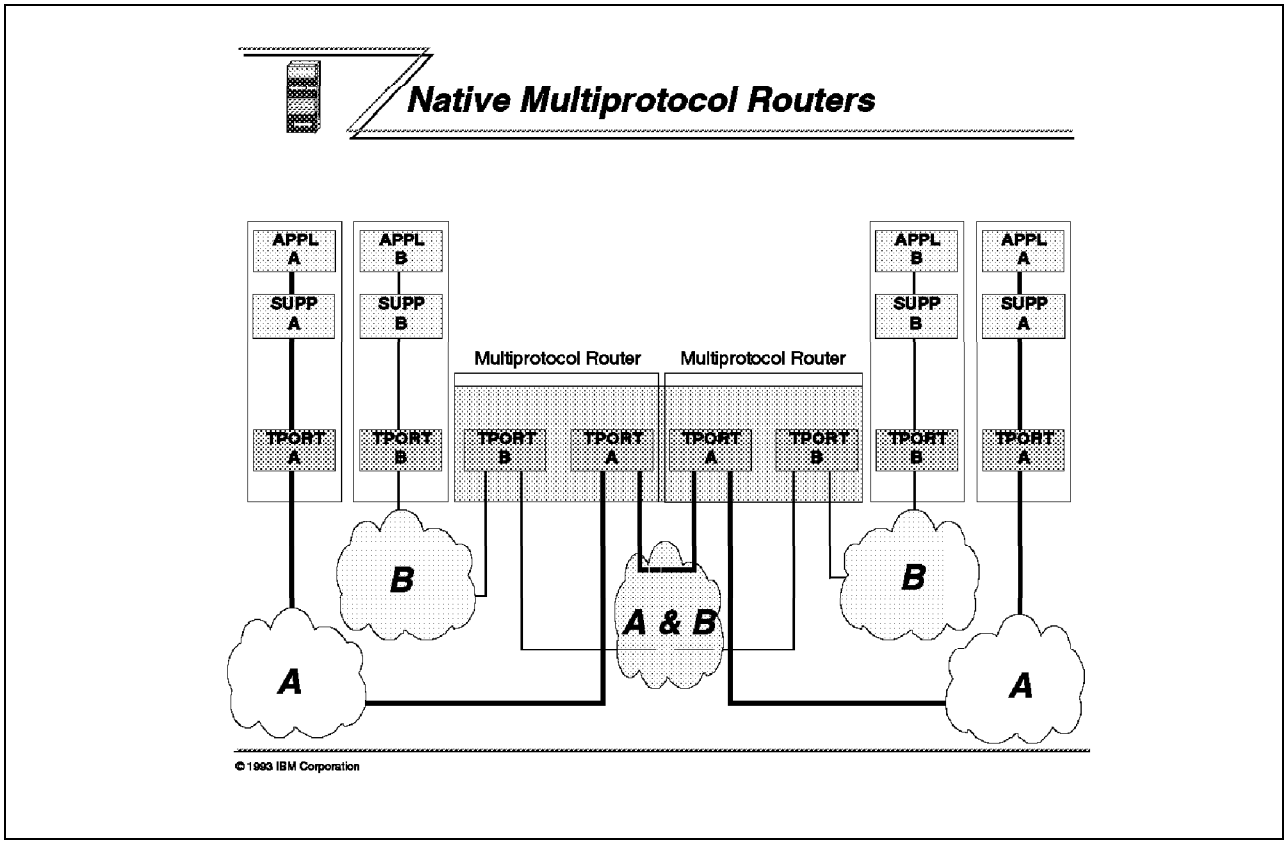



*Figure 48. Multi-Protocol Router Technique. Native Multi-Protocol Routers*

The essence of this technique is that each protocol is handled separately from end to end; all intermediate node routing is done in a protocol-specific way. Each router in the network must utilize the transport levels for each protocol for which it has support. As shown in Figure 48, the routers contain both transport layers, Tport-A and Tport-B. Appl-A in the left node uses Tport-A to communicate with its partner nodes indicated by the A network on the left side. When Appl-A wishes to communicate with its partner Appl-A in the right node, three networks are actually crossed: the A network on the left of which the local router is connected, the A&B network for the routers, and the A network on the right to reach the desired end node. The same transport protocol is used throughout the data transmission. The transport protocols do not interact; essentially they share the transmission facilities in the case of the A-B network between the routers. This is often referred to as “ships-in-the-night” routing.

How these protocols share the transmission bandwidth in the network between the routers can vary widely. There are some approaches that are standards-based, such as the PPP protocol, but many routers also use a proprietary algorithm. The routers communicate among themselves utilizing a particular “routing protocol” to keep track of possible paths through the network and the status of available routes. These routing protocols may be standards-based, such as RIP and OSPF, or proprietary, such as cisco's IGRP.

Figure 48 illustrates *native multi-protocol routers*: that is, each protocol will be transmitted over the router network A&B in its own native mode, without modification. Using this approach, if there are 10 transport protocols to be

transmitted over the intermediate router network, then 10 transport protocols will exist natively in that network. Many routers available today also encapsulate certain protocols with another protocol before transmission through the router network occurs. Encapsulation techniques were discussed earlier in 2.2.6, “Encapsulation Techniques” on page 61.

There are many examples of routers. For SNA networks, the IBM 3745 and NCR Comten products provide this routing capability. For multi-protocol networks, the cisco and Wellfleet product lines, as well as the IBM 6611, provide this functionality.

**Note:** 3745/NCP now also “natively” routes IP.

Routers can attach to the local subnetworks as another device. There is usually no impact to the end nodes or to the applications. Transmission facilities may need to be procured between the routers if wide area connectivity is required.

**Advantages of Native Multi-Protocol Routers**

- Each transport protocol is treated natively.
- Since routers are aware of the protocol intricacies of each transport protocol, they can perform some important filtering techniques, which prohibit extraneous messages, such as TCP/IP broadcast storms, from flooding the network.

**Disadvantages of Native Multi-Procotol Routers**

- Many transport protocols will exist on the intermediate router network. Each transport protocol that is supported requires its own network management. There are also implications in terms of configuration and change management, and address and directory access.
- The multi-protocol router may not be able to mix different vendors' routers in a single network, if proprietary point-to-point transmission techniques, routing protocols, or encapsulation techniques are used.
- Multiple transport protocols complicate network design and planning.
- Additional bandwidth may be required to accomodate the multiple transport protocols efficiently.

## 2.3.2 'Mixed-Protocol Standards for TCP/IP' Technique

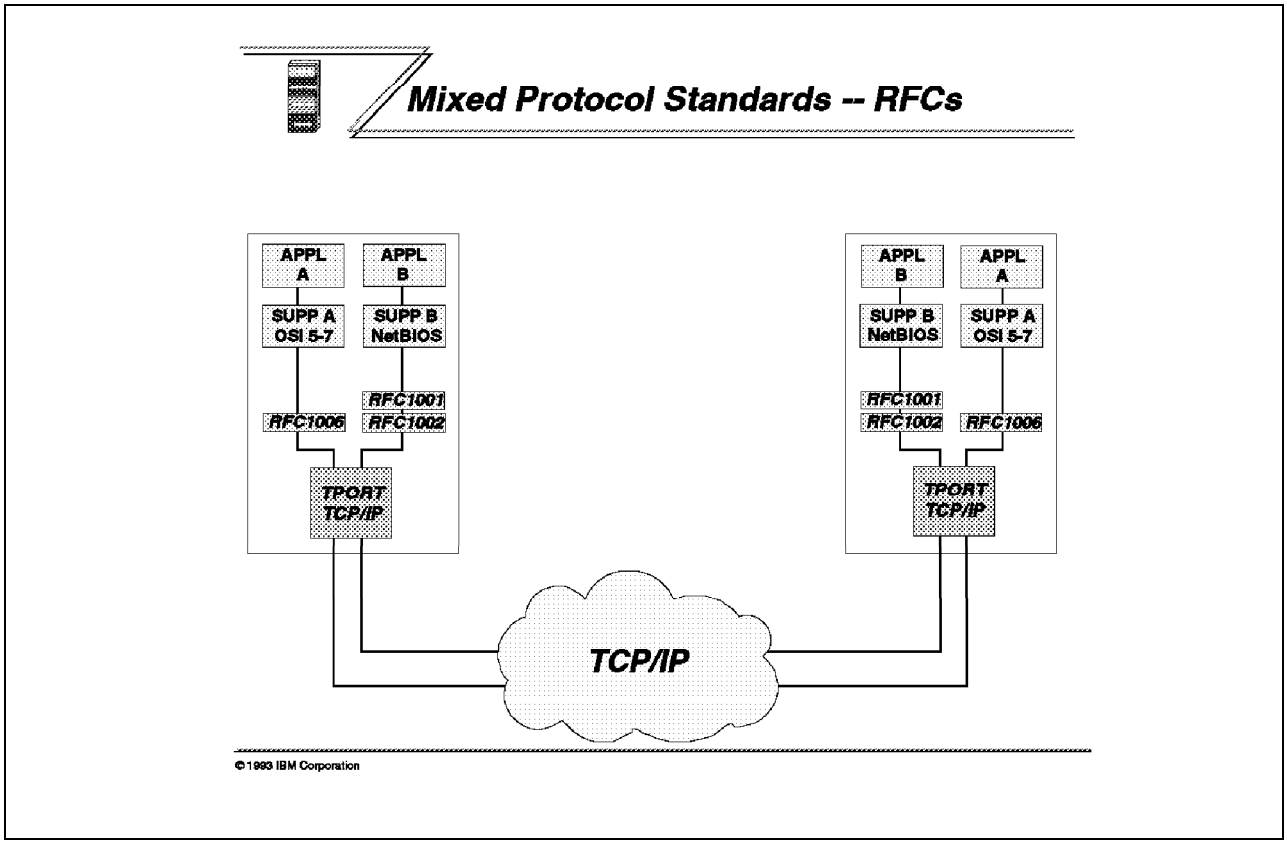



*Figure 49. 'Mixed-Protocol Standards for TCP/IP' Technique*

As described earlier in Chapter 1, “Fundamental Concepts of Technologies” on page 1, application programming interfaces and application services have historically been tied to a particular transport protocol. Thus, applications utilizing the upper OSI layers (layer 5-7), usually had to unconditionally run over an OSI transport network. There are some techniques, often referred to as a “hybrid stack” or “mixed-protocol stack” approaches, where the protocol stack to support a particular application is built from parts of two different protocol stacks. For instance, a mixed-protocol stack allows an OSI application to communicate over a TCP/IP network to a partner mixed-stack with the identical OSI application.

There are a variety of mixed-stack approaches that exist, but only a few are standards-based. There are a set of mixed-stack standards definitions for specific protocols, such as with MAP/TOP and the Internet Request for Comments (RFCs), which are discussed later in this section. There is also a more generic approach to this mixed-stack situation, which the forthcoming MPTN standard addresses.

RFCs are documents that describe a need, provide information, or define a standard for the Internet community. The particular RFCs discussed in this section describe methods to allow certain non-native applications to utilize a TCP/IP transport network. These RFCs are as follows:

**RFC 1001, RFC 1002** NetBIOS applications

**RFC 1006** OSI applications

These RFCs allow such major applications as X.400 to run over TCP/IP networks. There is a clear separation between application services and transport layers in this technique. Hybrid stacks exist to support this function; for instance, for RFC 1006, there exists the upper part of the OSI stack and the lower part of the TCP/IP stack in the end nodes. Note that in order for this hybrid approach to be functional, some extra code is required to compensate for the differences of the dissimilar stacks. This extra code is defined by the RFCs for this situation.

In Figure 49 on page 74, Appl-A is an OSI-based application that needs to communicate with its partner across a TCP/IP network. The RFCs allow Appl-A to be coded as usual using the standard application programming interface and application services required for that application type; the same application coding takes place as if it were an application running on a native OSI network. RFC 1006 provides the compensations required to mask the differences between OSI and TCP/IP transport requirements, and the Appl-A packet can be sent across a TCP/IP network to a corresponding partner. Address mapping is provided by the RFCs, so Appl-A is unaware that a different transport is being used. The partner must also have RFC 1006 implemented to allow the compensations to be performed in reverse before handing the packet to the Appl-A protocol stack. Similarly, a NetBIOS application can be handled over TCP/IP using RFCs 1001 and 1002.

Products must be purchased that support the implementation of these RFCs, and this software will reside in end nodes. There should not be any impact to the TCP/IP network to support this functionality.

**Advantages of Mixed Protocol Standards -- RFCs**

- NetBIOS and OSI applications can now run over a TCP/IP network without requiring modification. These applications are unaware that they are running on a different transport network.
- A single transport protocol, TCP/IP, can be selected for the entire network. Using a single transport protocol reduces the complexity and cost of the network, since a single network management, configuration management, and change management approach can be selected.

**Disadvantages of Mixed Protocol Standards -- RFCs**

- For each application service and transport combination, a separate RFC identifying the necessary compensations is required. Only OSI over TCP/IP, and NetBIOS over TCP/IP compensations exist currently. If other mixed-stack situations are required, additional RFCs must be created. A new RFC will need to be defined for each mixed-protocol stack combination that is desired.

### 2.3.3 Multi-Protocol Transport Network (MPTN) Technique

Multi-Protocol Transport Networking (MPTN) is a relatively new technology that permits an application to utilize different transport protocols. It is a generalization and standardization of “hybrid stack” or “mixed-protocol stack” technology, with an approach very similar to the Mixed Protocol Standard RFCs just described. Whereas the RFCs are currently specific for particular mixed-protocol stacks, namely for OSI over TCP/IP or NetBIOS over TCP/IP, MPTN provides a methodology to make this approach general for any application service over any transport protocol. MPTN provides a set of documented, standardized *compensations* on top of the transport layer to permit the application services to be separated from the native transport, yet have all their application's protocol-specific requirements met. These compensations also include address mapping. Thus, a CPI-C-based application (normally requiring SNA) can now run over a TCP/IP network, or a sockets application (normally requiring TCP/IP) can now run over an SNA network.

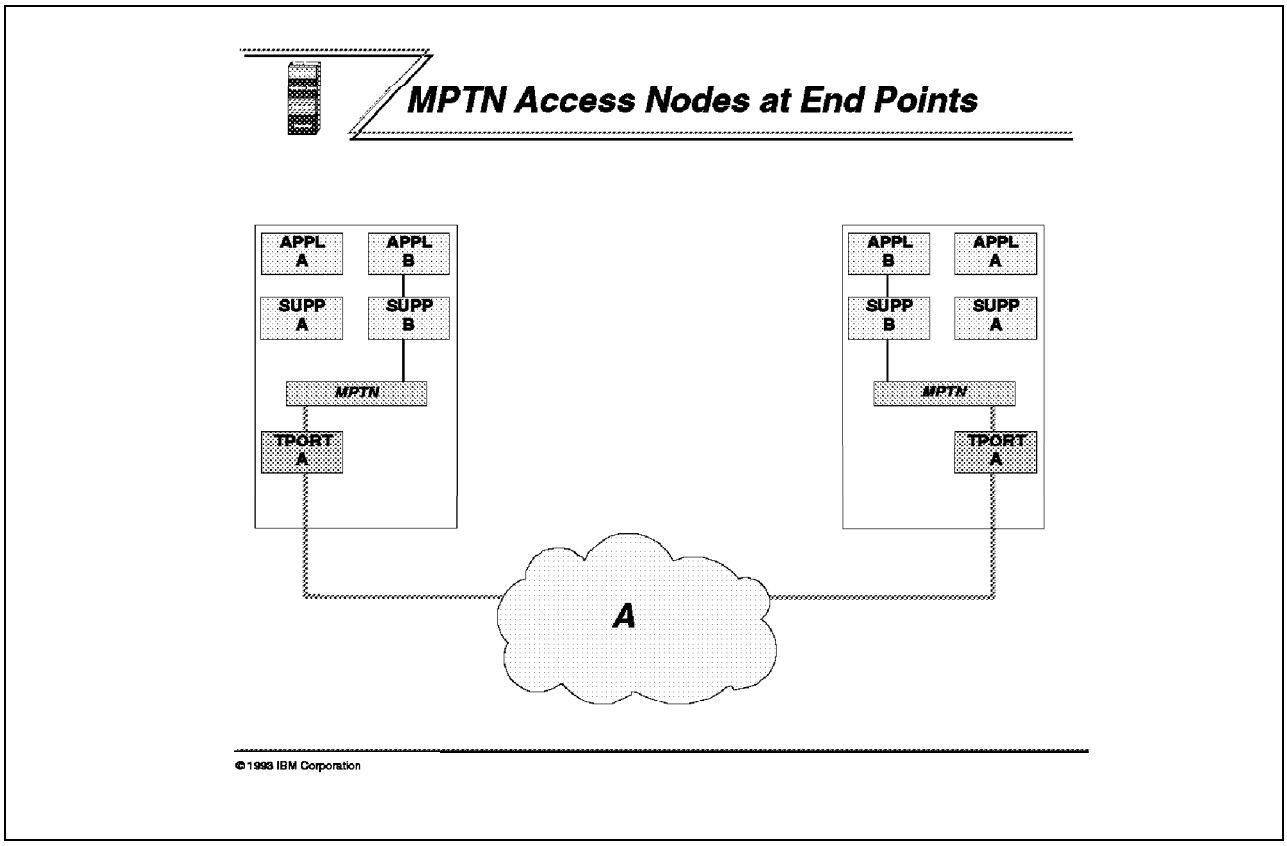



*Figure 50. MPTN Technique. MPTN Access Nodes at End Points*

The code required to perform this function must be available in a paired arrangement. As shown in Figure 50, Appl-B in the left node needs to send data to Appl-B in the right node over a transport network which only supports Tport-A. The data from Appl-B in the left node is intercepted by MPTN code in that node after passing through Appl-B's application services layer. MPTN provides compensations to mimic the native Tport-B, therefore satisfying session requirements needed by Supp-B. MPTN then provides compensations for Tport-A, so that this data can be sent over a normal Tport-A session. A *surrogate* session between the two MPTNs is set up, providing information on these compensations, so that appropriate actions can be taken by MPTN in the right node once the packet is received across the network. There is one

surrogate session per Appl-B to Appl-B session, therefore maintaining a unique end-to-end session identity and providing all necessary session management functions for Appl-B. From an Appl-B viewpoint, it is simply in session with its partner Appl-B over a native Tport-B network: MPTN is transparent to the application.

Although Appl-B's data is being encapsulated in a Tport-A packet, MPTN provides more of a *conversion* function than a traditional encapsulation technique; that is, Tport-A is actually being substituted for Tport-B, instead of the standard encapsulation procedure of Tport-A headers and trailers being wrapped around a Tport-B packet. There is a session-level conversion taking place, not an application-level conversion.

Figure 50 on page 76 illustrates the situation when the MPTN partners exist in end nodes. There are also other configurations possible. A **single gateway** is illustrated in Figure 51 on page 79. There is some important terminology mentioned in this diagram: any end node that contains MPTN code is referred to as an access node; any node that does not contain MPTN code, but just its native protocol stack, is called a native node. A gateway that contains the MPTN code is referred to as a **transport gateway**, to emphasis the fact that this function (shown as MPTN Relay) is taking place at OSI layer 4, the transport layer. No changes need to be made to the native node to permit communication over its native Tport-B network to the transport gateway; the transport gateway will provide all compensation functions.

Another configuration is illustrated in Figure 52 on page 80, where two transport gateways are present, and there is no MPTN code in the end nodes. Note that there is also an intermediate conversion from Tport-A to Tport-C for the intermediate network; MPTN permits this conversion to take place, and a "relay" of the application packets over one or more transport protocols to take place.

MPTN access node specifications have been submitted to X/Open** to be considered as a standard for mixed-protocol processing; X/Open has published an initial Guide to MPTN.

The impact of MPTN is mainly on networking system software developers, who must be able to split their current implementations of networking protocols between the application services (SUPP) and transport network (TPORT) levels to permit the insertion of the MPTN layer. Thus, networking protocol software may need to be changed in end nodes to provide MPTN access node functionality, or in gateway node to provide the transport gateway functionality. User applications should not be affected, unless these applications utilize functions that are not documented in the compensation standards.

Current examples of MPTN implementations are the IBM AnyNet* products, available for MVS/ESA and OS/2. These products currently provide SNA APPC support over TCP/IP, and TCP/IP sockets over SNA. Support has also been announced for NetBIOS applications over SNA networks. Other vendors have indicated their intention to utilize MPTN, such as Proginet for OSI over SNA, and KI Research for DECnet over SNA. Once X/Open adopts MPTN, it is anticipated that many more vendors will use this technology.

**Advantages of MPTN Access Nodes**

- MPTN permits applications to use non-native transport protocols, therefore enhancing connectivity.
- As a software-only solution, it could be lower cost than hardware-based implementations.
- Existing applications, written to a standard API and utilizing supported application services for which compensations exist, will be supported without modification.
- Compensations for transport protocol differences are provided by MPTN.
- A single transport protocol can be selected for a given network, therefore reducing the complexity and cost of multiple transport networks. A single network management, configuration management, and change management approach based upon this selected transport protocol can be chosen.

**Disadvantages of MPTN Access Nodes**

- At least two nodes in the network must have MPTN software.
- Commercial availability from multiple vendors is limited.

### 2.3.3.1 Single MPTN Gateway Technique

A single MPTN gateway can interconnect two unlike networks by performing a "relay function," as shown in Figure 51 .

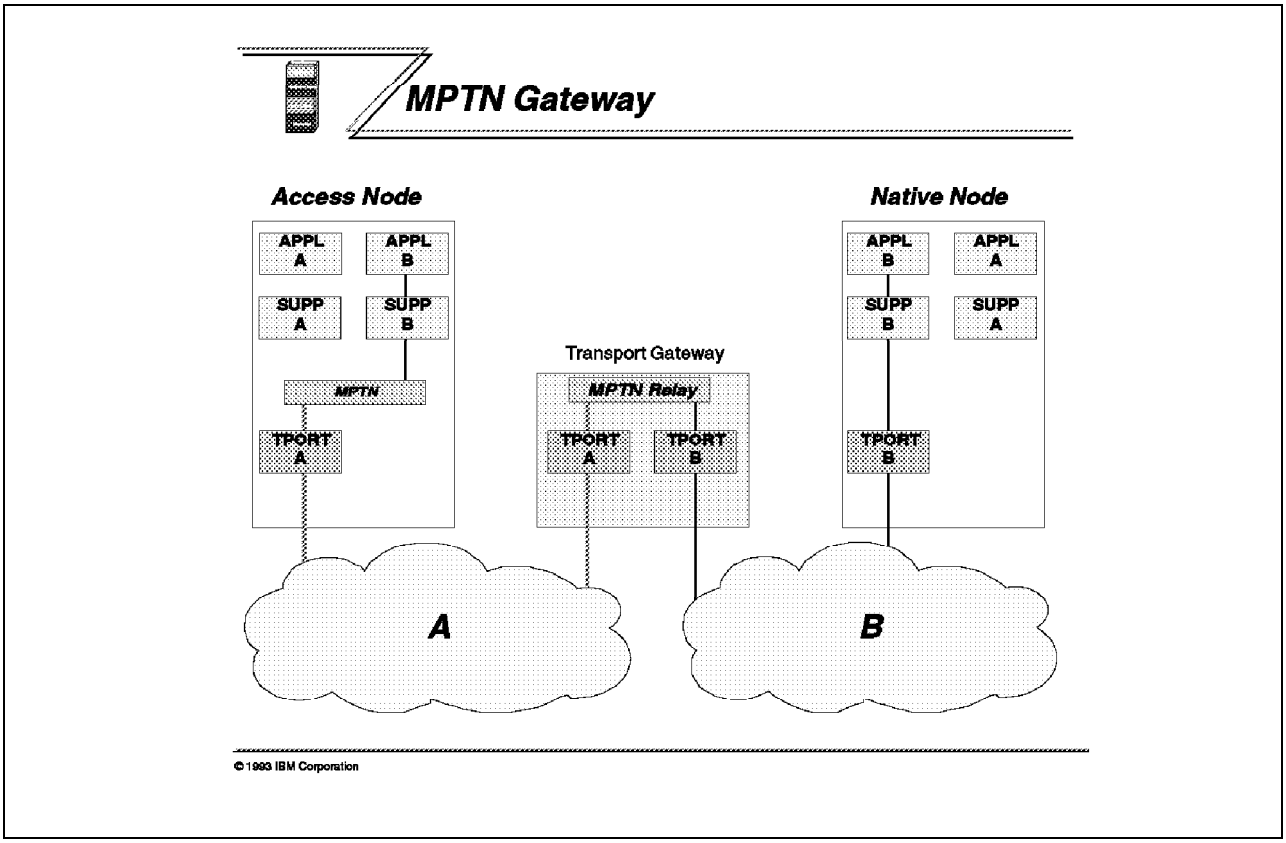



*Figure 51. Single MPTN Gateway Technique. Two Adjacent Networks (of unlike types)*

In the above diagram, all traffic from Application-B (upper left-hand corner, second column from left) flows first through a "foreign transport network" of type-A, then through an MPTN relay point, and finally through a "native transport network" of type-B to its partner Application-B (upper right-hand corner, second column from right). The application partners are connected indirectly through a combination of both a native and a non-native transport network.

### 2.3.3.2 Multiple MPTN Gateways Technique

Multiple MPTN gateways can interconnect two like networks by each performing a “relay function,” as shown in Figure 51 on page 79 .

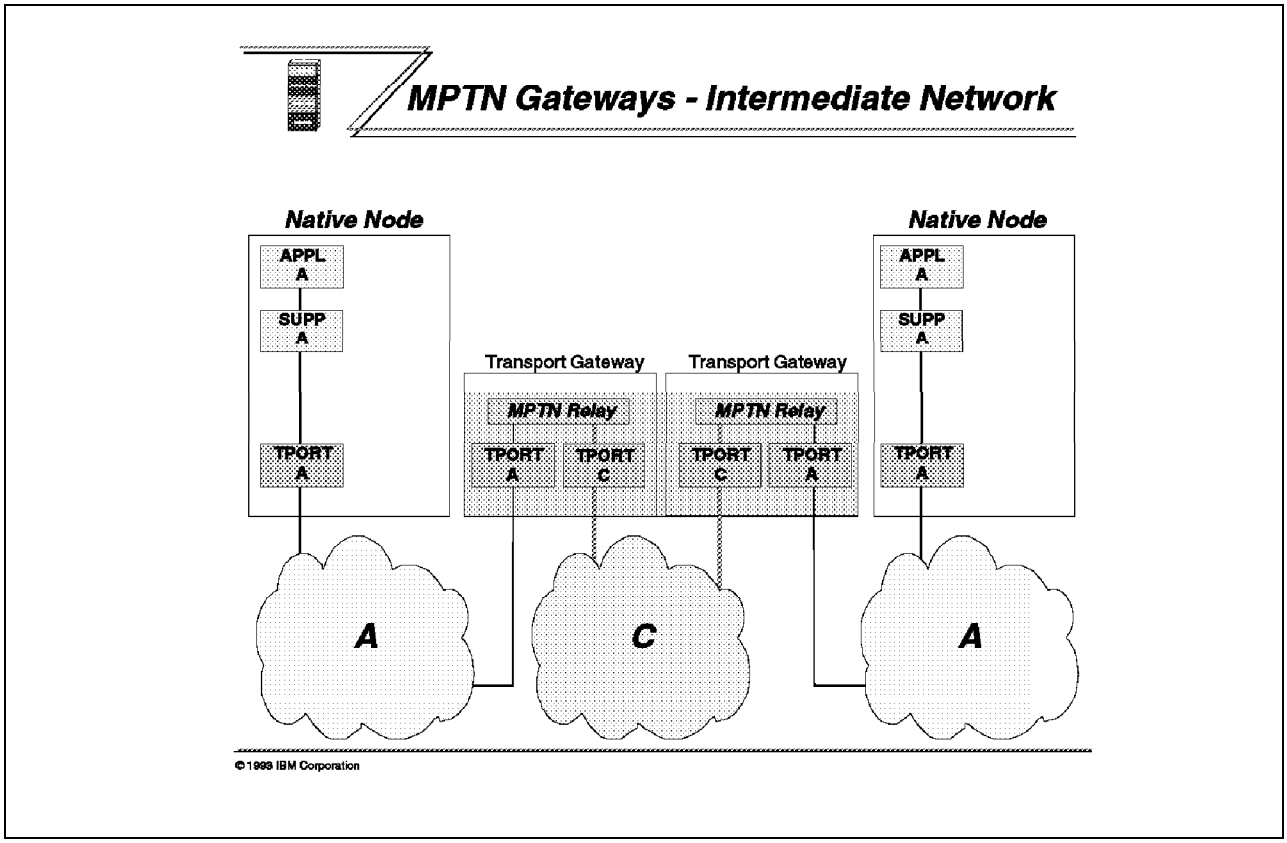



*Figure 52. Multiple MPTN Gateways Technique. Non-Adjacent Networks (of same type)*

In the above diagram, all traffic from Application-A (upper left-hand corner) flows first through a “native transport network” of type-A, then through an MPTN relay point into a “non-native transport network” of type-C, then through another MPTN relay point into a “native transport network” of type-A to its partner Application-A (upper right-hand corner). The application partners are connected indirectly through two native transport networks and one non-native transport network.

## 2.4 Application Support Technologies

Application support technologies are implemented outside of a network.

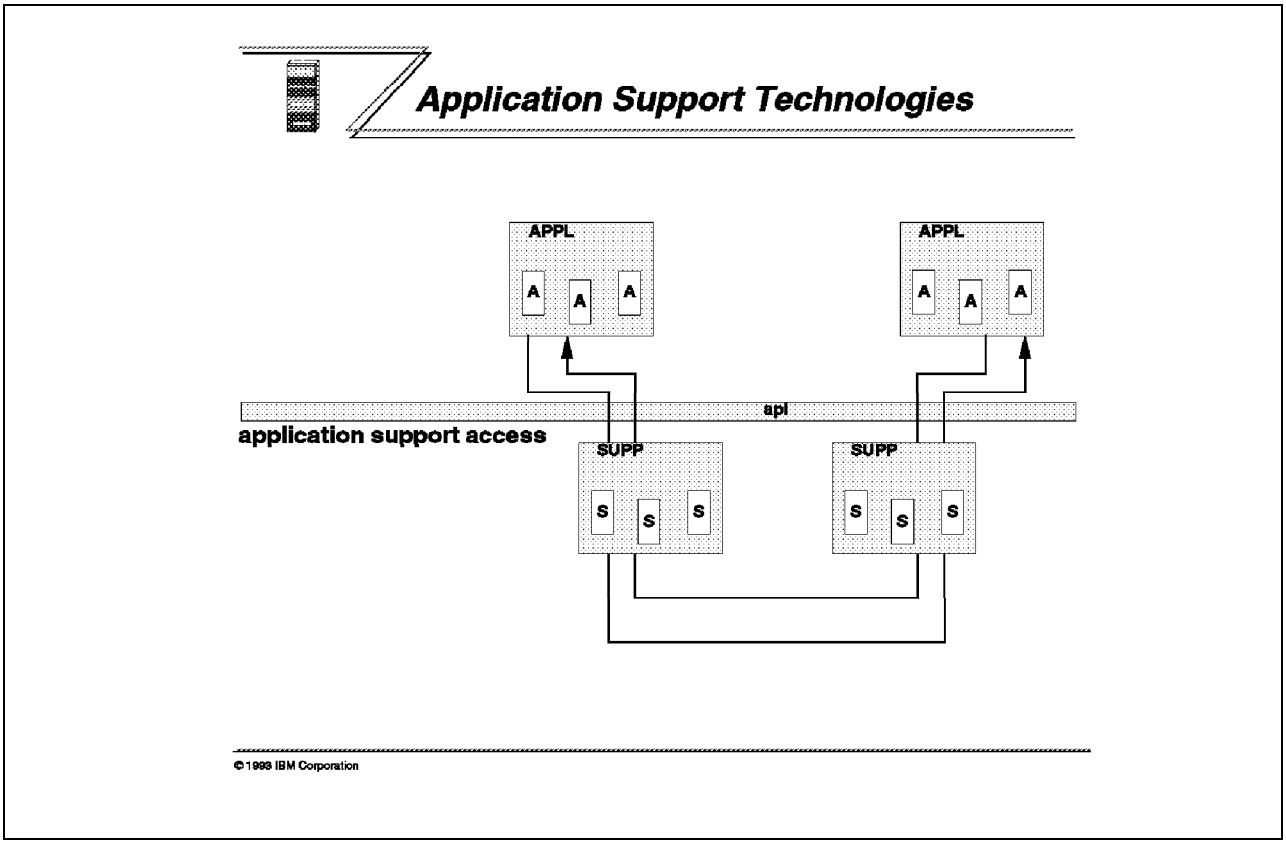



*Figure 53. Application Support (SUPP) Technologies*

Some of the technologies implemented within the application support layer (program group) are:

- Middleware
- XTI (Transport Interface Selection)
- Remote API
- Application Gateway
- Multi-Protocol Server/Gateway

## 2.4.1 Middleware Technique

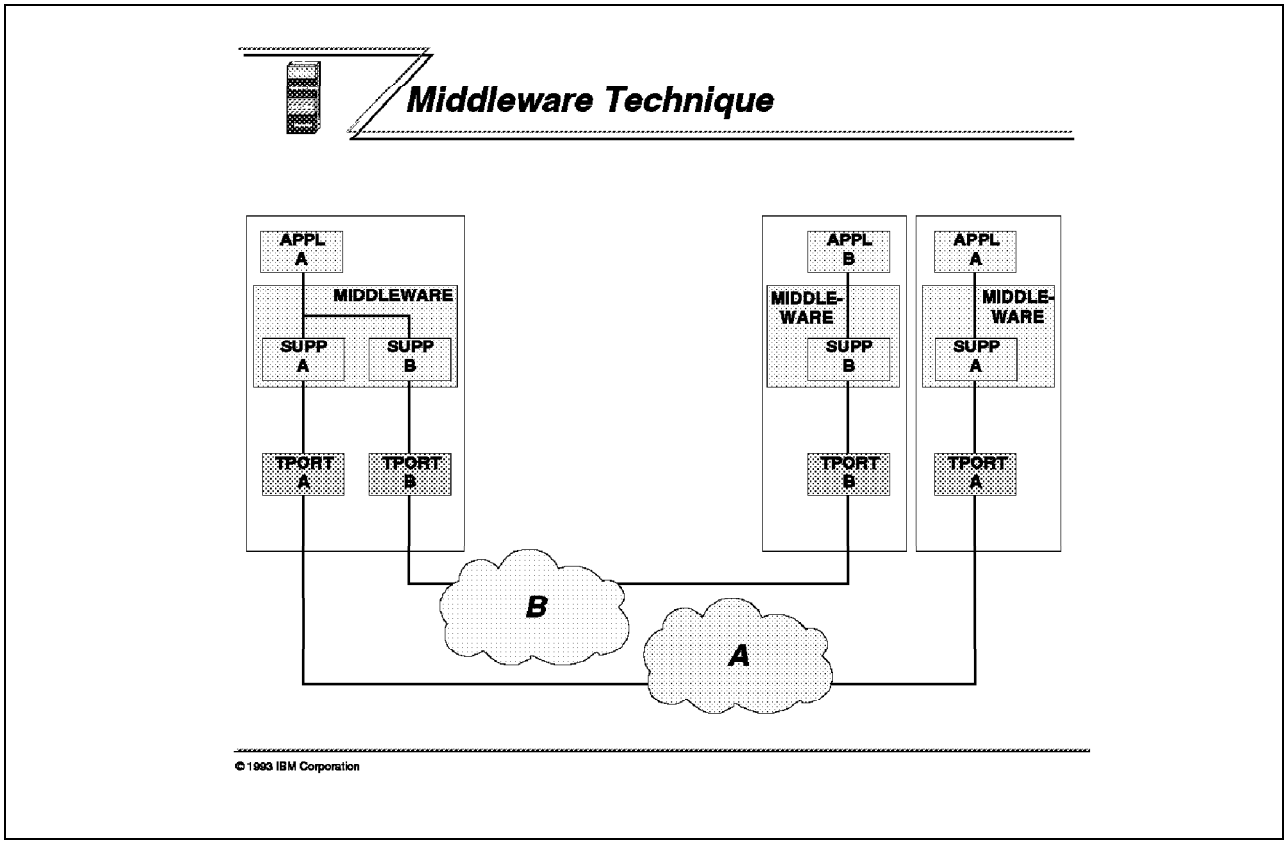



*Figure 54. The Middleware Technique*

Historically, **middleware** is any software that "sits in the middle between an application and something more complex" — such as, between an application program and an operating system function or between an application and a file storage device or (especially for program-to-program communications) between an application program and a communications network.

Communications *middleware* is a relatively new term, which describes a particular approach to provide software in the "middle" between the application and the transport protocols. Most affected are the application programming interface and application support layers, OSI layers 5-7. Middleware provides support across multiple transport protocols, thus providing application-to-application connectivity for the applications that use middleware.

Communications middleware products provide a variety of programming, data management, and networking services.

Middleware products typically have two distinctive characteristics:

- They usually provide their own unique, often simplified, API.
- They provide support for their applications across more than one transport protocol.

Due to the multiple transport protocols support, the claim is often made that middleware is *protocol-independent*. Actually, this is product-wise true only for a limited number of transport protocols that the middleware product has either implemented within its own product or accessed through interfaces to full-stack

implementations. In order to provide for this multi-protocol support, the middleware product usually manages its own name space, providing for mapping to the appropriate transport addresses. The details of the transport protocols are often hidden from the application programmer. The transport protocol is chosen by the middleware platform dependent upon the transport requirement of the remote system. The remote system must have a corresponding application, developed using the same middleware product.

In Figure 54 on page 82, Appl-A in the left node can utilize Tport-A and Tport-B to communicate with the end nodes on the right; however, the node on the right will always use Tport-A, and may not be easily changed to Tport-B. Thus, separate transport networks may still be needed to support the various transport protocol requirements.

Implementations of this middleware approach vary widely in terms of functionality and standard interfaces. Some examples include Arthur Andersen's Foundation, IBM's DAE, and IBM's CICS*.

The impact to the end nodes, as well as to the applications, is quite significant. Each node must have a copy of the middleware product, and existing applications must be recoded to utilize the middleware product's API. There is usually minimal impact to existing networks, assuming that these networks implement the same transport protocol support as the middleware products.

**Advantages of Middleware Technique**

- Application programming may be easier with the middleware's simplified API than an equivalent industry-standard API. Toolkits are often available to assist application developers.
- Application portability usually exists across multiple hardware and operating system platforms supported by the middleware product.
- Communication over multiple transport protocols is supported, allowing for application interoperability for applications developed with the middleware product.

**Disadvantages of Middleware Technique**

- All nodes must implement the same middleware product in the appropriate hardware or operating system platform version.
- Applications are limited to the transport protocols that the middleware product either provides within the product or accesses through interfaces to full protocol stacks.
- Existing applications will need to be redeveloped on the middleware product, which may be quite expensive.
- Multiple transport networks exist, resulting in increased costs in terms of network, configuration, and change management.

## 2.4.2 X/Open Transport Interface (XTI) Technique

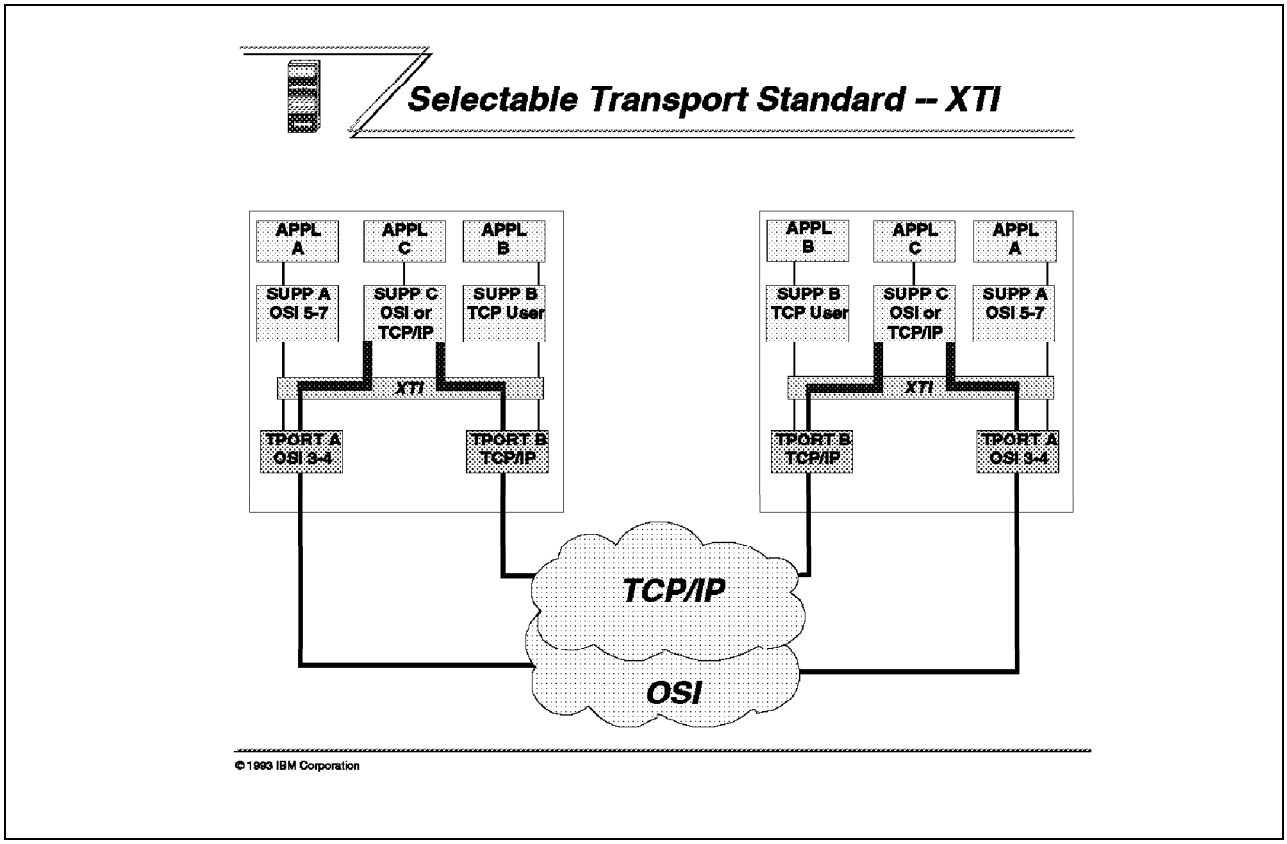



*Figure 55. X/Open Transport Interface (XTI) Technique. A Selectable Transport Standard*

The XTI (X-Open Transport Interface) is the *network access* mechanism in the above diagram. The 'SUPP-C' layer, based upon application-provided parameters, selects the transport network to be used.

The X/Open Transport Interface (XTI) provides the ability to code an application to run over TCP/IP, OSI, or NetBIOS networks. There is also a forthcoming specification for an application to run over an SNA network. XTI actually provides an application programming interface directly to these transport layers. However, it does not provide transparent access to these transport layers; the application support programs must be aware of the protocol to be selected.

Figure 55 illustrates how products that implement this interface operate. If Appl-A is a standard OSI application, it will proceed through its protocol stack as normal to communicate with its partner application over an OSI network. Similarly for Appl-B; the presence of the XTI layer on top of the transport layers is ignored. However, if Appl-C wishes to be able to select which transport protocol to utilize, it must be coded as a common subset of OSI and TCP/IP transport services. If only this subset is utilized, then either the OSI or TCP/IP transport can be selected. This selection could be made either at compilation time or runtime, depending upon the implementation. The application must also be aware of the addressing format of the partner, which is transport protocol-specific.

An application that uses this XTI interface can communicate to a partner application which is coded natively, assuming that only the common subset of application services is utilized.

In order to implement this approach in end nodes, a product must be found which supports this XTI binding to the transport protocol. These are available for OSI, for TCP and UDP over IP, and for NetBIOS. Applications may need to be recoded to support the XTI. XTI code must exist in the end nodes. There should be no impact to the native transport networks.

**Advantages of Selectable Transport Standard -- XTI**

- New applications can be developed using XTI, permitting interoperability over NetBIOS, OSI and TCP/IP networks.

**Disadvantages of Selectable Transport Standard -- XTI**

- Few transport protocols are currently supported.
- XTI is currently not widely implemented.
- Recoding existing applications might be cost prohibitive.
- Restricting application development to the least common denominator of functions of the transport protocols can severely hinder a major development effort. It may prohibit the recoding of older applications if major functions are missing.
- XTI does not specify a directory service, so application developers must choose their own. This directory service must be compatible with the combination of application and transport protocols chosen.
- Multiple transport networks exist, resulting in increased costs in terms of network management, configuration management, and change management.

**Note:** The reason XTI is not included in the same discussion with MPTN and the Internet RFCs is that it is more like middleware in that it does not provide *transparent access* to the underlying transport(s).

## 2.4.3 Remote API Technique

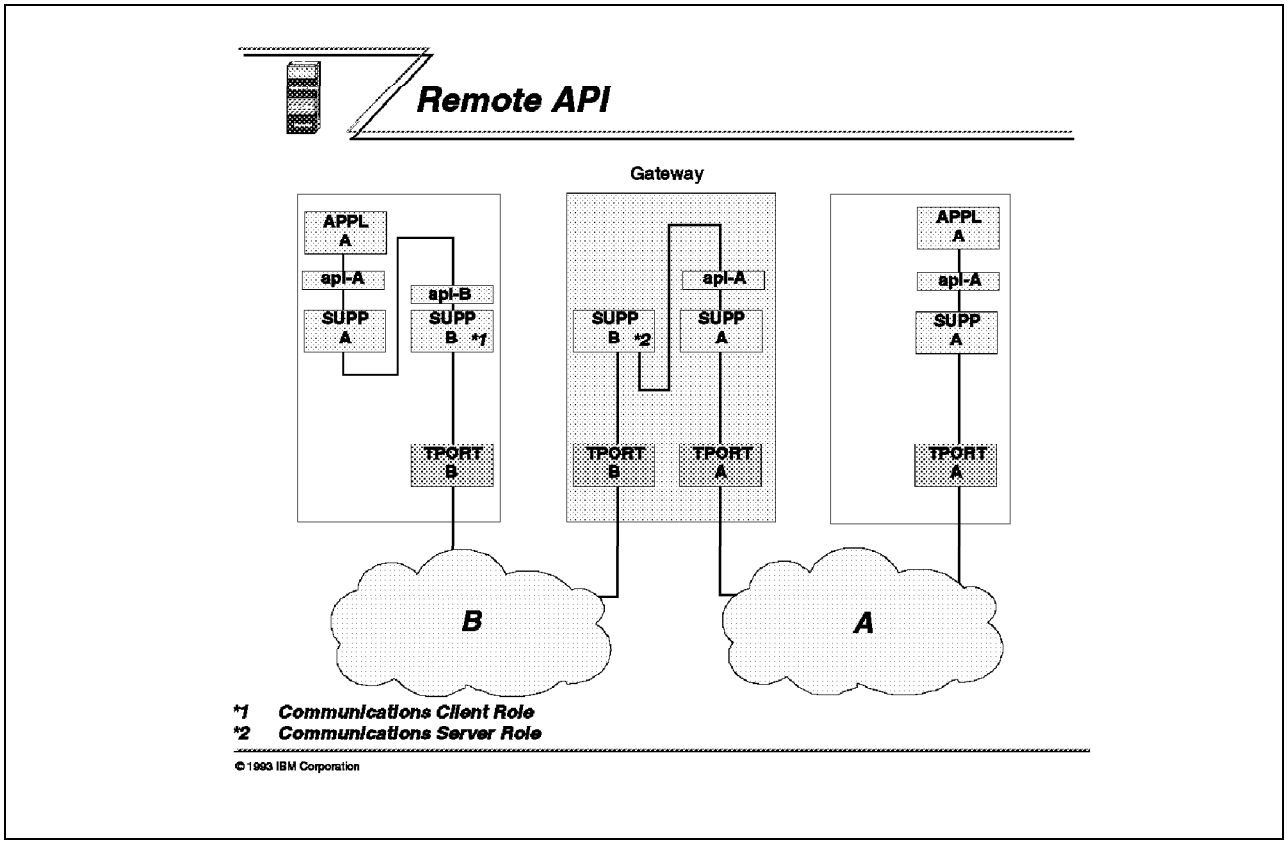



*Figure 56. Remote API Technique*

Remote APIs allow the use of a different API over a non-native network. A **communications server** is required to provide the necessary missing protocol support. As shown in Figure 56, Appl-A in the left node wishes to communicate with its partner Appl-A in the right node, but must cross Network-B first. The left node does not have a full protocol stack for Appl-A, but just the necessary API. A special client stub is provided, which will take the information from the calls to api-A, and envelop it within protocol B to reach the communications server. The communications server will extract the information needed to call the api-A which communicates over Network A to the correct node. Appl-A in the left node believes it is communicating directly with its partner, and is unaware of the extra processing taking place over the first network.

For systems which are storage-constrained, particularly PCs on LANs needing to communicate with a remote system, this “skinny client” approach is quite popular. In fact, it is sometimes referred to as a “LAN-based middleware” or “client-server” approach. In this diagram, Network B would be the LAN, needing to access Appl-A in the right node over wide area network A. Many products implement this approach. For instance, the SNAps gateway from Apple** Computer permits LU6.2 applications on the end nodes to communicate with the communication server using AppleTalk**, and only the communication server has the full SNA protocol stack to interact with the host over the SNA network. The IBM OSI/CS products provide another example, where a remote OSI API can be used over an SNA LU6.2 session between the client and server.

A communication server must be purchased and possibly an application coded to provide this functionality. The communication server must provide dedicated support for each desired application programming interface, and there may be limitations imposed due to the number of supported protocols.

**Advantages of Remote API Technique**

- Remote APIs eliminate the need for an end node to have to implement multiple protocol stacks.

**Disadvantages of Remote API Technique**

- They are dependent on the communication server, which can be a single point of failure.
- Due to the dependency on the intermediate server, there is no end-to-end session visibility, and hence session management and error recovery may be impacted.
- The same API cannot be easily or efficiently used to communicate with a partner on the same network (program 'B' in the example).

## 2.4.4 Application Gateway Technique

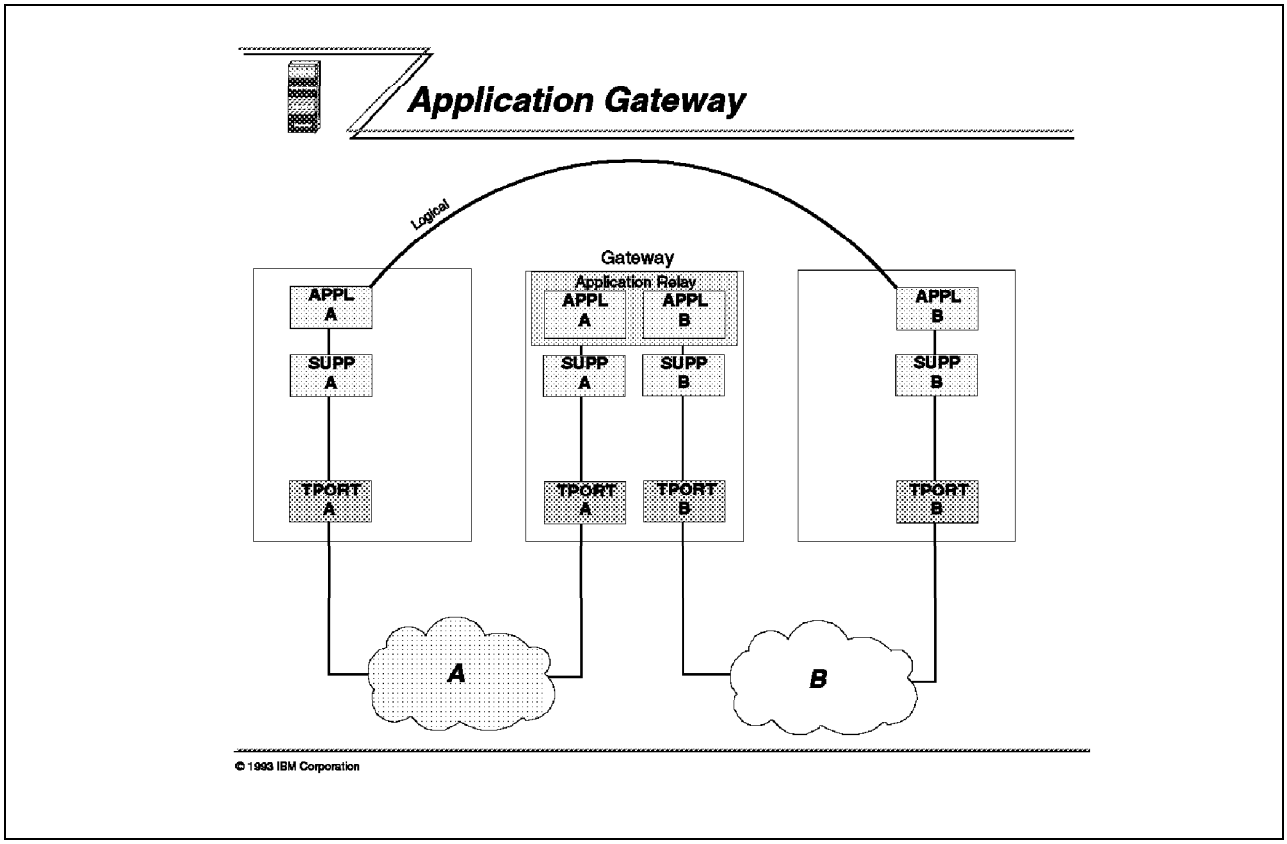



*Figure 57. Application Gateway Technique*

Application gateways are commonly used to allow interoperability between two functionally similar applications that may have quite differing requirements in terms of transport protocols, application services, application programming interfaces, or data formats. A very common type of application gateway exists to convert messages from dissimilar electronic mail applications. For instance, there are application gateways to convert messages between DEC All-in-One and IBM Officevision/VM*. Application gateways are very protocol- and/or product-dependent. A different application gateway might be required if messages need to be exchanged between cc:Mail** and IBM Officevision/VM.

The functionality of an application gateway is illustrated in Figure 57. Appl-A in the left node needs to send information to Appl-B in the right node. Everything about Appl-B is quite different from Appl-A: transport protocol, application services, application programming interface, and data format.

Appl-A in the left node will address its packet for Appl-B, and send this packet over Network A. The application gateway in the middle node, which is connected to both Network A and Network B, realizes that this packet is destined for Appl-B in the remote node, and will receive the packet. The packet will traverse the Appl-A protocol stack in the gateway, and then be converted by the gateway relay code into the appropriate format for Appl-B. The new packet, with Appl-B's address indicated, will proceed through the Appl-B stack in the gateway, cross Network B, and be received and processed by Appl-B in the remote node.

In theory, both Appl-A and Appl-B believe that they are conversing with native partners, and neither should be aware of the conversion. In reality, the gateway may not be able to compensate for all difference between the applications, or between the transport layers. Thus, users may have to restrict themselves to a subset of functionality that the application gateway can convert.

Application gateways are often implemented in different nodes than the end nodes; in fact, the application gateway node may serve to physically connect the two different networks as well as convert the application data. Some implementations may have additional functionality in terms of security access and the filtering of traffic between the networks. The application code on the end nodes is not impacted.

**Advantages of Application Gateway Technique**

- Application gateways allow for the interoperability of disparate applications.
- For applications that were never built on a standard protocol stack, such as many electronic mail packages, this approach may be the only choice to provide interoperability.

**Disadvantages of Application Gateway Technique**

- Application gateways are protocol- and/or product-specific. A combination of gateways may be required to achieve the desired result.
- No consolidation of transport protocols is provided.
- Functionality of the native applications may be limited depending upon how fully the gateway can convert between the applications.
- All traffic must go to the gateway for conversion. Thus, unless the gateway is duplicated, it becomes a single point-of-failure.
- Application gateways are expensive in terms of performance overhead, and in terms of requiring dedicated hardware and software.

### 2.4.5 Multi-Protocol Server Technique

A piece of software can play the role of server to many clients, where the clients are connected to the server across networks of many types. Such a server is often designated a **multi-protocol server**, since it serves its clients across multiple networking protocols.

In some cases the multi-protocol server plays only the role of server; in other cases the multi-protocol server plays not only the role of server but also the role of gateway.

### 2.4.5.1 Multi-Protocol Server (Server-Only) Technique

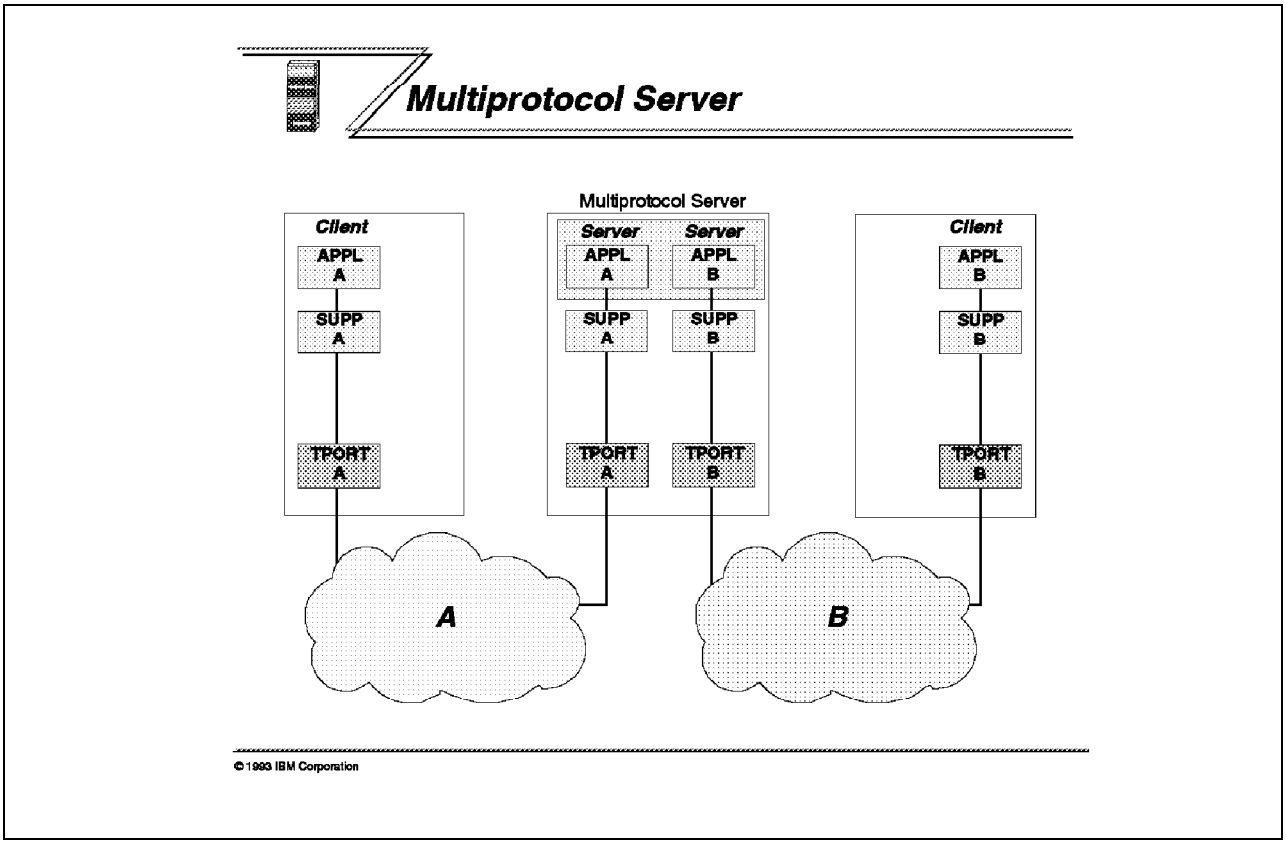



*Figure 58. Multi-Protocol Server Technique, Server-Only*

In the above diagram, the application in the middle is playing the role of server, communicating (with application support program help) through two different network types.

With multi-protocol servers, the server will communicate with the native application and provide a service on behalf of this application. The server often supports applications which run over different transport protocols. This is very common with LAN operating systems, such as Novell Netware** or Banyan Vines**, which permit a variety of requestors, running over different transport protocols, to access the server and access information. Note that in Figure 58, Appl-A communicates with its partner Appl-A, which is part of the server code. Likewise, Appl-B communicates with the server. However, Appl-A does not communicate directly with Appl-B; they may share results maintained by the server (for example, databases), and thus communicate indirectly.

A more complex situation is illustrated in Figure 59 on page 93, where a single application provides both server and gateway functionality. Appl-A and Appl-B can make requests of the server; if the server does not have the desired information locally, it will invoke a client Appl-C within itself to obtain the information from Appl-C. Once it has the information, it will reformat the data in a form appropriate for the original requesting application and return the data. This is much like a “subcontract operation” or “farmed-out operation,” and is often so labeled. Note that the original application is unaware that a remote application has been invoked on its behalf. The remote server responds back to the original server with the desired data, and the original server reponds back to

the appropriate client. An example of this combination is the EDA/SQL** products from Information Builders, Inc. The EDA/SQL product provides SQL data access to a range of databases on interconnected networks regardless of the file structures, hardware environments, or transport protocols. To achieve this interoperability, the products employ multiple protocol stacks and applications to translate between the different file structures and different SQL-call syntax.

There are no changes to the original applications (Appl-A and Appl-B) in the end nodes. A separate server is used, and this node will have multiple protocol stacks and applications to provide this functionality. This server function will be provided for specific protocols and/or products.

**Advantages of Multi-Protocol Server**

- Only one protocol stack is required in the end stations.
- Users are usually aware of the server, but need not be aware of the other applications that the server might invoke. To the user, he receives data in the usual format, even though this data may come from a different application. Thus, neither the user nor the originating application needs to be concerned with addressing the remote application.

**Disadvantages of Multi-Protocol Server**

- Multi-protocol servers are limited by the protocols and/or products that it can support.
- The server is a single point-of-failure.
- No reduction in network protocol is achieved.
- Skills need to be developed to manage the server software and hardware.
- Network management is more complex, since it is difficult to see beyond the server to the end stations.

### 2.4.5.2 Multi-Protocol Server (Server and Gateway) Technique

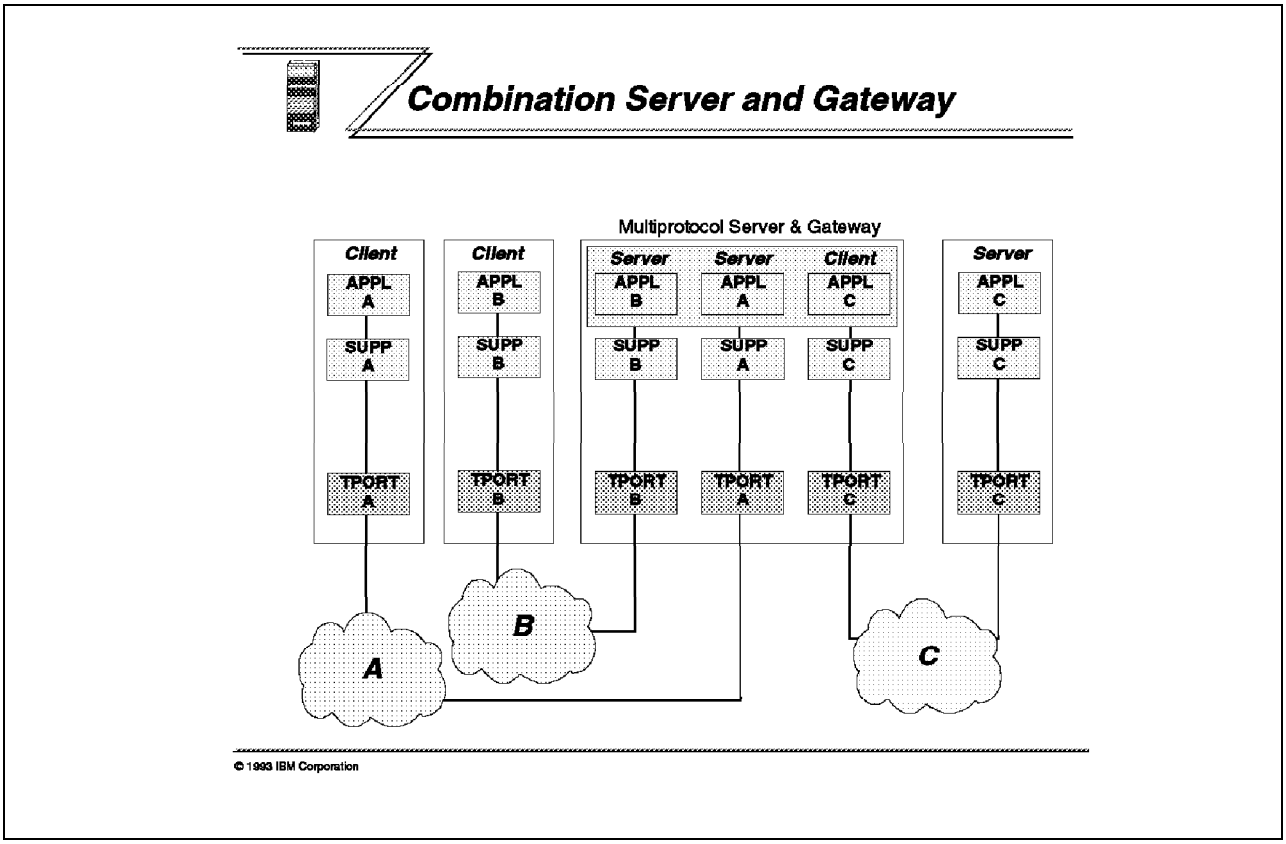



*Figure 59. Multi-Protocol Server Technique, Server and Gateway*

In the above diagram, the server/gateway application in the middle is playing both the role of server (to programs A and B) and the role of client (to program C). There are three client-server pairs: A-(MP Server/GW), B-(MP Server/GW), and (MP Server/GW)-C. You can notice a “Y-shaped” flow with Appl-A and Appl-B at the top of the 'Y', Appl-C at the bottom of the 'Y', and the multi-protocol server in the middle of the 'Y'.

## 2.5 Summary

As discussed in this chapter, there are many considerations that must be kept in mind as one evaluates these technologies to provide alternative solutions to complex connectivity and interoperability problems. Each particular technology must be selected with regard to a specific customer situation In Chapter 3, “Typical Customer Situations” on page 95, the applicability of these technologies within a particular situation is illustrated using a set of typical customer situations.

# Chapter 3. Typical Customer Situations

The objective of this chapter is to evaluate, through typical customer situations, the use of various existing and emerging technologies.

## 3.1 Introduction

Now that we have examined these open networking technologies, which of these technologies is the best choice to solve a particular interoperability problem? Unfortunately, there is rarely a clear answer.

There is an old joke among networking professionals that there are always three additional layers or levels of complexity beyond whatever is shown in a protocol stack (for OSI, SNA, TCP/IP, etc.). These three additional layers of complexity are financial, political, and religious. Quite frankly, most networking decisions are made on the basis of these three factors, and **not** on the basis of the "best" technology.

Financial decisions are made on the basis of total cost, and the contribution towards the future profitability of the company. "Total cost" should be examined carefully; it is more than the price quote given by a vendor for hardware and software products, but includes the personnel costs of the people involved in administering and managing the new network configuration. These personnel costs are often hidden, and are difficult to estimate. Also hidden are the costs to train end users on using this new equipment, and impact of performance, and availability/reliability differences on personnel productivity. Can the solution be expanded in a scalable manner as the business grows and more people are added to the network? And finally, does the solution provide a competitive edge to the company, either in terms of efficiency or by providing a new service to its customers?

Due to its very nature of interconnecting people and/or organizations, networking implementations have political implications: Who are the key decision makers? Who will run this new network implementation? Who has control?

Finally, there is the "religious" attribute of the decision making process. No matter how many facts are shown to certain individuals to argue for or against a particular technology, there are people who are simply set in their opinion, and nothing can dissuade them.

Given these constraints, how should one proceed? We can examine a set of general scenarios, and describe an approach to at least determining the applicable technologies for a given situation. We can identify some general questions to assess when examining these technologies for a given situation. However, it is impossible to determine a single solution without determining which products in the marketplace are available for the desired platforms, and also judging the features and functions that individual vendors provide beyond the core technology.

Something else to consider is that these techniques as described are NOT specific to any particular protocols or products. In particular, it should be noted that products may implement one or several of these approaches; and, it may very well be necessary (in fact, it is almost a certainty) to use several products

© Copyright IBM Corp. 1994

to obtain the desired solution. So, this exercise is only a beginning; it should be used to prepare to evaluate products.

## 3.2 Defining Four Customer Situations

We will now examine of set of common customer situations, and identify the applicable technologies to solve the connectivity or interoperability problems posed by each situation. Many complex customer scenarios are often composed of a mixture of the four fundamental situations listed below. These scenarios were developed and defined based upon the authors' experiences with customer situations; other publications may define and approach these situations differently.

- Situation 1. Adding A New Application
- Situation 2. Application Interoperability
- Situation 3. Network Interconnection
- Situation 4. Network Consolidation

The first two situations focus on *application* requirements, and must be evaluated from the application's viewpoint, from the application layer down the protocol stack (a “top-down” approach); whereas the last two situations focus on the lower networking layers, and will be evaluated using a “bottom-up” approach. Figure 60 illustrates the *range of consideration* when one is assessing these situations.

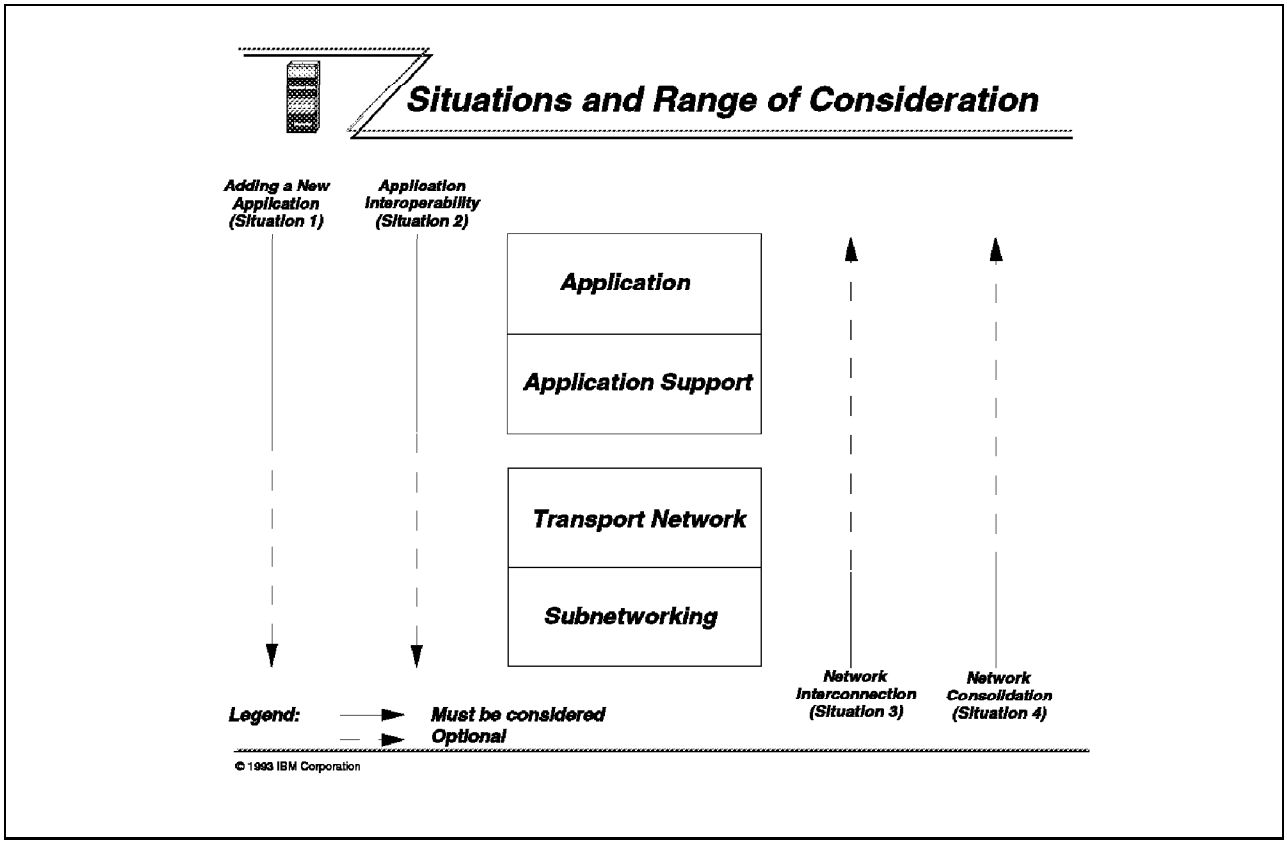



*Figure 60. Situations and Range of Consideration*

In the discussion of each major situation, some key decision-making criteria are identified. A few considerations for the applicable technologies are listed for each situation. No attempt is made to define the optimum solution; there are too many dependencies, as noted above, to allow us to be able to effectively compare and contrast these technologies, to the point of identifying the very "best."

### 3.2.1 Situation 1 - Adding A New Application

For this general situation, we are assuming that a customer has a *single physical network* (either WAN or LAN), running a *single transport protocol* currently (Tport-A), and this network is controlled by one organization. A new application must be added to this single physical network, which utilizes a different transport protocol (Tport-B). The customer may be adding a very new type of application which never existed before in the network (for example, adding transaction processing for the first time). There is no requirement for the new application to interoperate with existing applications.

Suppose the customer is either unwilling or unable to add the new protocol stack to the network. In this situation, only the original transport protocol, Tport-A, is allowed to run on the existing network. If a separate parallel network cannot be built for Tport-B, then Situation 1A (as depicted in Figure 61) has occurred. If a separate physical network can be built for Tport-B, or a subnetwork can be shared by both Tport-A and Tport-B, then Situation 1B (as depicted in Figure 63 on page 101) exists.

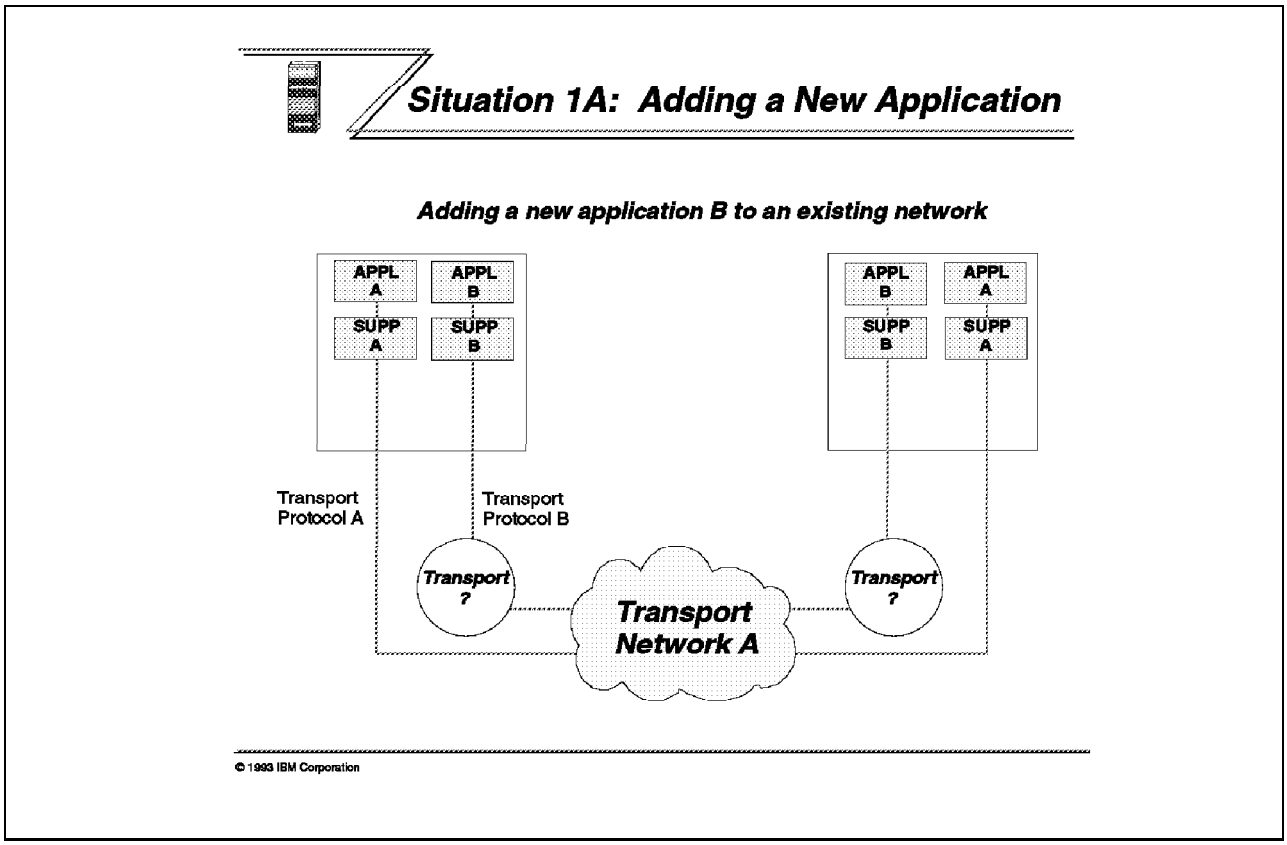



*Figure 61. Situation 1A: Adding a New Application B to an Existing Network*

Figure 61 illustrates a situation where the current network utilizes Transport Protocol A. Appl-B is the new application, and requires Transport Protocol B to communicate with its partner application over the network. The network is probably a wide area network due to the assumption that the new transport protocol cannot be accommodated on the same subnetwork (LANs do allow the sharing of the subnetwork). This assumption also means that the extended subnetworking approaches are not be currently used (such as Frame Relay or X.25), or else the wide area network would be multi-protocol-capable. If a separate wide area network is not possible, then the technologies illustrated in

Figure 62 on page 99 can be used. All of these technologies permit the utilization of a mixed-protocol stack approach, or some form of encapsulation.

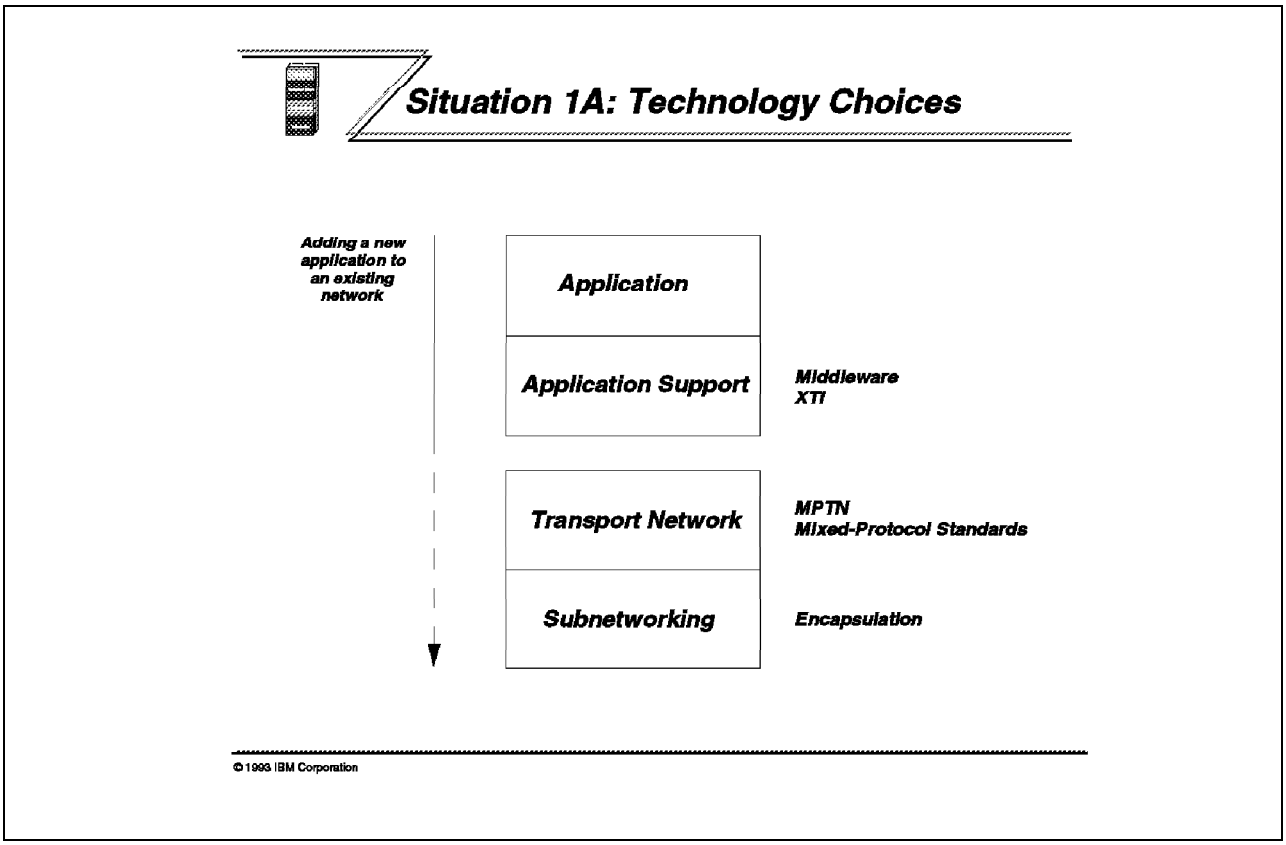



*Figure 62. Situation 1A: Technology Choices*

If this new application will be developed by the customer, it may be possible to select an Application Support-level solution, such as middleware or XTI. These solutions permit flexibility in choosing the desired transport protocol. For instance, if the current network is TCP/IP-based, and X.400 is the desired new application, it may be possible to code the new application to use XTI, allowing the transport to be TCP/IP instead of OSI.

If an off-the-shelf application is being purchased, it is more probable that a solution operating at the transport or subnetworking levels will be selected. These technologies are transparent to the application itself.

Most of these solutions function at the transport level, with the exception of middleware, which operates at the application services and application programming interface level. Middleware may be an appropriate solution when new applications are being developed by the customer; it is usually not a solution for off-the-shelf applications. Application programmers are usually aware of restrictions imposed by middleware and XTI; the other approaches are usually transparent from an application programming viewpoint.

The selection process will be limited by what actually exists in the marketplace for the particular protocols and operating system platforms involved. Considerations to be kept in mind as these technologies are evaluated include the following:

**Middleware**

- Does this middleware implementation exist on all required operating systems in the end nodes?
- How much flexibility does the middleware application programming interface give to the application developers?
- What are the extra facilities provided by the middleware implementation in terms of directory, security, and recovery?

**XTI**

- Is the new transport protocol a currently available XTI protocol, such as TCP/IP, NetBIOS, or OSI?
- If this is a new customer-developed application, can communications software and compilers be found that utilize this interface for the currently installed hardware and operating system platforms?
- If flexibility is desired in this customer developed application, can the application be coded with just the necessary subset of the transport functions?
- If this is an off-the-shelf purchased application, has it been coded to utilize the XTI interface? If so, does it support the transport protocol you want?

**MPTN**

- Do MPTN implementations exist for the necessary protocols?
- If MPTN implementations exist for these protocols, do they exist on the required operating system platforms in the end nodes? If not, do they exist for operating system platforms which could be used as a gateway?

**Mixed-Protocol Standards**

- Is it a supported application/transport combination currently supported by the RFCs or another standard, such as a NetBIOS or OSI application, which needs to run over TCP/IP?
- Do the protocol stacks installed in the end nodes support the standards?

**Encapsulation**

- Will this encapsulating software be placed in the end nodes? Or will it be placed in a separate node, requiring additional hardware?
- How is the encapsulation actually performed? Are Data Link Control protocols properly terminated and emulated? Are the transport-level protocols properly emulated? Is filtering of unnecessary packets performed?
- What are the performance characteristics of this encapsulation software? Is this acceptable?
- How are the different network addresses handled? What directory services exist?
- How expandable is this approach if the network grows and more nodes are added?

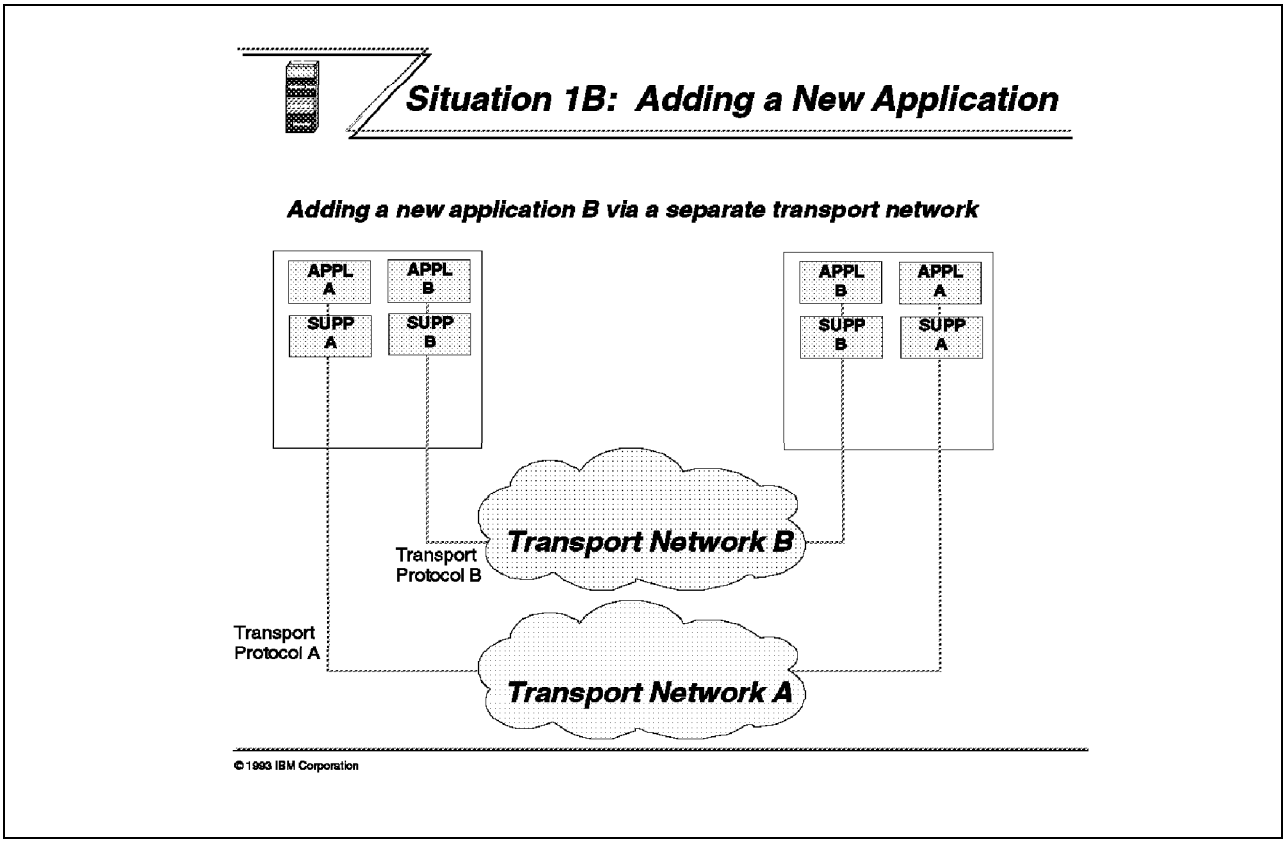



*Figure 63. Situation 1B: Adding a New Application B via a Separate Transport Network*

Three possibilities exist in this case:

1. The network is already multi-protocol and is able to accommodate the new transport protocol. For instance, if the network is a LAN, then additional protocols can use the same shared subnetwork.
2. The customer is willing to modify the network to make it multi-protocol; for instance, by adding routers.
3. The customer is willing to procure a separate wide area network to handle just the new transport protocol. This may be done for security or performance reasons.

All three of these possibilities are represented in the conceptual diagram in Figure 63. Two separate transport networks exist, whether or not they share the same physical media.

If a new transport protocol is to be added, it requires that a new protocol stack be installed in all end nodes which desire to use this new application. Additional equipment may be required, depending upon which of the choices is selected from the options listed in Figure 64 on page 102. Since the network can handle the new protocol, there is no need for mixed-protocol stacks or encapsulation approaches. Thus, the solutions are concentrated at the transport and subnetworking levels.

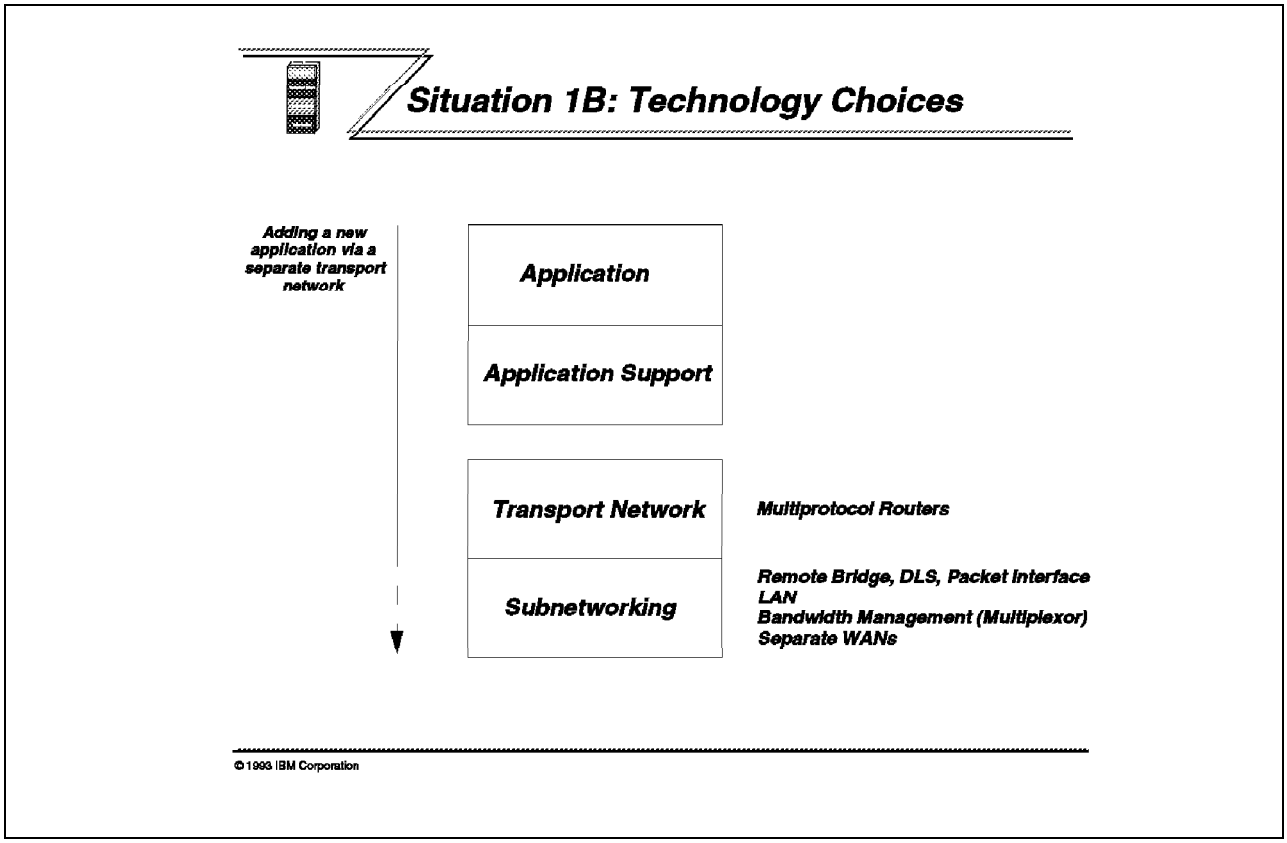



*Figure 64. Situation 1B: Technology Choices*

Obviously, LANs are a solution only if the communicating nodes are local in the same facility or campus. The other solutions are oriented mainly towards wide area networks. These wide area networks might be public or private, depending upon the availability of services.

Considerations to be kept in mind as these technologies are evaluated include the following:

**Multi-Protocol Routers**

- Are all needed protocols handled by the router?
- Are there any Data Link Control level concerns? Does the router properly handle these protocols?
- What happens in congestion situations? Are all protocols equal? Can the traffic be prioritized?
- How much filtering is available for each protocol?

**Extended Subnetworking**

- Is a public service (Frame Relay, X.25, SMDS) preferable to a private network? What are the costs involved, and how much control over the network is desired?
- Is this critical traffic? Does a prioritization scheme exist in this extended subnetworking implementation to handle higher priority traffic?
- How do the end nodes interface to the extended subnetwork? What is the cost of upgrading equipment or adding additional equipment?

**Shared Subnetworks - LANs**

- Are the nodes within the same physical location?
- Can the same adapter card be shared by multiple protocols?
- What is the performance impact of adding this new application to current LAN traffic?

**Bandwidth Management**

- How many logical point-to-point connections are required for each node to communicate with its partners? Are there adequate communications ports on these nodes?
- How many locations are involved and thus how many bandwidth managers are needed? Is a mesh network needed?
- How expandable is this network if more devices or more locations are needed in the future?

**Separate Wide Area Networks**

- What are the costs involved with a separate network - additional hardware and software in end nodes, modems, routing equipment, leased line costs, and personnel costs to manage this network?
- Are there performance or security concerns to mandate that this protocol be handled separately?

### 3.2.2 Situation 2 - Application Interoperability

Application interoperability is required when two different applications need to exchange data and communicate. The need for application interoperability might arise due to a merger of two companies, establishment of a joint business operation, or the provision of a service by an enterprise to other enterprises. In the simplest case, the two applications might simply need to convert data formats if both applications use the same application programming interface, application services and transport protocols. For instance, if both Appl-A and Appl-B used CPI-C over TCP/IP, then if Appl-A was on UNIX** system and Appl-B was on a MVS system, the data formats might need to be converted between ASCII and EBCDIC as well as any particular file formats. This conversion would take place at the application or application support levels. Electronic mail packages written to the OSI X.400 message handling system standard may also require this type conversion.

More complicated scenarios arise when conversion is required, not only of the application-level data, but also of the information from other protocol layers as well. This is a very common scenario with differing mail systems which need to exchange user messages, or database and query functions. Database formats and the inquiry tools used to interrogate the data have historically varied considerably in format and access mechanisms. Likewise, there is a range of transport protocols supported. Thus, not only will data formats need to be changed, but also the application service and transport protocols may need conversion.

For this general situation, we are assuming that a customer has a *single physical network* (either WAN or LAN), running a *single transport protocol* currently (Tport-A), and this network is controlled by one organization. If the customer can rewrite the applications to use similar application programming interfaces, application services, and/or the same transport protocol, fewer conversions will be necessary. This scenario is illustrated in Situation 2A (Figure 65 on page 105). The more complex situation, when the application cannot be recoded, is shown in Situation 2B (Figure 67 on page 108).

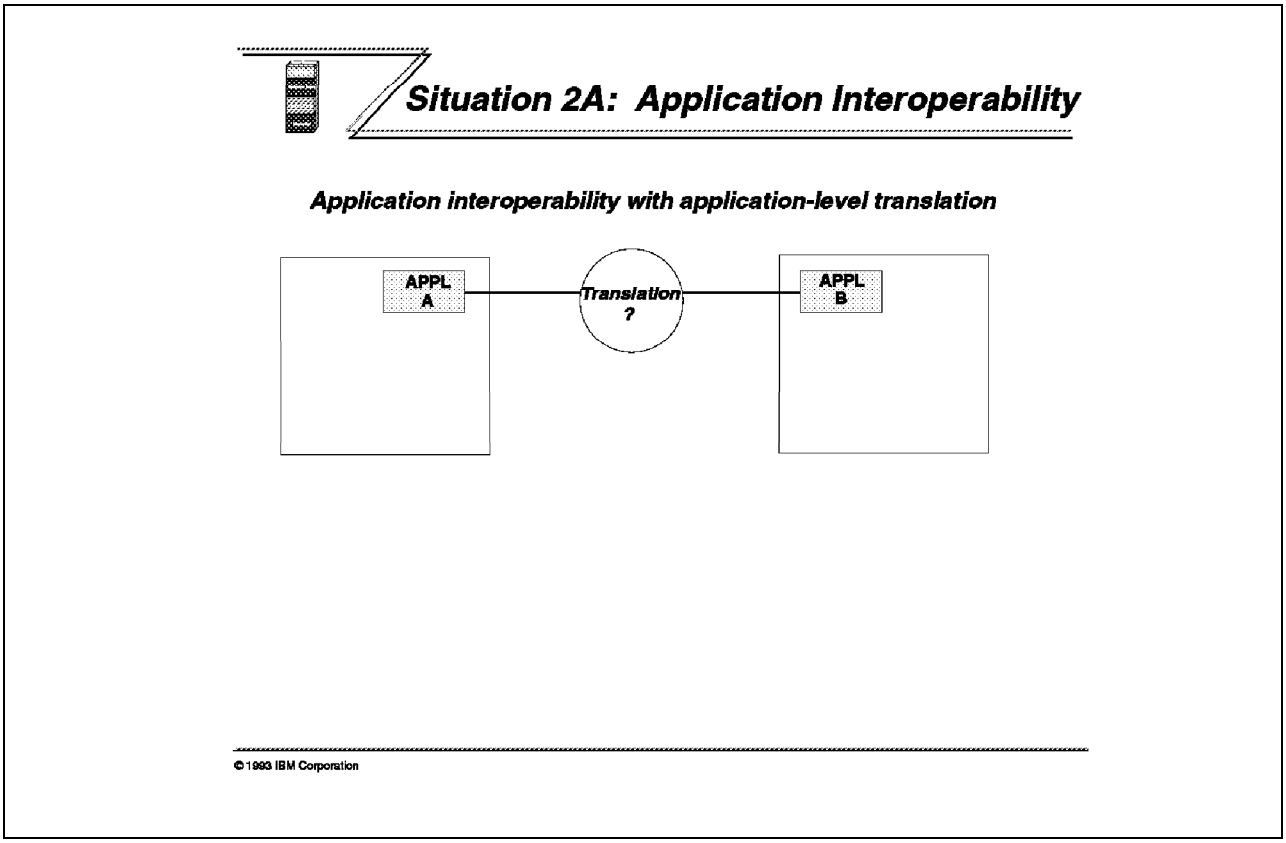



*Figure 65. Situation 2A: Application Interoperability with Application-Level Translation*

If the customer has the flexibility to recode part or all of the application, there are many choices. There are basically two levels of complexity involved. The first level of complexity involves the choice of a common API and Application Services level, therefore minimizing the amount of conversion that must take place at these levels. The second level of complexity involves the selection of a common transport protocol.

If one of the Application Support-level choices shown in Figure 66 on page 106 is selected, these technologies would provide the consistent API and Applications Services-level required, as well as permitting the utilization of more than one transport protocol, hopefully a transport protocol that is compatible with the existing user network.

There may also be situations where the customer chooses an industry standard API and Application Services approach, such as RPC for TCP/IP, but has a problem with the transport network (for example, OSI). A mixed-protocol stack approach, such as with MPTN or mixed-protocol standards (RFCs or other standards), might be appropriate to transmit the data across the network, and then the user code would still need to convert any data format differences at the highest layers. Encapsulation could also be used for this data transmission, assuming that the user application is still prepared for the upper layer conversions.

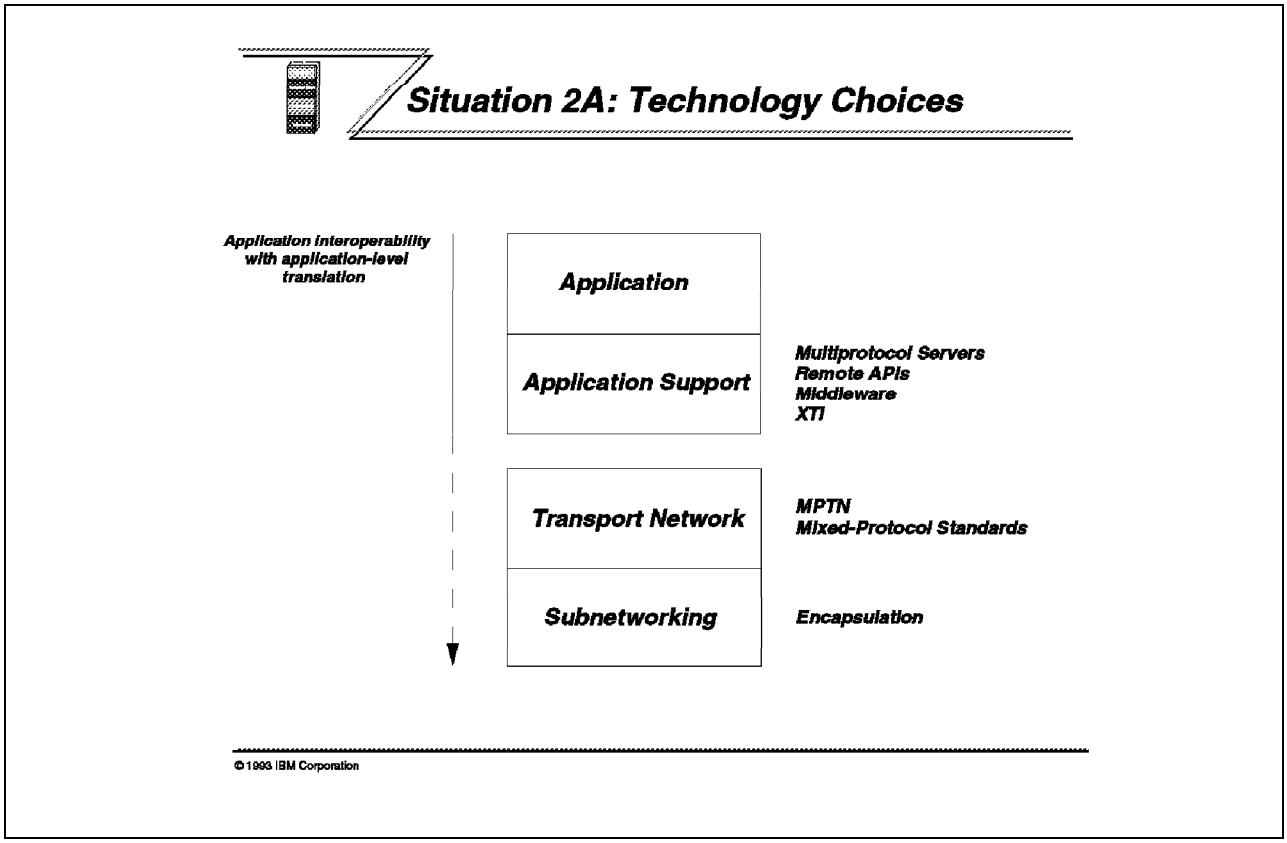



*Figure 66. Situation 2A: Technology Choices*

As before, the marketplace will determined what is actually available for a particular desired combination of application services, transport protocols and operating systems. Considerations which should be kept in mind as these technologies are evaluated include:

**Multi-Protocol Servers**

- Is an end-to-end connection required? Or is it adequate for the server to provide translation capabilities for various applications?
- Is this a mission-critical application which should be dependent on a potential single point of failure?
- How much traffic can a particular server handle efficiently? How can additional users be handled?
- Can additional transport protocols be supported? Can additional applications be supported?

**Remote APIs**

- What APIs are supported over which protocols - does the correct combination exist?
- Does this “skinny client” provide enough functionality to satisfy the application requirements?

**Middleware**

- Does this middleware implementation exist on all required operating systems in the end nodes?
- How much flexibility does the middleware application programming interface give to the application developers?
- What are the extra facilities provided by the middleware implementation in terms of directory, security, and recovery?

**XTI**

- Is the new transport protocol a currently available XTI protocol, such as TCP/IP, NetBIOS, or OSI?
- If this is a new-customer developed application, can communications software and compilers be found that utilize this interface for the currently installed hardware and operating system platforms?
- If flexibility is desired in this customer-developed application, can the application be coded with just the necessary subset of the transport functions?
- If an off-the-shelf purchased application, has it been coded to utilize the XTI interface? If so, does it support the transport protocol you want?

**MPTN**

- Do MPTN implementations exist for the necessary protocols?
- If MPTN implementations exist for these protocols, do they exist on the required operating system platforms in the end nodes? If not, do they exist for operating system platforms which could be used as a gateway?

**Mixed-Protocol Standards**

- Is it a supported application/transport combination currently supported by the RFCs or another standard, such as a NetBIOS or OSI application, which needs to run over TCP/IP?
- Do the protocol stacks installed in the end nodes support the standards?

**Encapsulation**

- Will this encapsulating software be placed in the end nodes? Or will it be placed in a separate node, requiring additional hardware?
- How is the encapsulation actually performed? Are Data Link Control protocols properly terminated and emulated? Are the transport-level protocols properly emulated? Is filtering of unnecessary packets performed?
- What are the performance characteristics of this encapsulation software? Is this acceptable?
- How are the different network addresses handled? What directory services exist?
- How expandable is this approach if the network grows and more nodes are added?

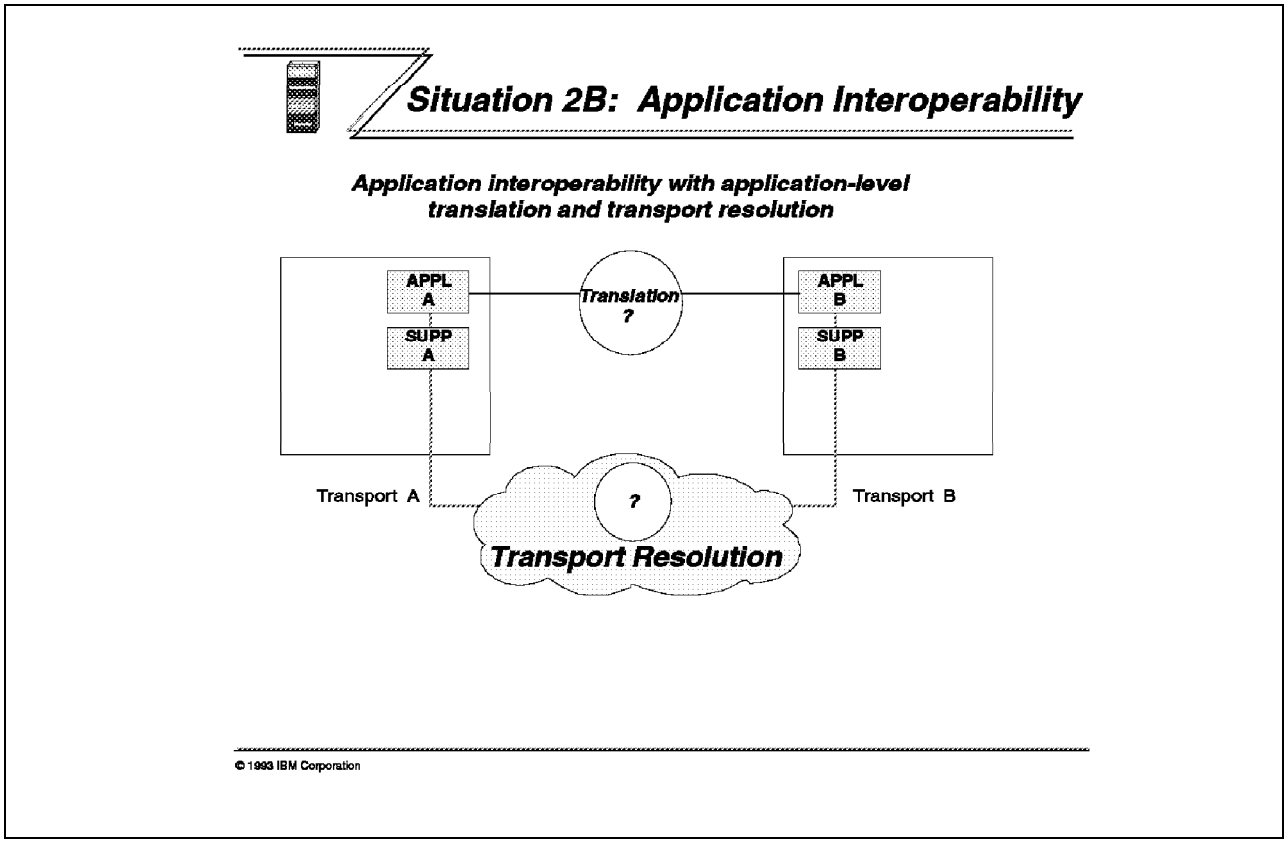



*Figure 67. Situation 2B: Application Interoperability with Application-Level Translation and Transport Resolution*

If there is no flexibility in recoding the application, then the situation shown in Figure 67 occurs. Usually multiple protocol layers must be converted, including the transport protocol. There are fewer solution possibilities in this case, as shown in Figure 68 on page 109.

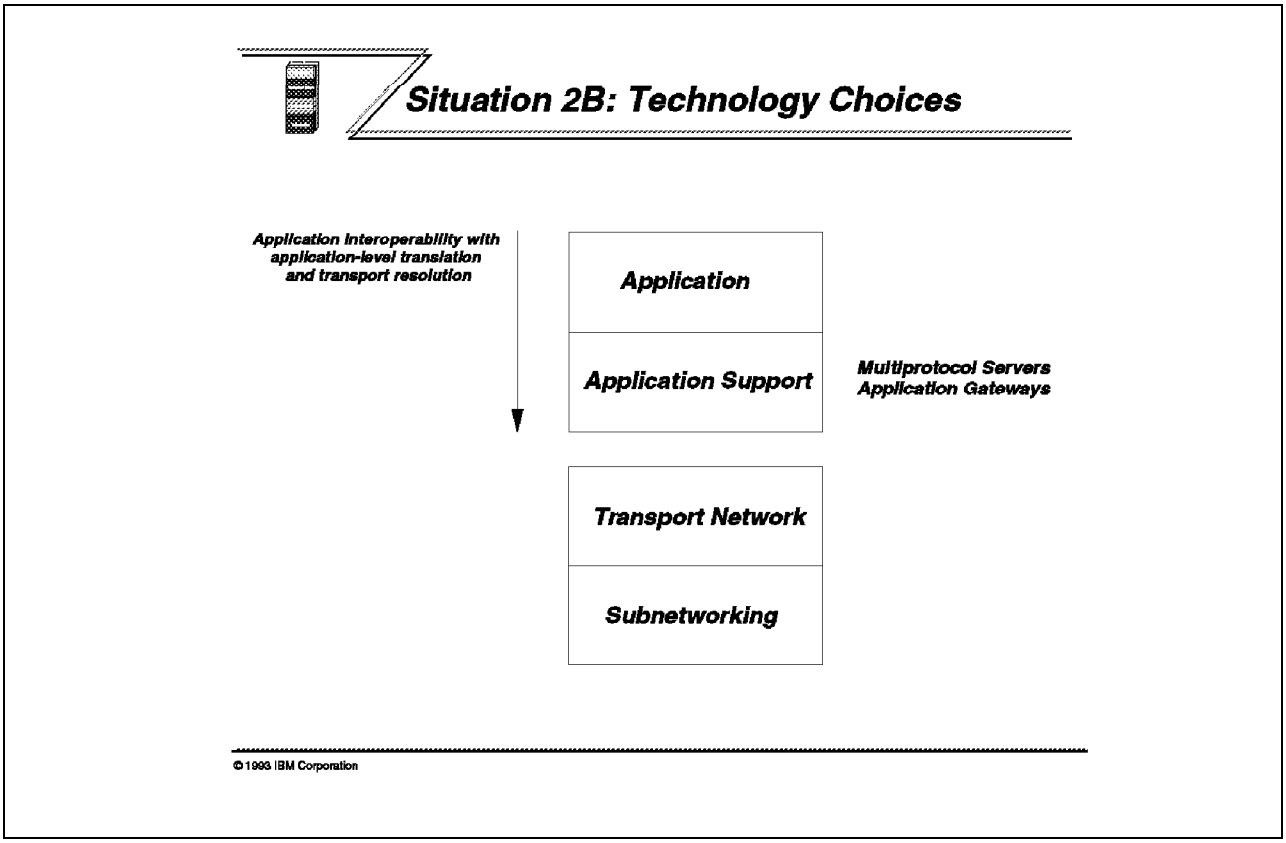



*Figure 68. Situation 2B: Technology Choices*

Both of these solutions can handle the necessary conversions at the upper levels of the protocol stacks, as well as the transport differences. The fundamental difference between the two approaches is whether end-to-end connectivity is required between the applications. Application gateways provide this end-to-end capability; multi-protocol servers do not. It might be critical for a database access application to maintain end-to-end connectivity, but perhaps not for an electronic mail “mailbox” application to do so. The choice of particular implementation will depend upon what is available in the marketplace for the desired application type and platforms.

Considerations to be kept in mind as these technologies are evaluated include the following:

**Multi-Protocol Servers**

- Is an end-to-end connection required? Or is it adequate for the server to provide translation capabilities for various applications?
- Is this a mission-critical application which should be dependent on a potential single point of failure?
- How much traffic can a particular server handle efficiently? How can additional users be handled?
- Can additional transport protocols be supported? Can additional applications be supported?

**Application Gateway**

- Does full translation capability exist to communicate with the remote application, or is just a subset of functions available?

- Is this a mission-critical application which should be dependent on a potential single point of failure?
- How much traffic can a particular gateway handle efficiently? How can additional users be handled?
- Can additional transport protocols be supported? Can additional applications be supported?

### 3.2.3 Situation 3 - Network Interconnection

As organizations within a single enterprise need to communicate to start a new business initiative, or companies merge, the need for existing networks to interconnect occurs. Interconnection always involves physically connecting two separate networks, which might also be some distance apart. Also, it assumes that the same application will be used in both networks for exchanging information. Thus, two steps are usually involved in network interconnection: first, addition of the application to the networks (if needed), and second, establishment of physical connectivity between these networks.

When evaluating the various alternatives available to establish the physical connectivity, two situations commonly arise. In the first situation, a low-cost, single gateway between the two networks is desired: this is Situation 3A (Figure 69). In the second case, an intervening network is placed between the two networks. This intervening network may be owned by one of the organizations, or in fact may be owned by a third organization. This intervening network may use a different transport protocol than the application requires. This scenario is discussed in Situation 3B (Figure 71 on page 114).

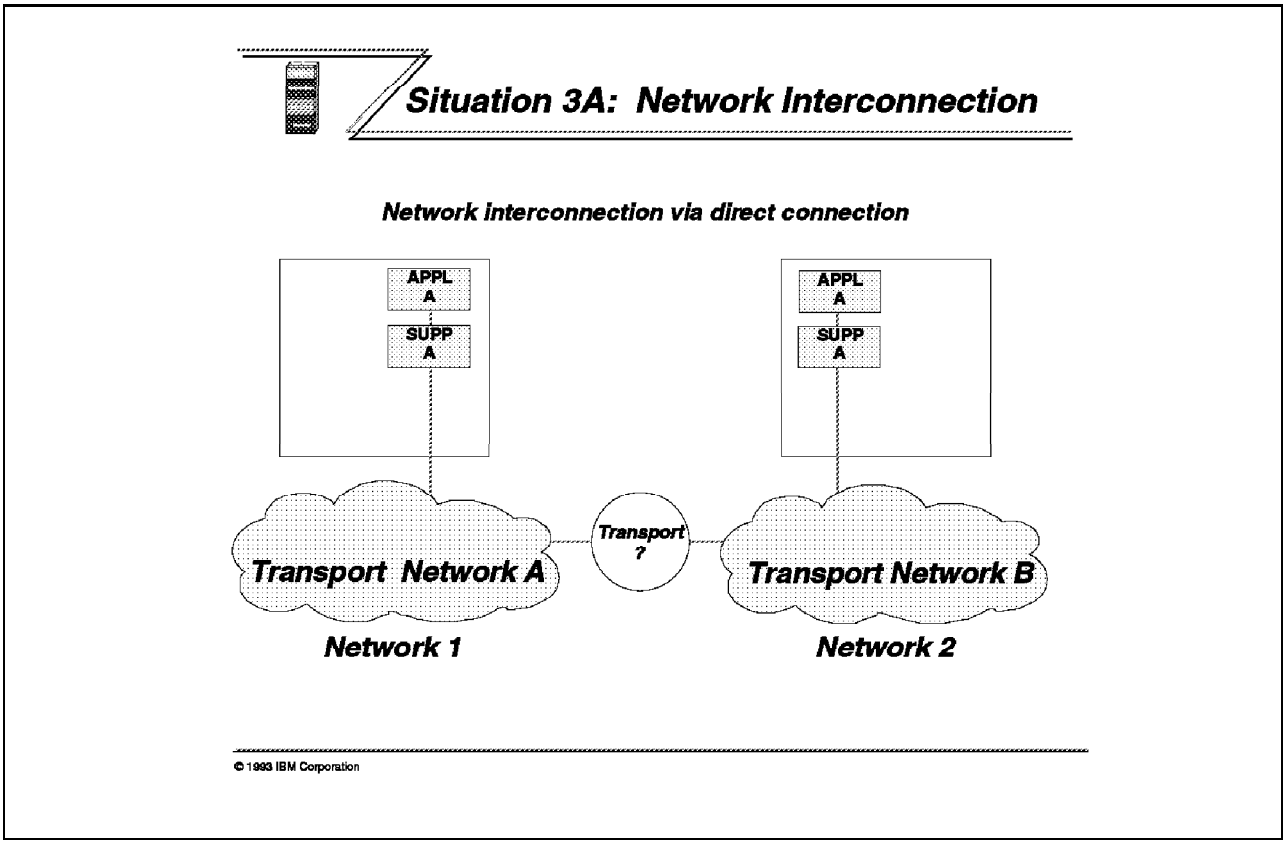



*Figure 69. Situation 3A: Network Interconnection via Direct Connection*

For Situation 3A, the interconnection involves the addition of an existing application in one network to the second network (often referred to as extending the “reach” of the application). As shown in Figure 69, Appl-A which uses transport protocol A already exists in Network #1, and now must be added to Network #2, which currently utilizes transport protocol B. This forces Network #2 to handle the second protocol.

Although this application addition to Network #2 might initially appear to be similar to Situation 1, there are several significant differences. In Situation 1, it was assumed that a single network under the control of a single organization was involved in the decision to add a new application. The issues become more complex in this situation, when the two networks are physically separate, perhaps quite a distance apart, and often under the control of different organizations.

As shown in Figure 69 on page 111, Network A must *directly* connect to Network B. For instance, Network A might be an SNA wide area network with 3745 front end processors, and Network B might be a token ring network with Novell Netware and IPX transport protocol - and now the Novell Netware users must access an SNA application. Multiple issues exist in this type of scenario, such as how to physically connect the local area network to the wide area network, and how to integrate the new transport protocol into the LAN users' environment.

Due to the condition that direct connection is required, with no intervening network, the following solutions shown in Figure 70 are possible. These are solutions that can exist in a single gateway configuration. Note that these solutions span quite a range; from those that focus strictly on the subnetwork level to solutions that operate at the upper layers.

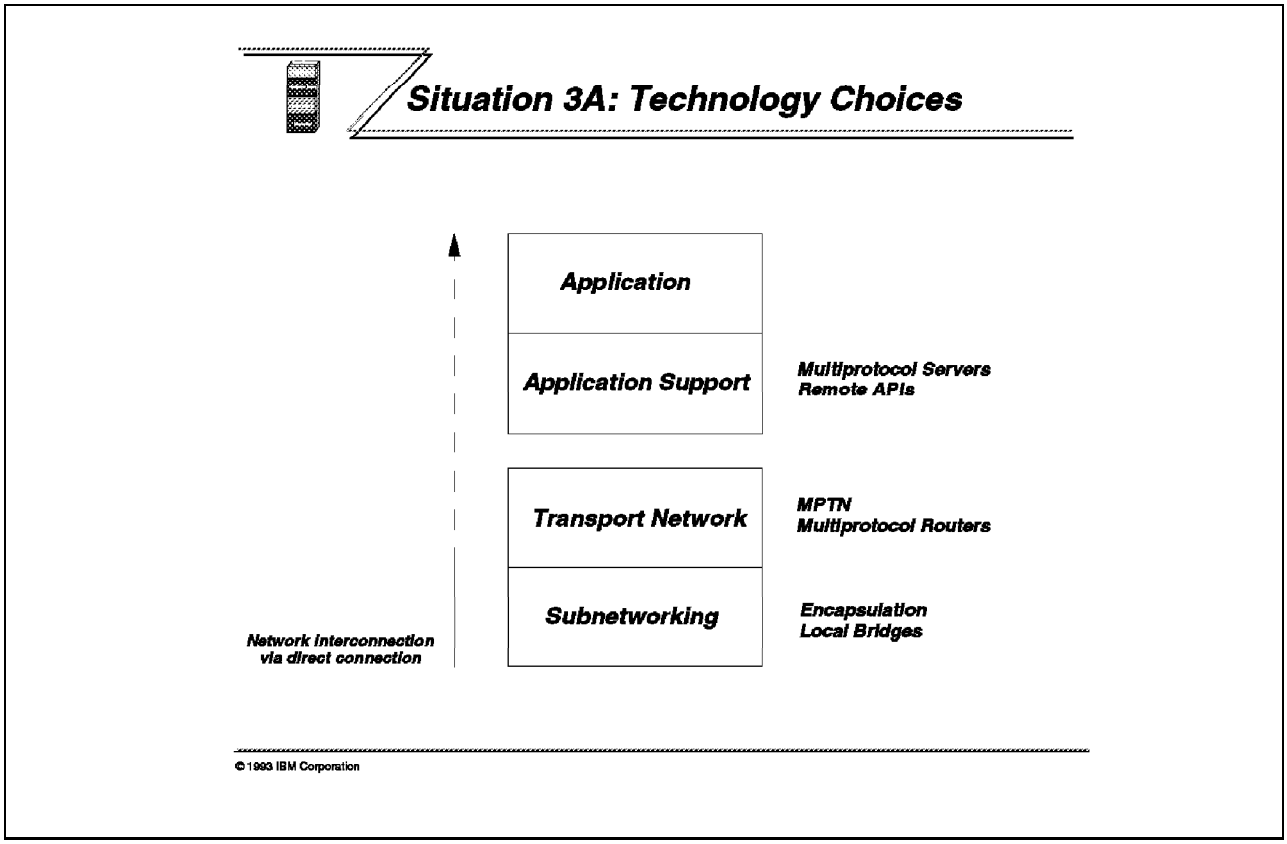



*Figure 70. Situation 3A: Technology Choices*

If the two networks are LANs that are physically located next to each other, then a local bridge can be considered. A multi-protocol router may also be considered in this local situation, especially since it may provide additional filtering capabilities over the bridge.

For wide area networks, particularly those networks which desire to restrict the number of transport protocols, encapsulation can be used. Encapsulation will be performed at an end node, the de-encapsulation at a gateway which physically connects the two networks. MPTN can similarly be used in a single gateway configuration, with one end node being the access node. Multi-protocol servers and a Remote API server can provide the physical connectivity as well as the application connectivity.

One might also wish to consider where the data is physically located in the network in deciding among these solutions. For instance, a multi-protocol server allows the data to be stored in one central facility that all users can access. For distributed data, remote APIs, encapsulation, and MPTN might be appropriate.

Considerations to be kept in mind as these technologies are evaluated include the following:

**Local Bridges**

- Are the LANs located in close enough proximity for a local bridge to attach?
- Is there too much unnecessary traffic to require filtering?

**Encapsulation**

- Will this encapsulating software be placed in the end nodes? Or will it be placed in a separate node, requiring additional hardware?
- How is the encapsulation actually performed? Are Data Link Control protocols properly terminated and emulated? Are the transport-level protocols properly emulated? Is filtering of unnecessary packets performed?
- What are the performance characteristics of this encapsulation software? Is this acceptable?
- How are the different network addresses handled? What directory services exist?
- How expandable is this approach if the network grows and more nodes are added?

**Multi-Protocol Routers**

- Are all needed protocols handled by the router?
- Are there any Data Link Control-level concerns? Does the router properly handle these protocols?
- What happens in congestion situations? Are all protocols equal? Can the traffic be prioritized?
- How much filtering is available for each protocol?

**MPTN**

- Do MPTN implementations exist for the necessary protocols?
- If MPTN implementations exist for these protocols, do they exist on the required operating system platforms in the end nodes? If not, do they exist for operating system platforms, which could be used as a gateway?

**Remote APIs**

- What APIs are supported over which protocols? Does the correct combination exist?
- Does this “skinny client” provide enough functionality to satisfy the application requirements?

**Multi-Protocol Servers**

- Is an end-to-end connection required? Or is it adequate for the server to provide translation capabilities for various applications?
- Is this a mission-critical application which should be dependent on a potential single point of failure?
- How much traffic can a particular server handle efficiently? How can additional users be handled?
- Can additional transport protocols be supported? Can additional applications be supported?

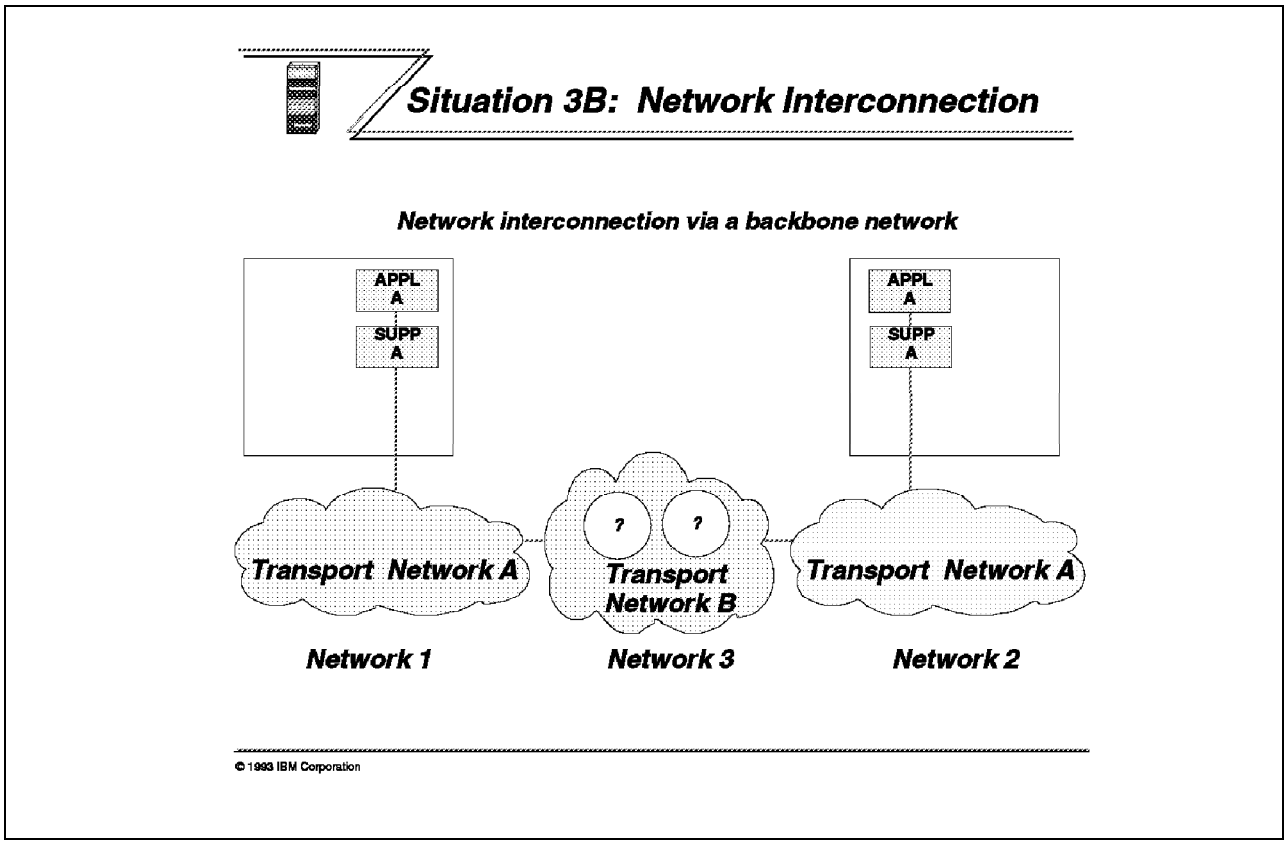



*Figure 71. Situation 3A: Network Interconnection via a Backbone Network*

This second case involves an intermediate network between two existing networks of a single transport protocol type. The intermediate network, Network #3 in Figure 71, is often called a *backbone* network. The backbone network might be a high-speed local area network (such as a FDDI network interconnecting slower-speed local area networks), a public carrier network (such as a value added packet switch network), or a private wide area network (such as an SNA network). The backbone network might be controlled by a different organization than Network #1 and Network #2. In this scenario, it is assumed that both Network #1 and Network #2 have Appl-A, which utilizes transport protocol A, but the backbone network might utilize transport protocol B. For instance, Appl-A might use TCP/IP, but Network #3 might use SNA. Again, there are physical connectivity problems as well as transport protocol resolution.

Solutions which have double gateway implementations, such as those shown in Figure 44 on page 66, will work quite well in this situation. Note that these solutions focus mainly on the subnetworking and transport layers; the upper application layers are not affected.

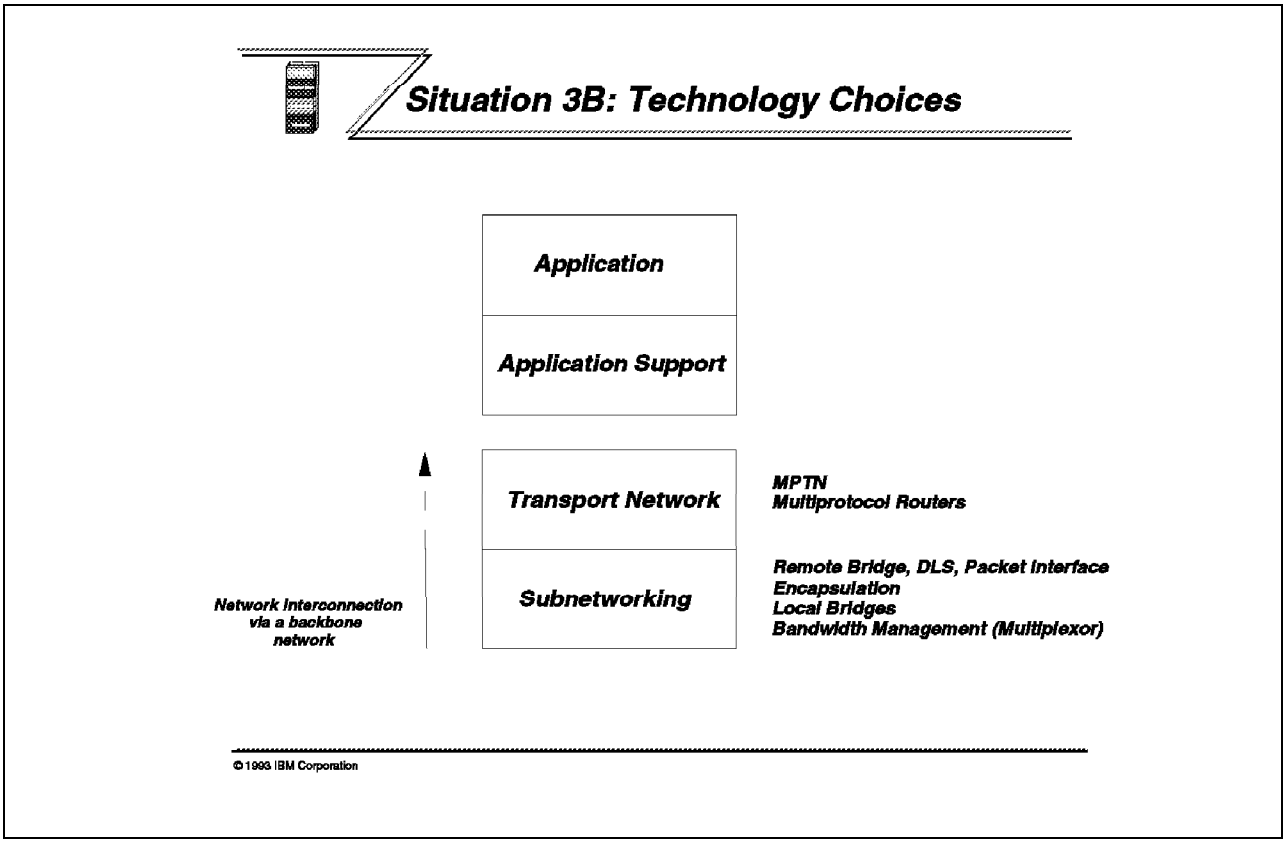



*Figure 72. Situation 3B: Technology Choices*

Considerations to be kept in mind as these technologies are evaluated include the following:

**Bandwidth Management**

- How many logical point-to-point connections are required for each node to communicate with its partners? Are there adequate communications ports on these nodes?
- How many locations are involved and thus how many bandwidth managers are needed? Is a mesh network needed?
- How expandable is this network if more devices or more locations are needed in the future?

**Local Bridges**

- Are the LANs located in close enough proximity for a local bridge to attach?
- Is there too much unnecessary traffic to require filtering?

**Extended Subnetworking**

- Is a public service (Frame Relay, X.25, SMDS) preferable to a private network? What are the costs involved, and how much control over the network is desired?
- Is this critical traffic? Does a prioritization scheme exist in this extended subnetworking implementation to handle higher priority traffic?
- How do the end nodes interface to the extended subnetwork? What is the cost of upgrading equipment or adding additional equipment?

**Encapsulation**

- Will this encapsulating software be placed in the end nodes - or in a separate node, requiring additional hardware?
- How is the encapsulation actually performed? Are Data Link Control protocols properly terminated and emulated? Are the transport-level

protocols properly emulated? Is filtering of unnecessary packets performed?

- What are the performance characteristics of this encapsulation software? Is this acceptable?
- How are the different network addresses handled? What directory services exist?
- How expandable is this approach if the network grows and more nodes are added?

**Multi-Protocol Routers**

- Are all needed protocols handled by the router?
- Are there any Data Link Control-level concerns? Does the router properly handle these protocols?
- What happens in congestion situations? Are all protocols equal? Can the traffic be prioritized?
- How much filtering is available for each protocol?

**MPTN**

- Do MPTN implementations exist for the necessary protocols?
- If MPTN implementations exist for these protocols, do they exist on the required operating system platforms in the end nodes? If not, do they exist for operating system platforms which could be used as a gateway?

### 3.2.4 Situation 4 - Network Consolidation

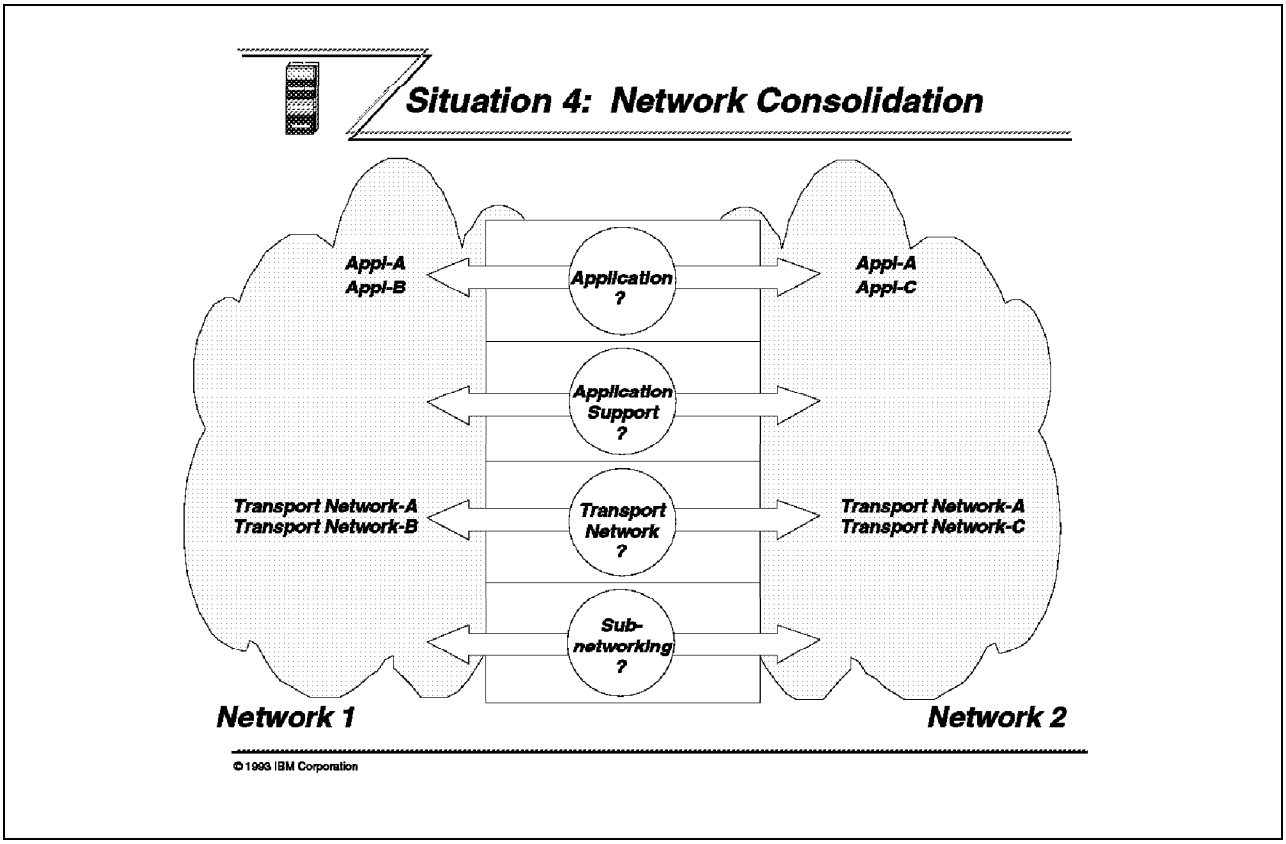



*Figure 73. Situation 4: Network Consolidation*

The next progressive step from Situation 3, Network Interconnection, is Network Consolidation. Assuming that networks are controlled by the same organization, and that they overlap at some of the same physical sites, it makes sense to reduce the number of physical networks to save costs. The first phase of network consolidation involves sharing the subnetworks. However, network consolidation can get more involved, and thus more money can be saved, if transport protocols can be reduced — perhaps down to a single transport protocol. A single transport protocol is much easier to administer and manage than multiple protocols.

But consolidating on fewer transport protocols clearly impacts existing applications and future application development. Since application services are often tied to the transport protocol, the choice of a single transport protocol restricts the choices for application development. For instance, consider Figure 73. Since both Network #1 and Network #2 utilize transport protocol A due to the presence of Appl-A in both networks, a decision might be made to use just transport protocol A for both networks. What happens to both Appl-B and Appl-C? And if new Appl-D is obtained, what happens if the application services needed by Appl-D don't typically exist with transport protocol A?

Network designers must assess the total impact of network consolidation based on the three main building blocks of the network protocol stack: subnetwork, transport, and application support. It might seem quite reasonable to save line costs by saving on subnetwork facilities, as well as administration and network management personnel costs by reducing transport protocols and/or

applications. The impact to the users must be considered as well as the impact to the company's profitability if applications are discontinued or modified to accommodate a change in transport protocol. Many considerations must be weighed before embarking on this complex task. A variety of common solutions are illustrated in Figure 74. The actual solution for a particular situation will usually involve a combination of several of these technologies.

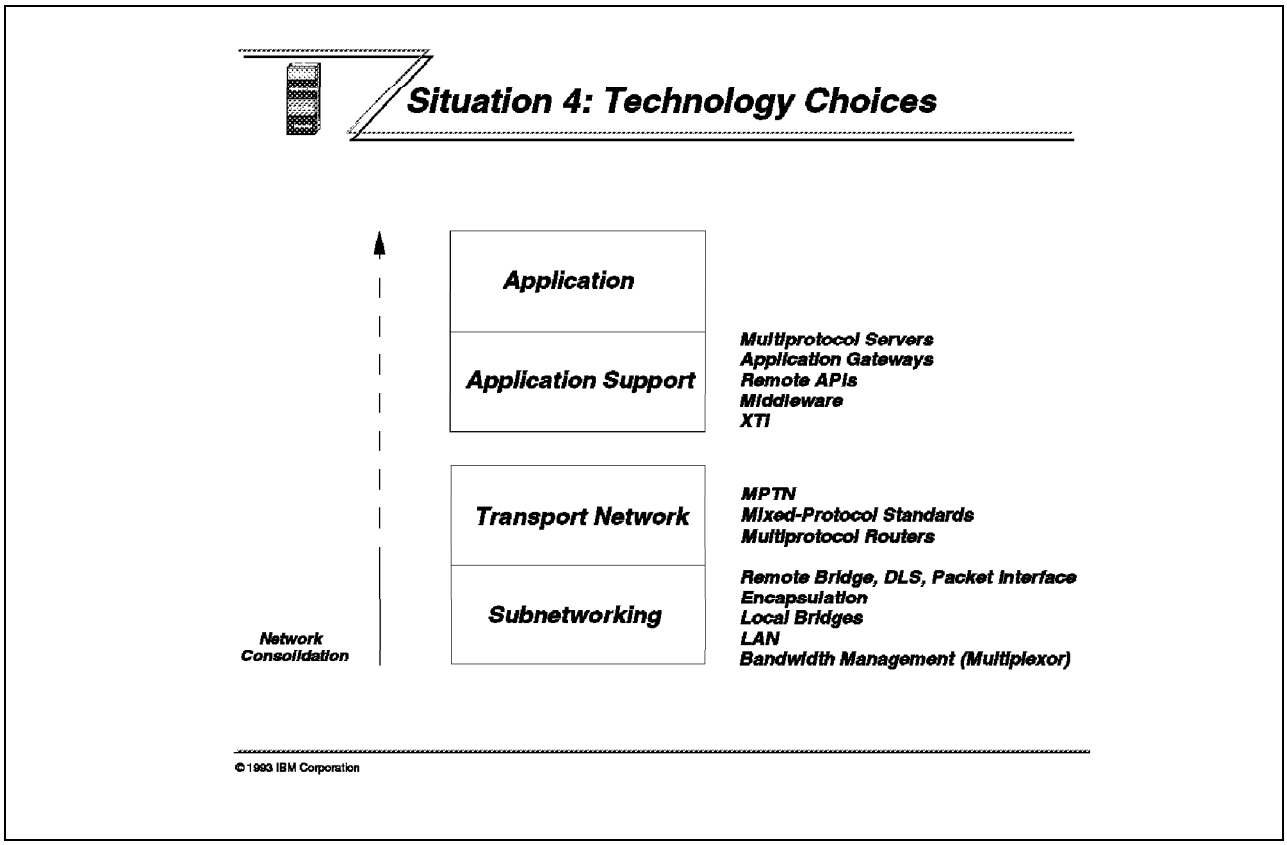



*Figure 74. Situation 4: Technology Choices*

Considerations to be kept in mind as these technologies are evaluated include the following:

**Bandwidth Management**

- How many logical point-to-point connections are required for each node to communicate with its partners? Are there adequate communications ports on these nodes?
- How many locations are involved and thus how many bandwidth managers are needed? Is a mesh network needed?
- How expandable is this network if more devices or more locations are needed in the future?

**Local Bridges**

- Are the LANs located in close enough proximity for a local bridge to attach?
- Is there too much unnecessary traffic to require filtering?

**Extended Subnetworking**

- Is a public service (Frame Relay, X.25, SMDS) preferable to a private network? What are the costs involved, and how much control over the network is desired?
- Is this critical traffic? Does a prioritization scheme exist in this extended subnetworking implementation to handle higher priority traffic?

- How do the end nodes interface to the extended subnetwork? What is the cost of upgrading equipment or adding additional equipment?

**Encapsulation**

- Will this encapsulating software be placed in the end nodes? Or qill it be placed in a separate node, requiring additional hardware?
- How is the encapsulation actually performed? Are Data Link Control protocols properly terminated and emulated? Are the transport-level protocols properly emulated? Is filtering of unnecessary packets performed?
- What are the performance characteristics of this encapsulation software? Is this acceptable?
- How are the different network addresses handled? What directory services exist?
- How expandable is this approach if the network grows and more nodes are added?

**Multi-Protocol Routers**

- Are all needed protocols handled by the router?
- Are there any Data Link Control-level concerns? Does the router properly handle these protocols?
- What happens in congestion situations? Are all protocols equal? Can the traffic be prioritized?
- How much filtering is available for each protocol?

**Mixed-Protocol Standards**

- Is it a supported application/transport combination currently supported by the RFCs or another standard, such as a NetBIOS or OSI application, which needs to run over TCP/IP?
- Do the protocol stacks installed in the end nodes support the standards?

**MPTN**

- Do MPTN implementations exist for the necessary protocols?
- If MPTN implementations exist for these protocols, do they exist on the required operating system platforms in the end nodes? If not, do they exist for operating system platforms which could be used as a gateway?

**XTI**

- Is the new transport protocol a currently available XTI protocol, such as TCP/IP, NetBIOS, or OSI?
- If this is a new customer-developed application, can communications software and compilers be found that utilize this interface for the currently installed hardware and operating system platforms?
- If flexibility is desired in this customer-developed application, can the application be coded with just the necessary subset of the transport functions?
- If this is an off-the-shelf purchased application, has it been coded to utilize the XTI interface? If so, does it support the transport protocol you want?

**Middleware**

- Does this middleware implementation exist on all required operating systems in the end nodes?
- How much flexibility does the middleware application programming interface give to the application developers?
- What are the extra facilities provided by the middleware implementation in terms of directory, security, and recovery?

**Remote APIs**

- What APIs are supported over which protocols? Does the correct combination exist?

- Does this “skinny client” provide enough functionality to satisfy the application requirements?

**Application Gateways**

- Does full translation capability exist to communicate with the remote application, or is just a subset of functions available?
- Is this a mission-critical application which should be dependent on a potential single point of failure?
- How much traffic can a particular gateway handle efficiently? How can additional users be handled?
- Can additional transport protocols be supported? Can additional applications be supported?

**Multi-Protocol Servers**

- Is an end-to-end connection required? Or is it adequate for the server to provide translation capabilities for various applications?
- Is this a mission-critical application which should be dependent on a potential single point of failure?
- How much traffic can a particular server handle efficiently? How can additional users be handled?
- Can additional transport protocols be supported? Can additional applications be supported?

## 3.3 Summary of Technology Positioning

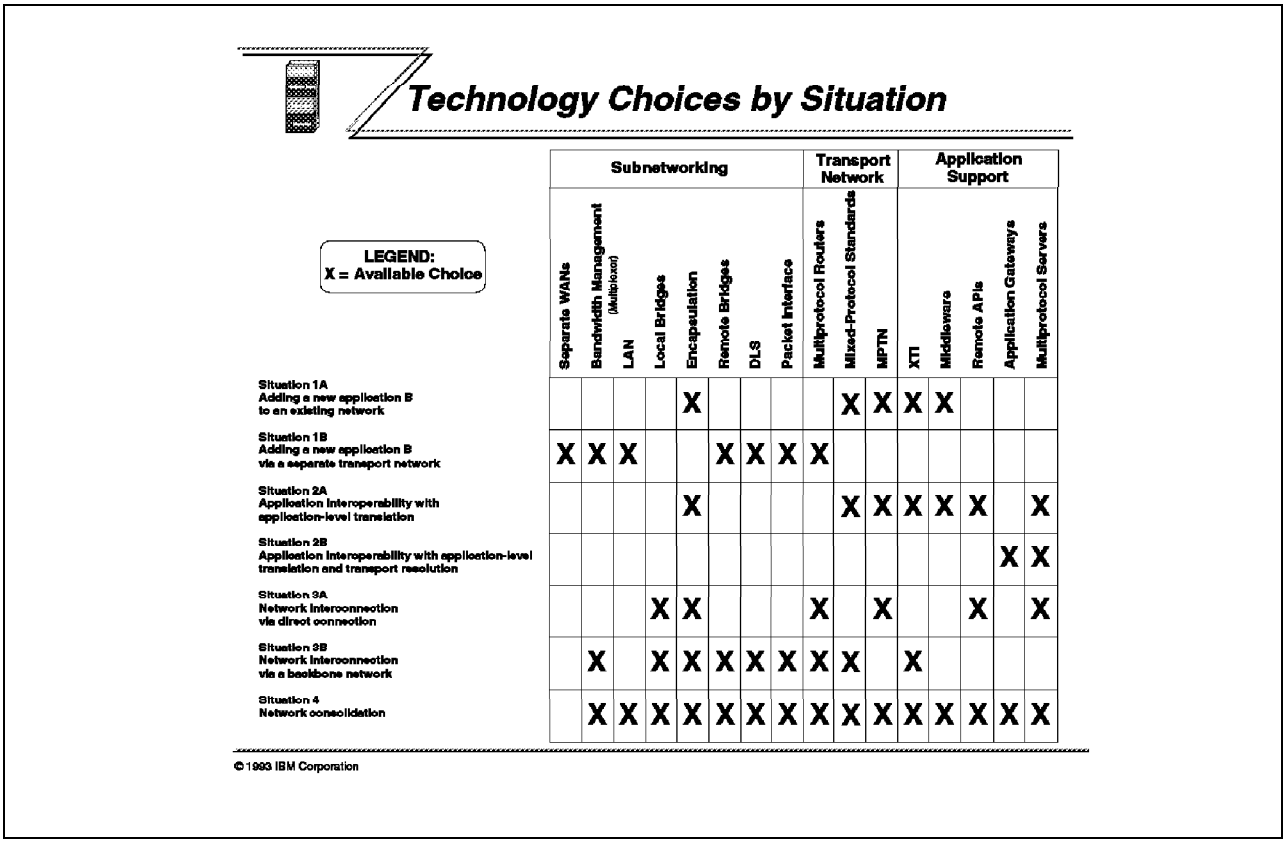



*Figure 75. Technology Choices by Situation*

Figure 75 summarizes the possible networking technology solution choices for each of the situations discussed. As we have seen, the range of techniques available for designing open networking solutions is very large. Each technique has advantages and disadvantages. In most situations, a combination of these techniques will be required to deliver the level of service and 'openness' desired. With such a diversity of problems and technologies, there is a tremendous need for a logical, systematic approach to be used to develop the flexible yet integrated network required in the 1990s. Like an architect designing a building, network designers need to architect their own plan to ensure that it can meet the ever changing business and technical environment.

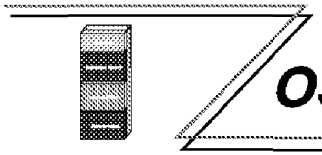
# Appendix A. Master Foil Set

© Copyright IBM Corp. 1994

# OSI Model and Our Terminology

| | ISO's OSI Model (approximation) | Our terminology | |
|---|---|---|---|
| *OSI layer 7* | application | APPL | APPLICATION |
| *OSI layers 5-7* | application services | SUPP | APPLICATION SUPPORT |
| *OSI layers 1-4* | networking | NETG | NETWORKING |

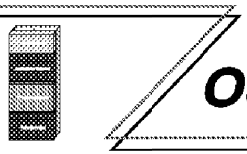
© 1993 IBM Corporation



# OSE, OSI, and Our Terminologies

| IEEE's POSIX-OSE Model | | ISO's OSI Model (approximation) | Our terminology | |
|---|---|---|---|---|
| software application | *OSI layer 7* | application | APPL | APPLICATION |
| application platform | *OSI layers 5-7* | application services | SUPP | APPLICATION SUPPORT |
| communication entities | *OSI layers 1-4* | networking | NETG | NETWORKING |

© 1993 IBM Corporation



# OSI and Our Terminology, Expanded

*OSI layer 7* — application — APPL — **APPLICATION**

*OSI layers 5-7* — application services — SUPP — **APPLICATION SUPPORT**

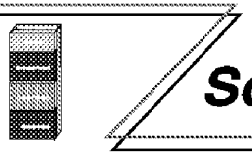
*OSI layers 1-4* — networking — TPORT — **TRANSPORT NETWORK**

SNETG — **SUBNETWORKING**

**ISO's OSI Model (approximation)**

**Our terminology**

© 1993 IBM Corporation



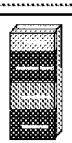
# Some Computing Equations

**Computing** = Application + Application Support + Networking

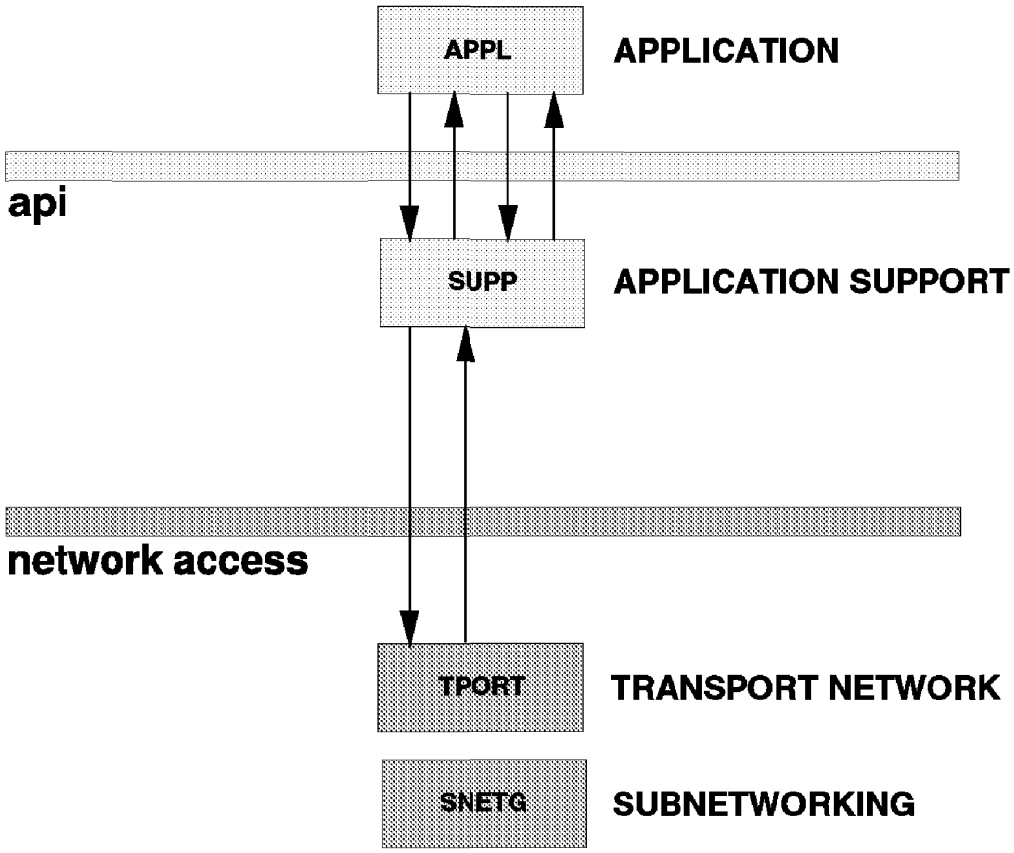
= Application + Application Support + Transport Network + Sub-Networking

= APPL + SUPP + TPORT + SNETG

© 1993 IBM Corporation

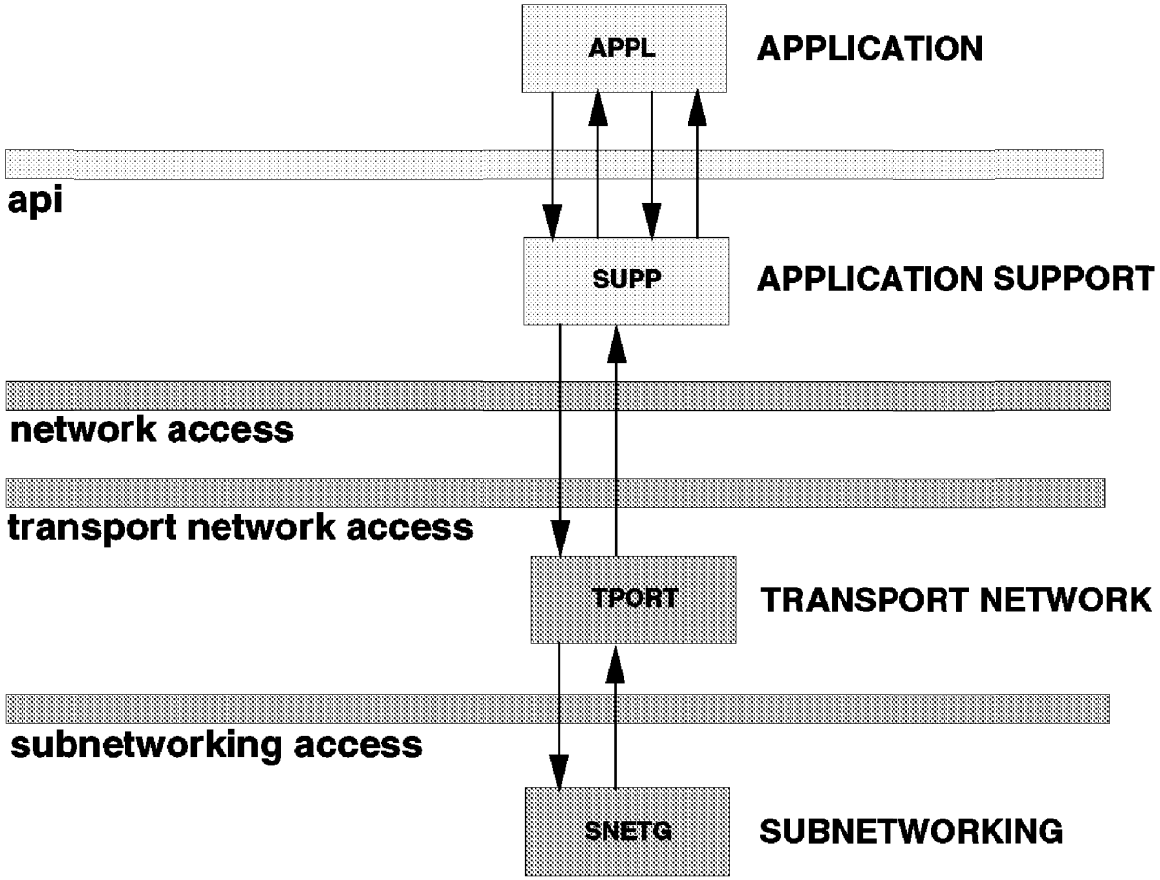


# Stacks of Software



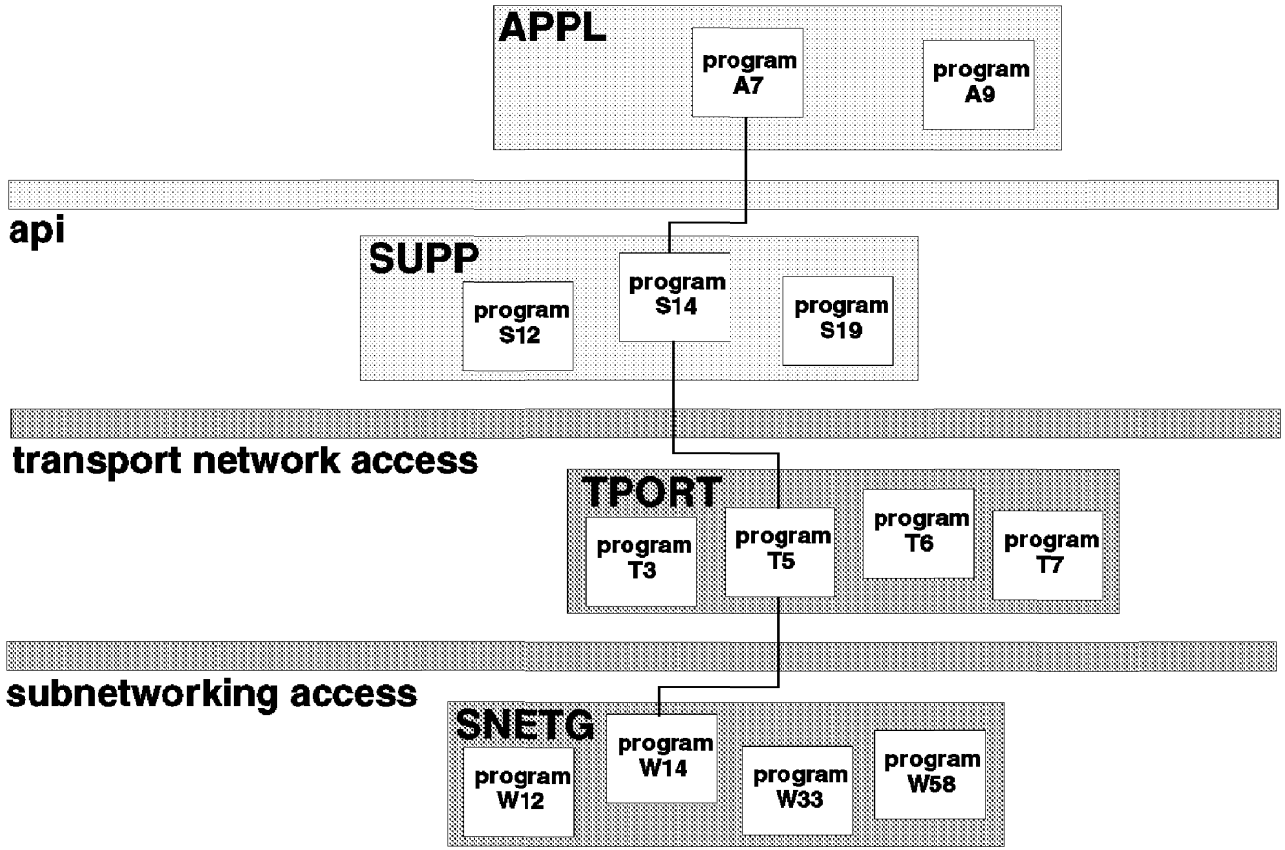

© 1993 IBM Corporation

# Network Access Mechanism

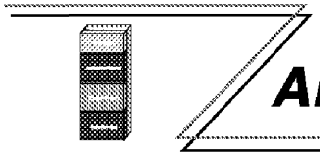



© 1993 IBM Corporation

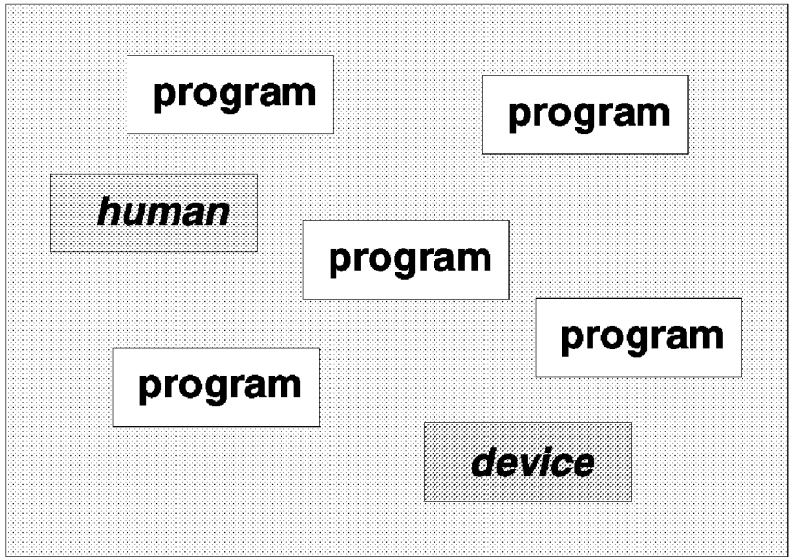
# Switching Points




© 1993 IBM Corporation



# APPLICATION Definition

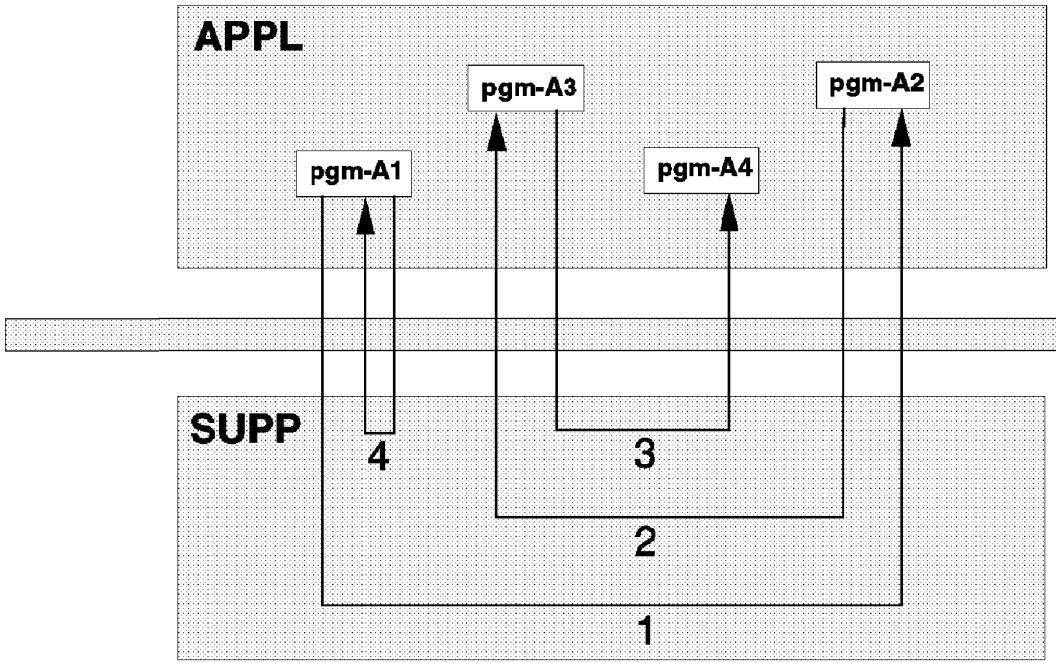



**APPLICATION**

A collection of programs (and human and device operations) that serves some useful purpose.

© 1993 IBM Corporation

# Non-Distributed Application



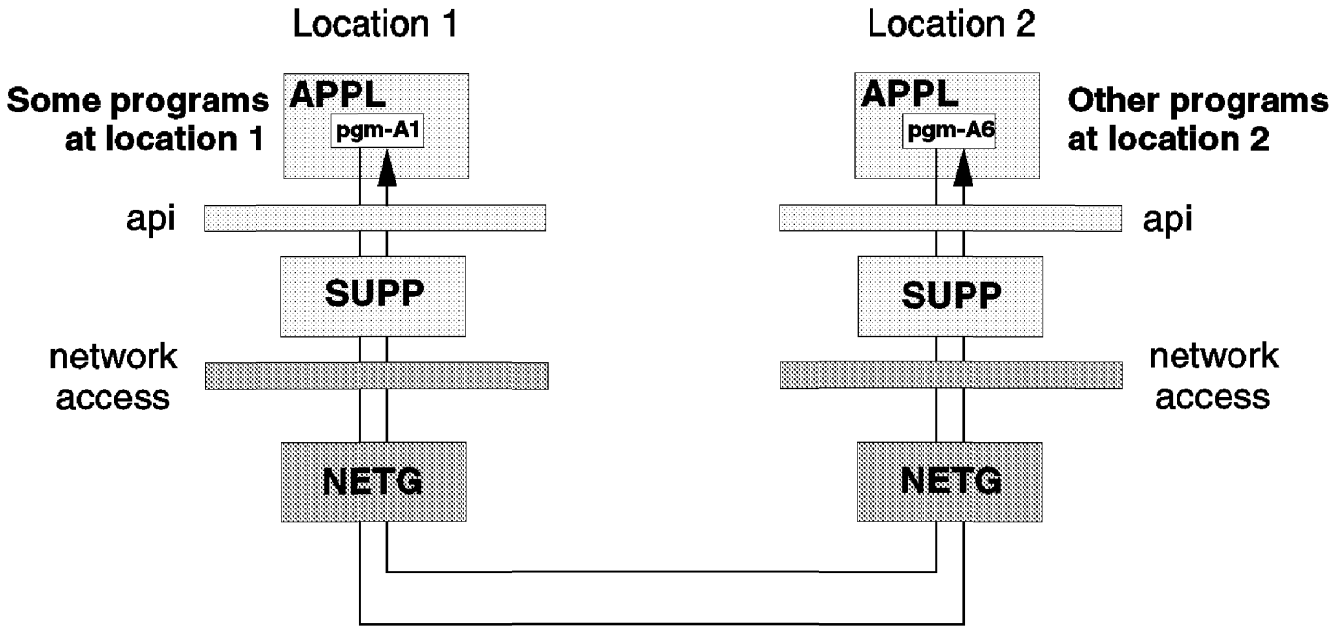

**APPLICATION**

All application programs at location 1.

**APPLICATION SUPPORT**

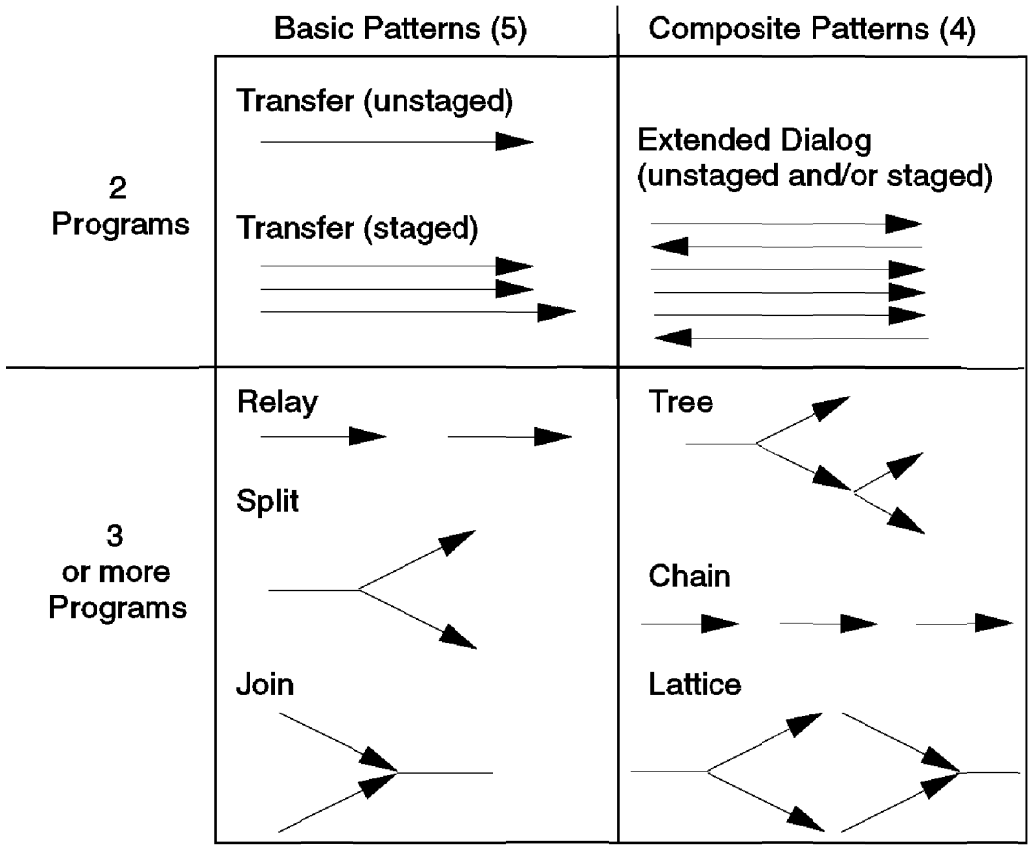
© 1993 IBM Corporation

# Distributed Application




© 1993 IBM Corporation

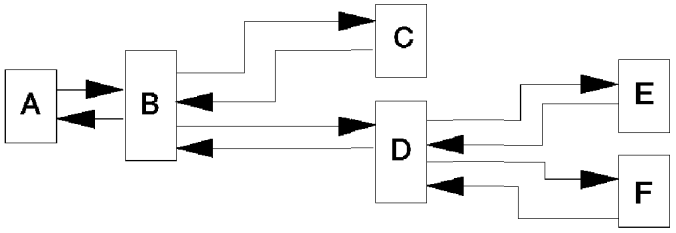
# Information Flow Patterns



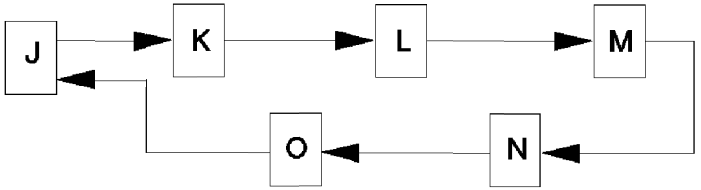

© 1993 IBM Corporation

# Composite Information Flow Patterns

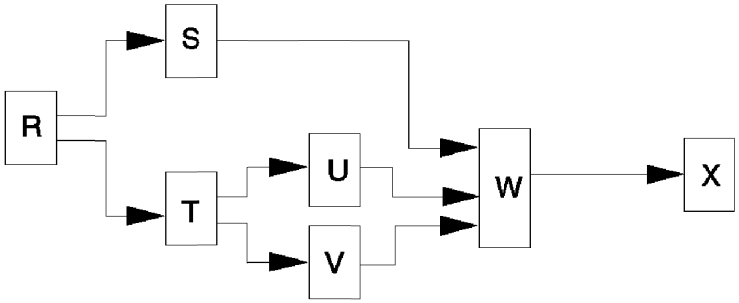
**TREE**




**CHAIN**

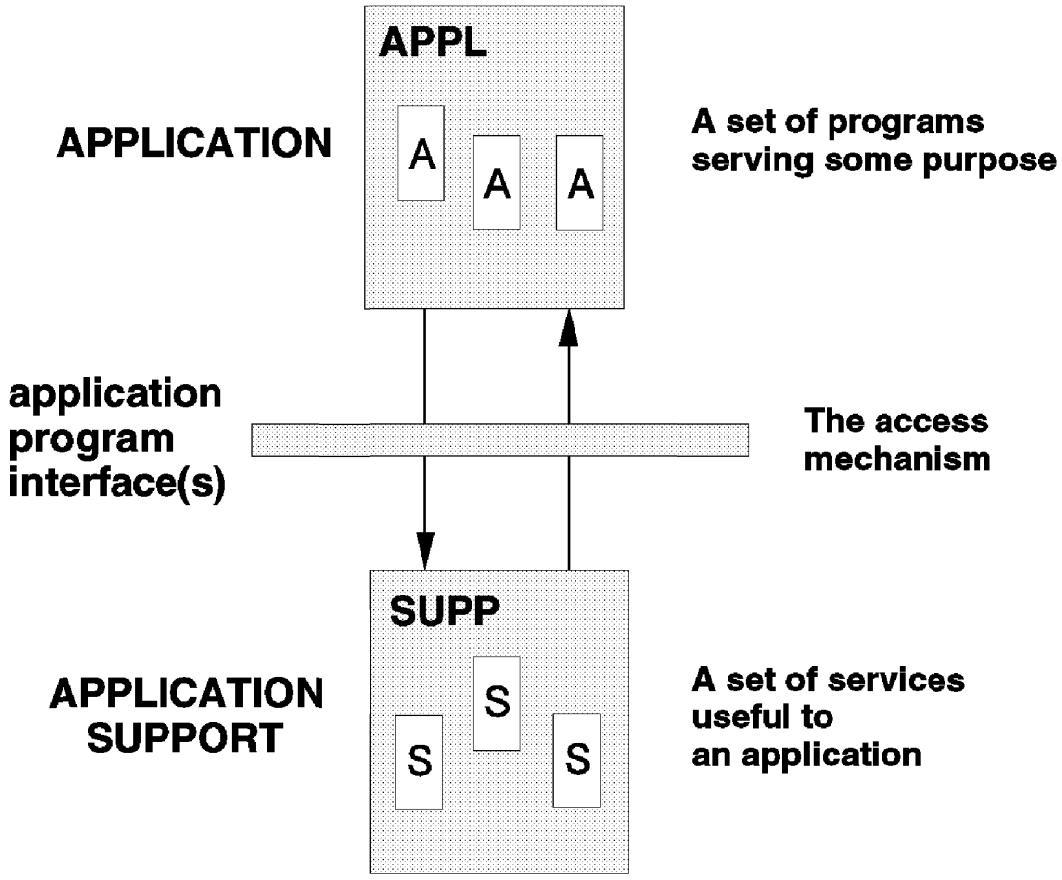



**LATTICE**



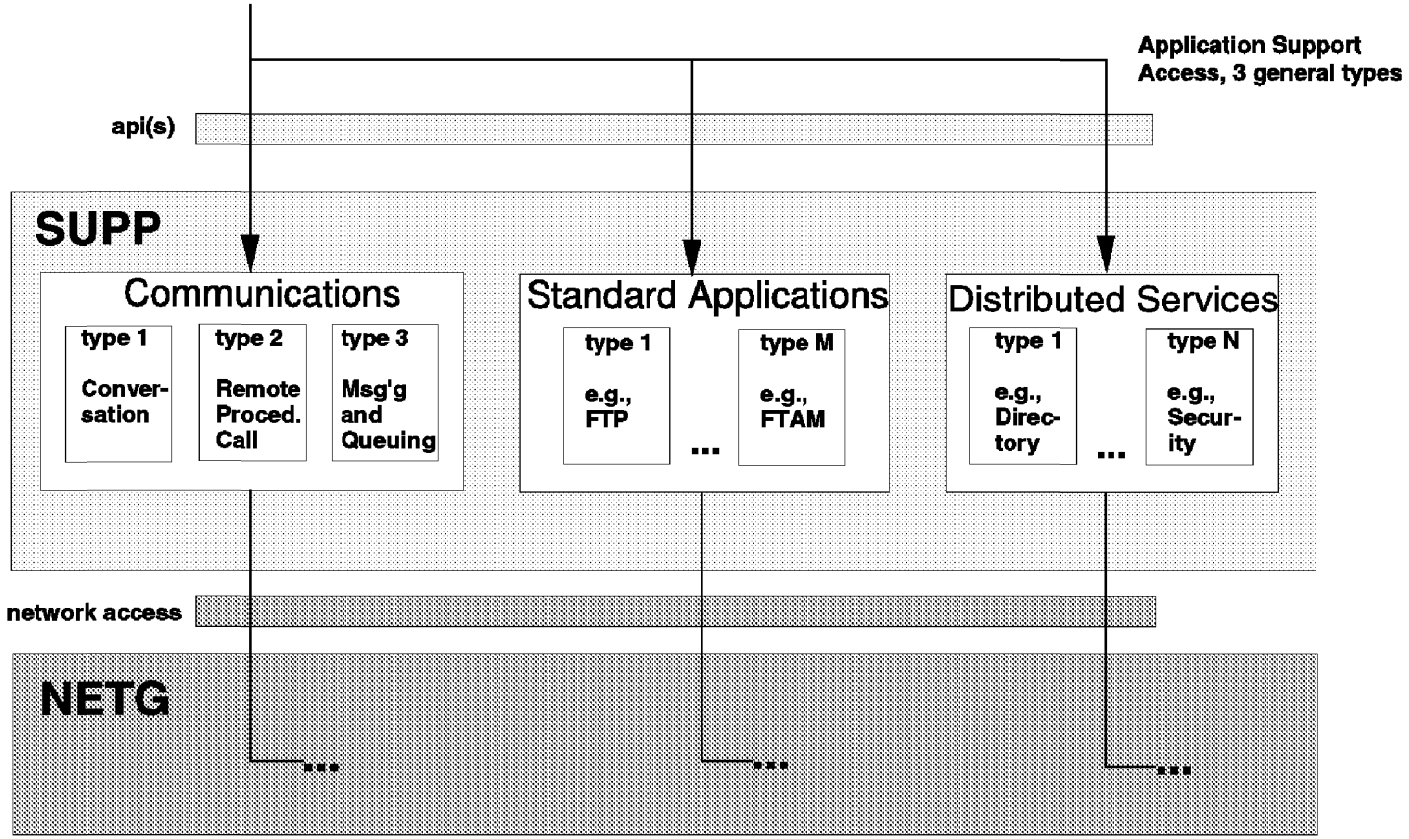

© 1993 IBM Corporation

# APPLICATION SUPPORT Definition



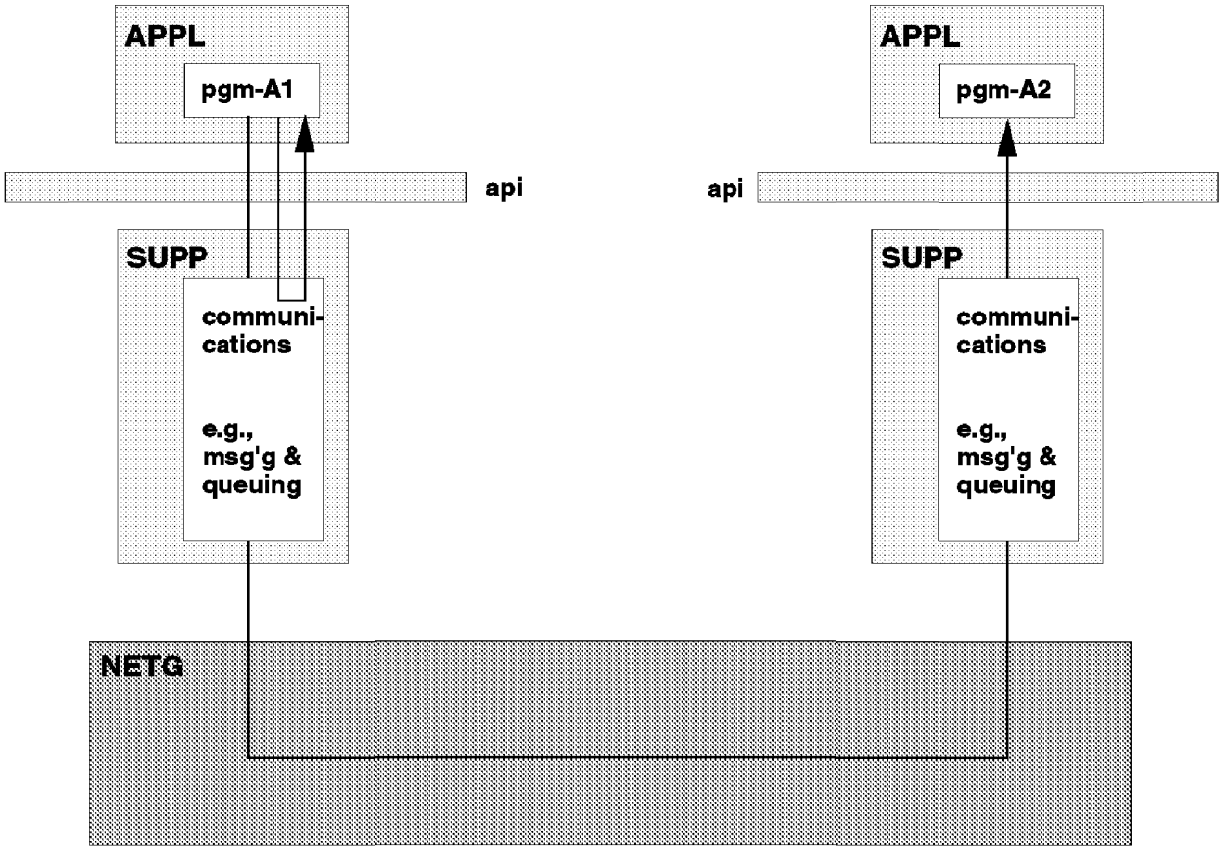

© 1993 IBM Corporation

# Types of Application Support



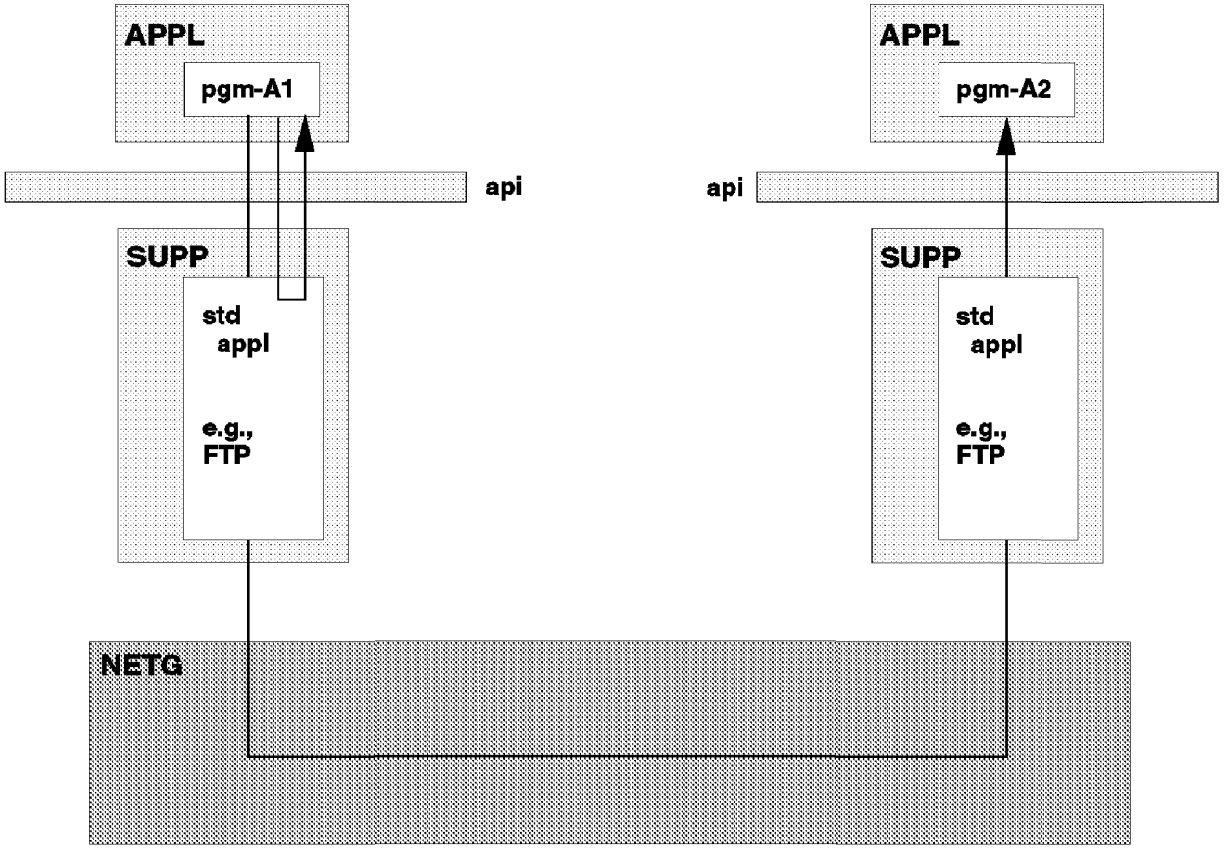

© 1993 IBM Corporation

# Communications Support



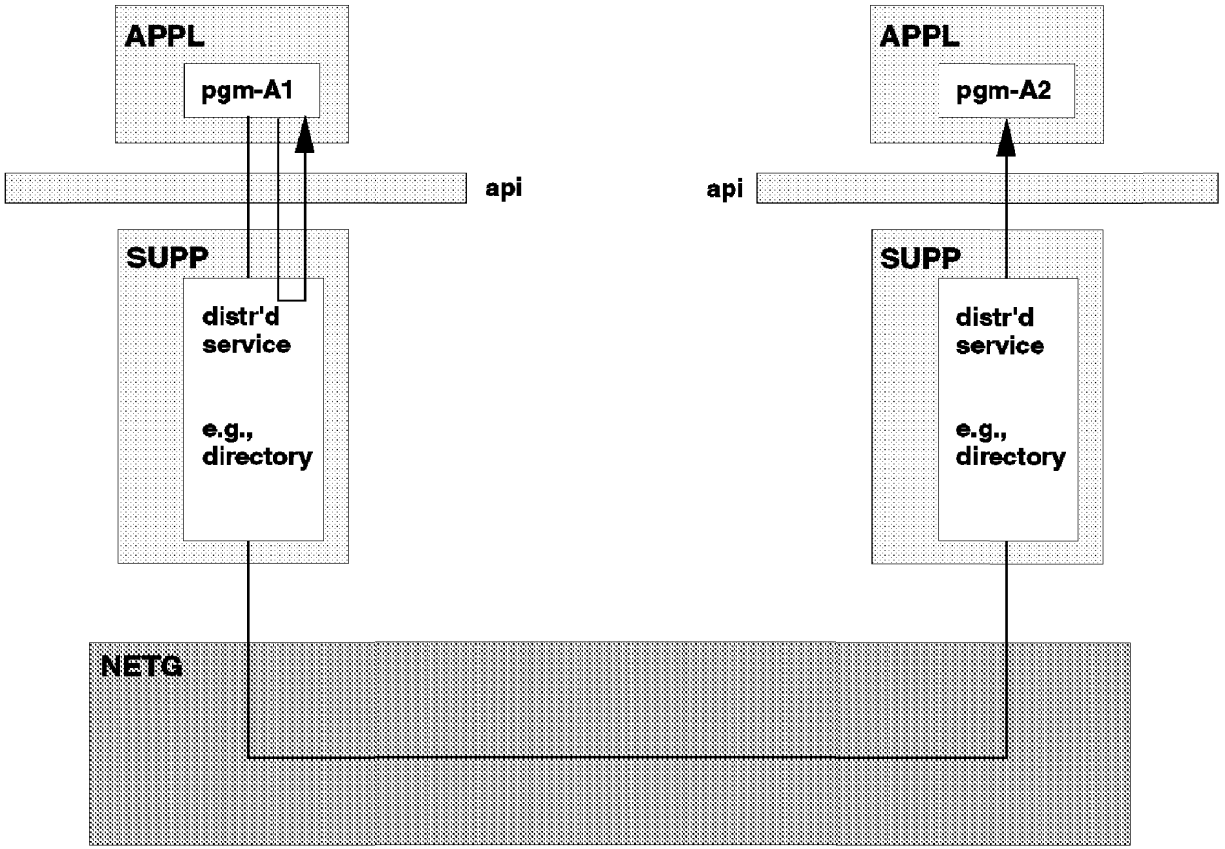

© 1993 IBM Corporation

# Standard Applications Support



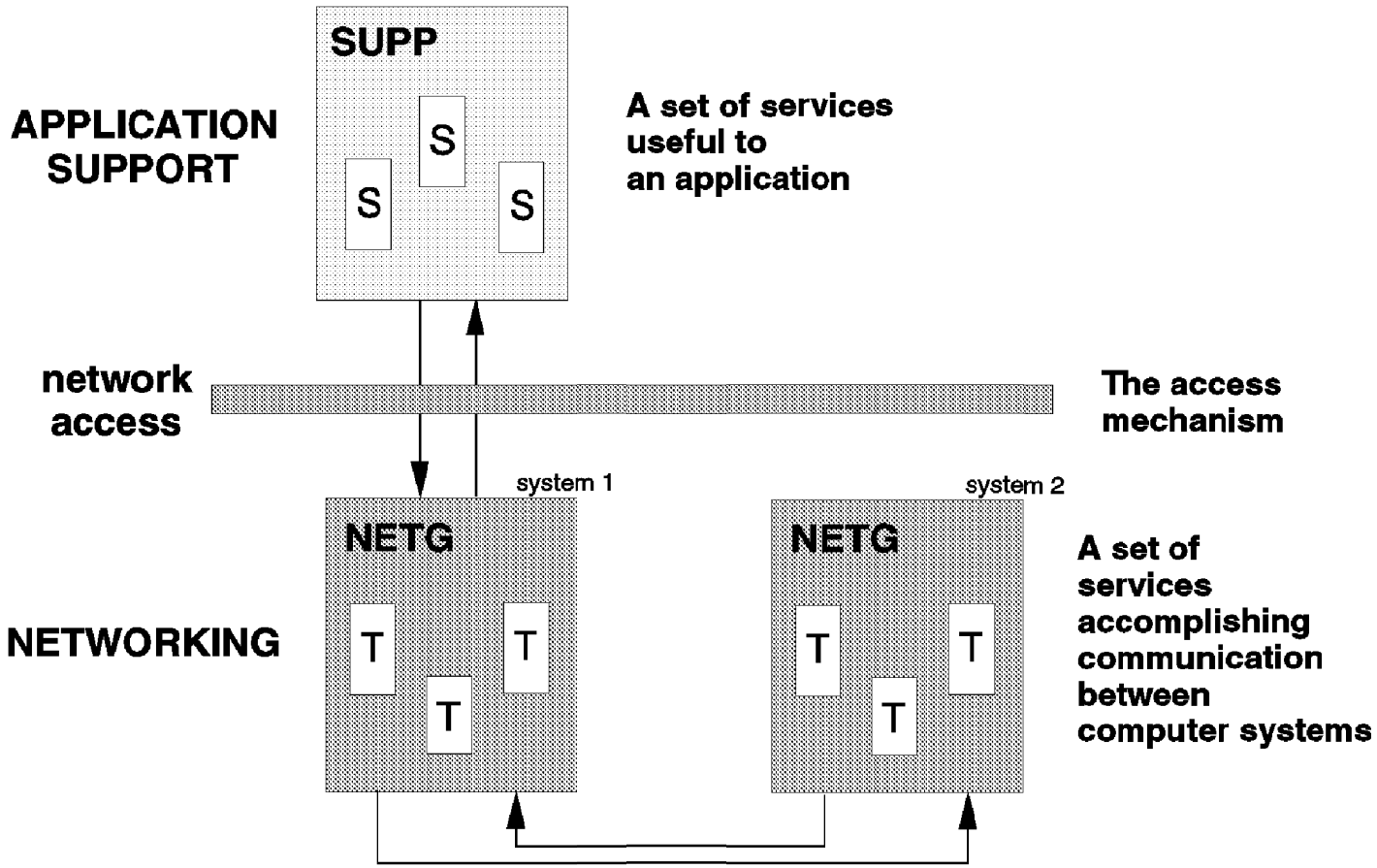

© 1993 IBM Corporation

# Distributed Services Support



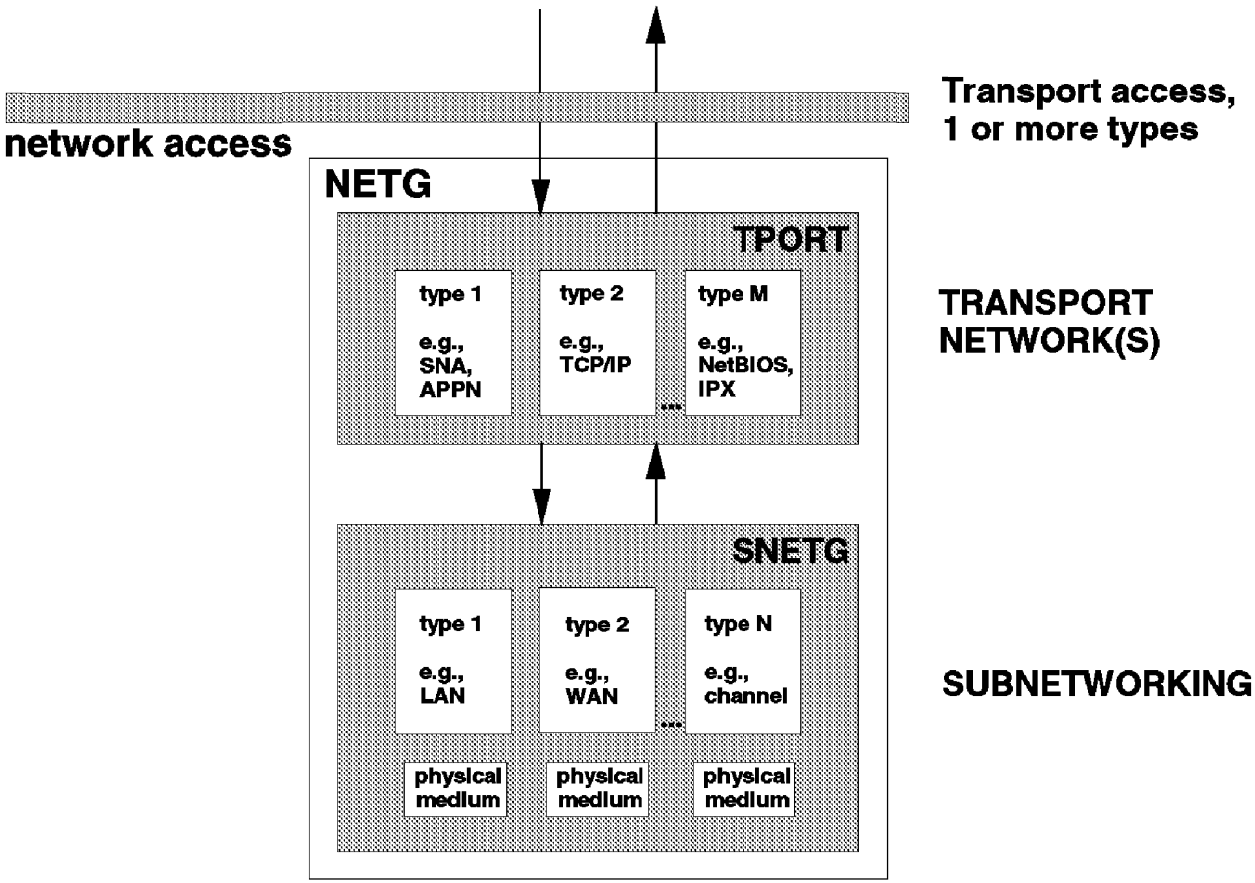

© 1993 IBM Corporation

# NETWORKING Definition



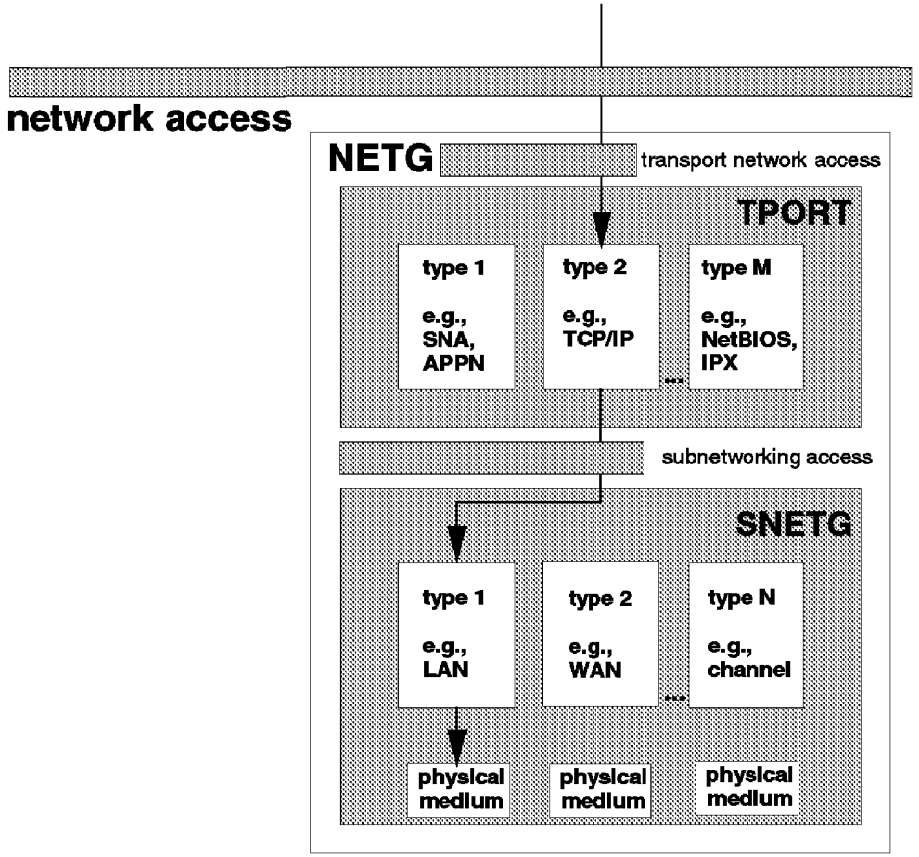

© 1993 IBM Corporation

# Networking Componentry




© 1993 IBM Corporation

# Networking Componentry Choices

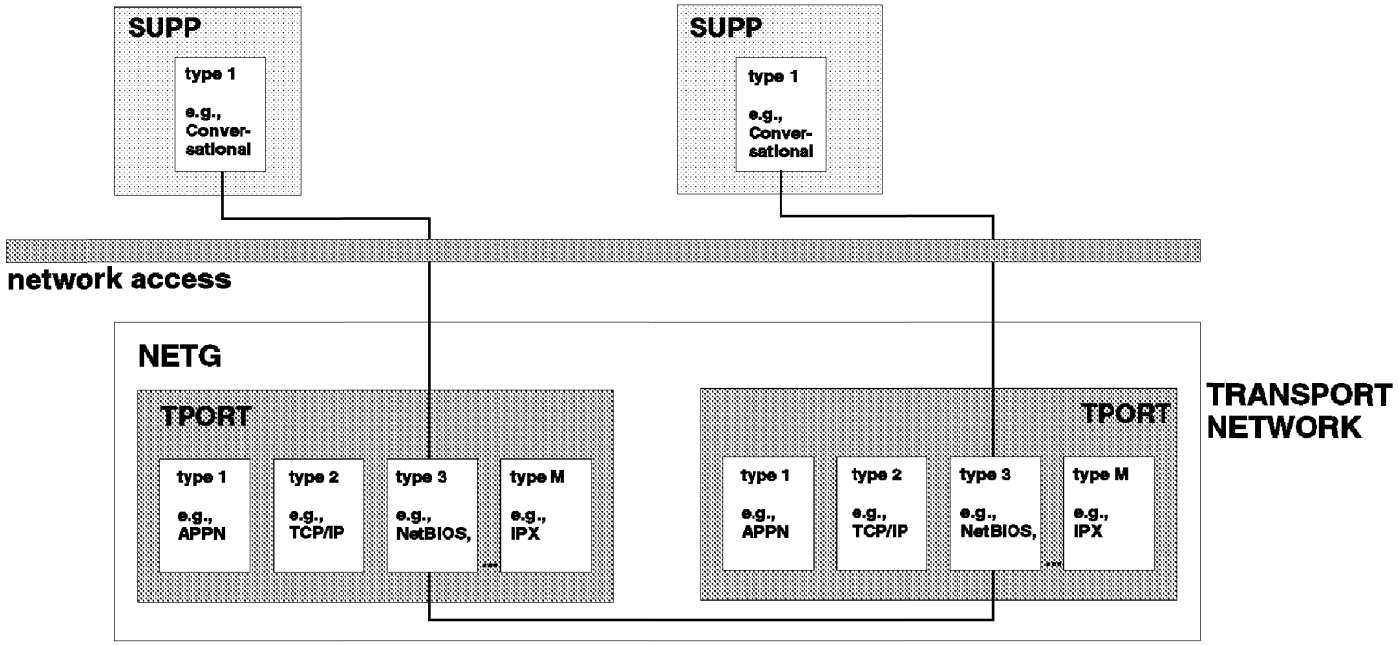



TRANSPORT NETWORK(S)

SUBNETWORKING

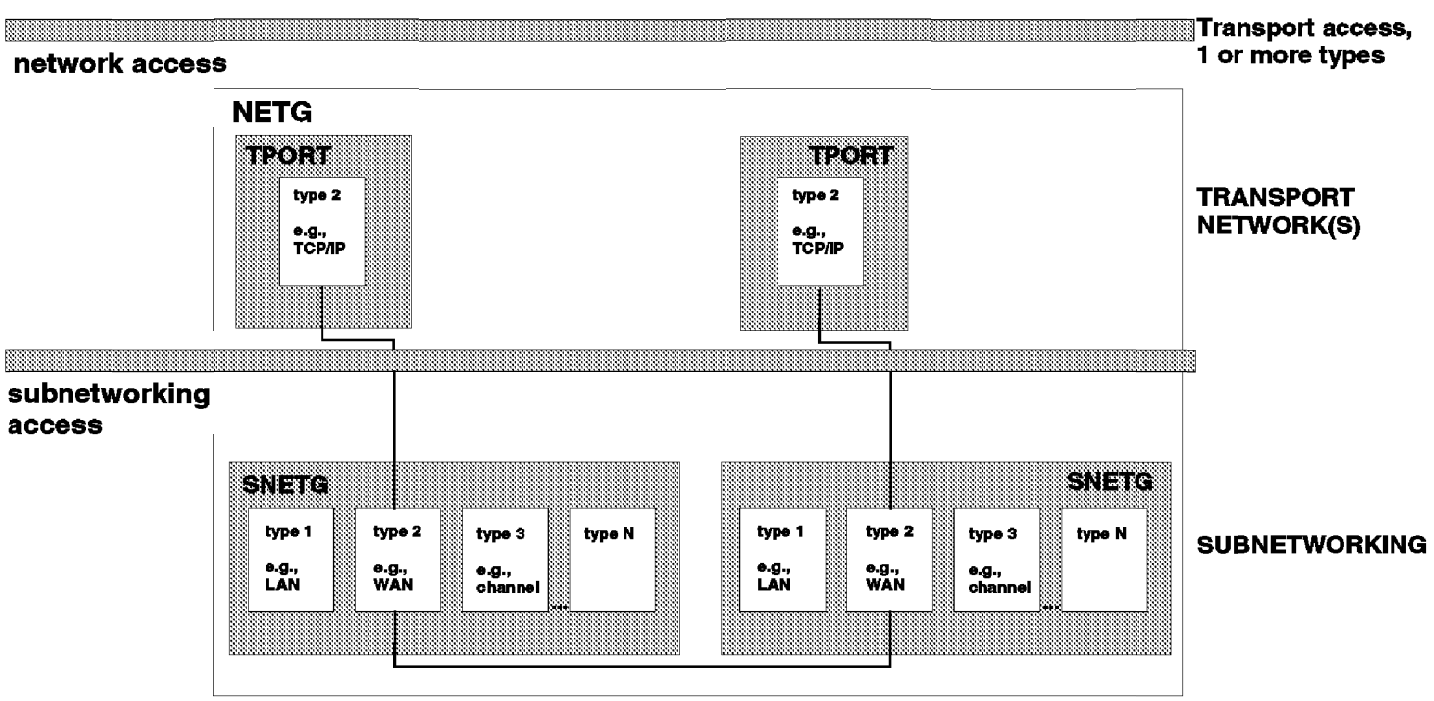
© 1993 IBM Corporation

# TRANSPORT NETWORK Definition



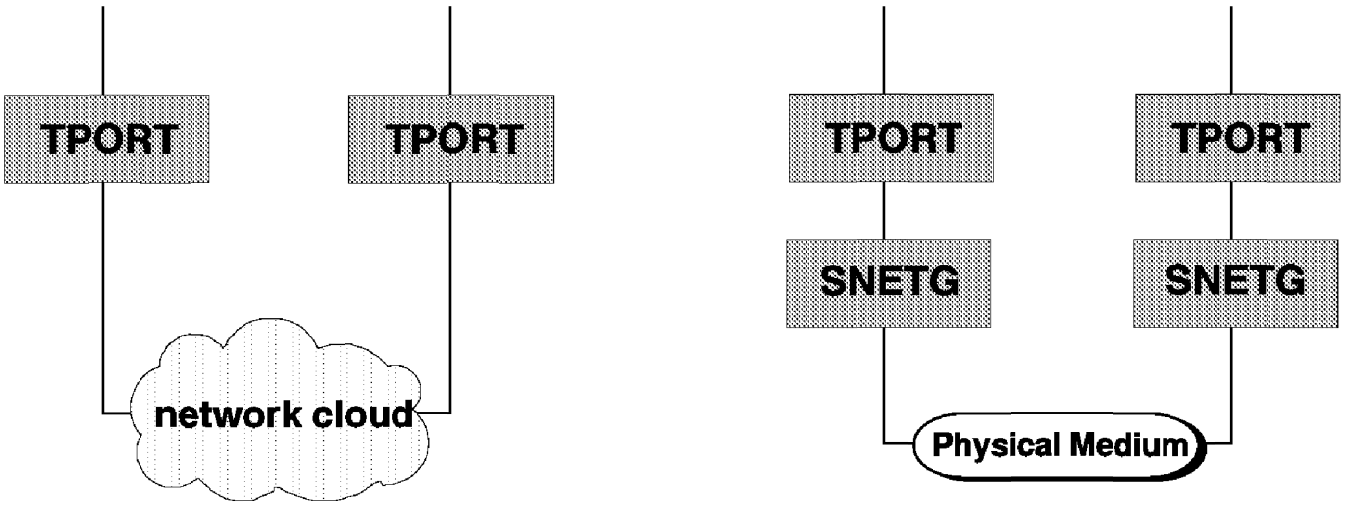

© 1993 IBM Corporation

# SUBNETWORKING Definition



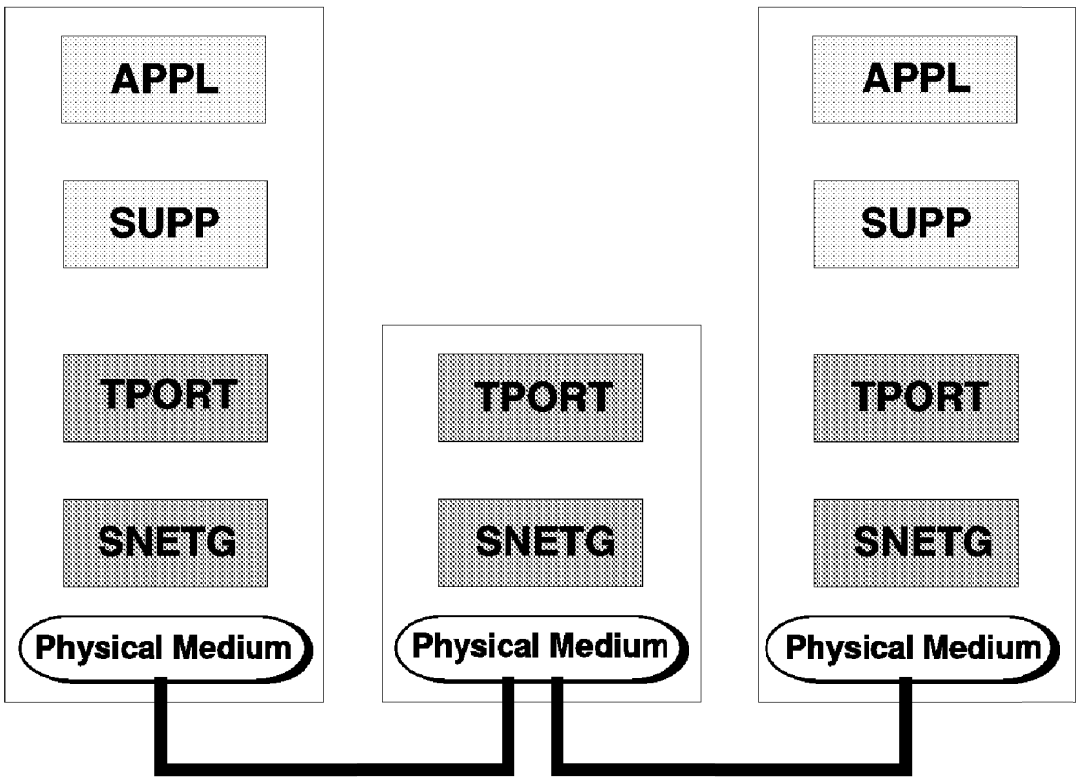

© 1993 IBM Corporation

# Network Cloud and Subnetworking

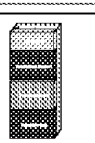



© 1993 IBM Corporation

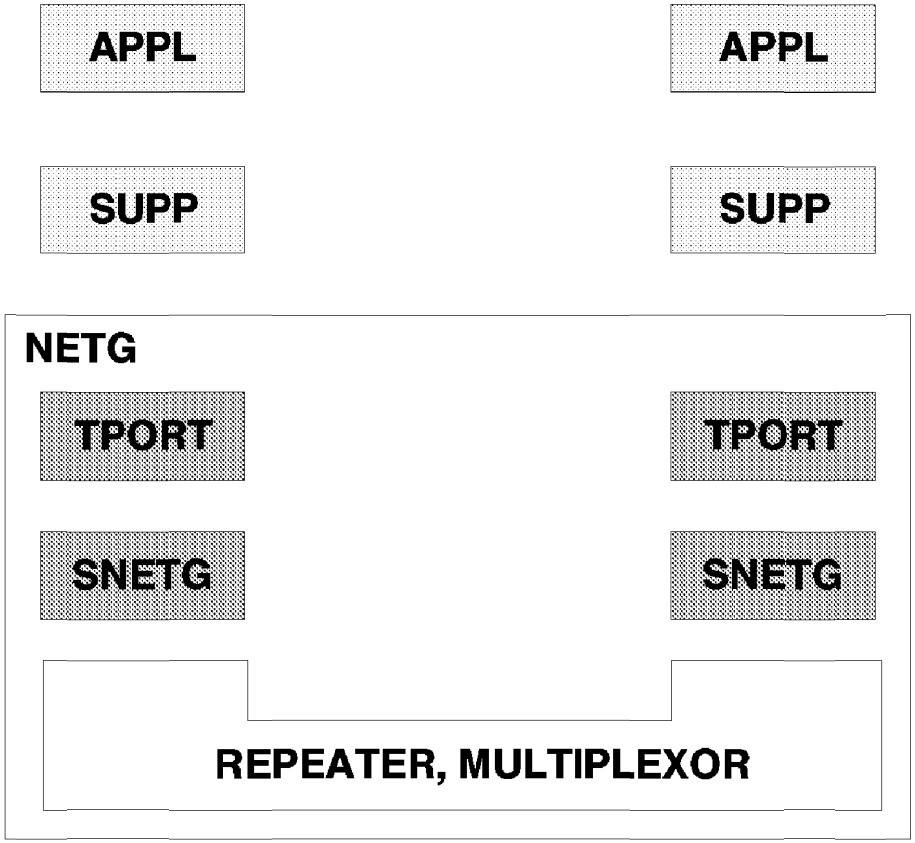
# Physical Media and "Short" Stacks




© 1993 IBM Corporation

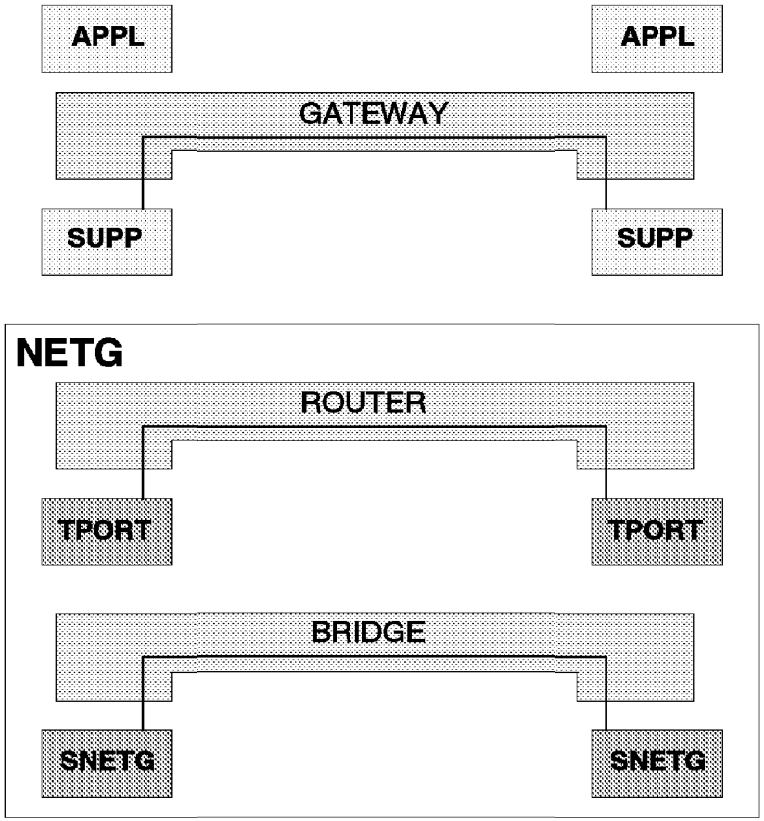


# Repeaters and Multiplexors

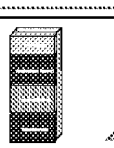



© 1993 IBM Corporation

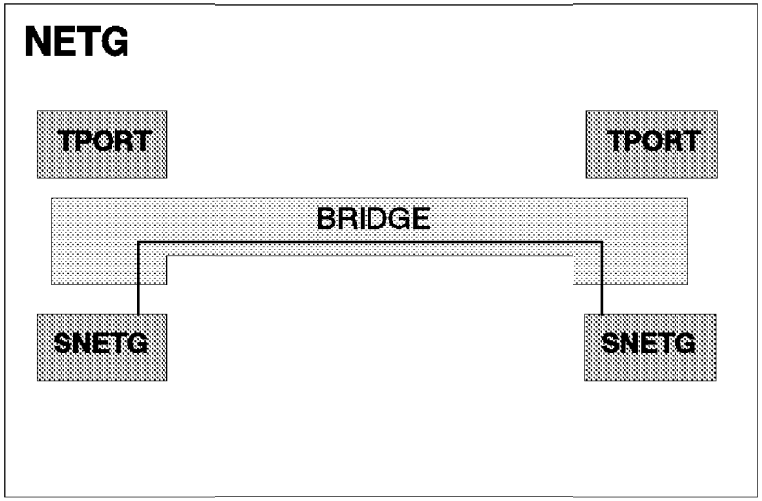
# Bridges, Routers, and Gateways




© 1993 IBM Corporation

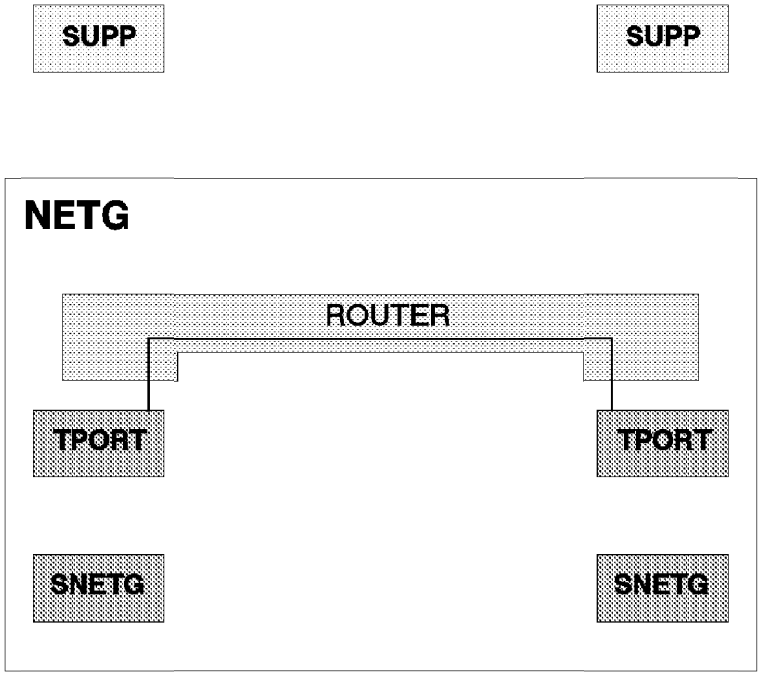


# Bridges



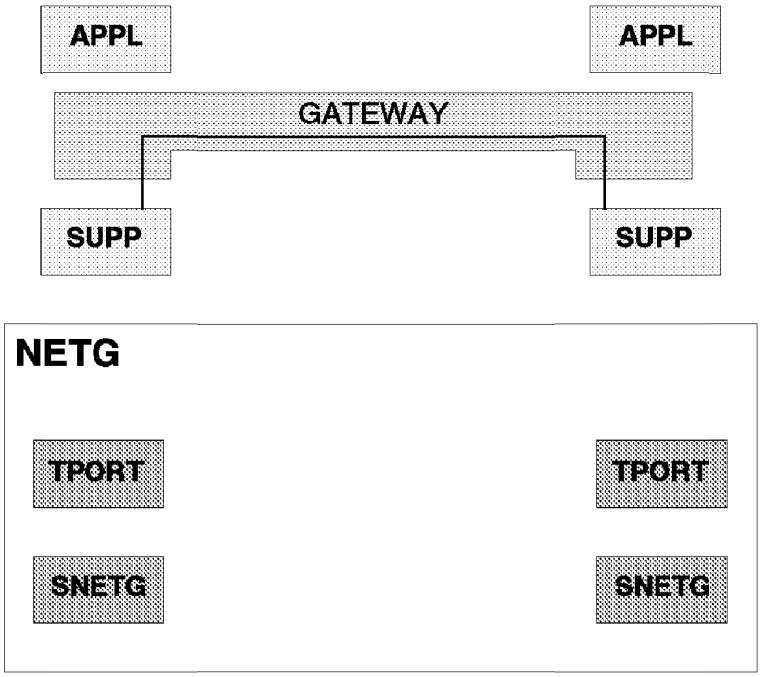

© 1993 IBM Corporation

# Routers

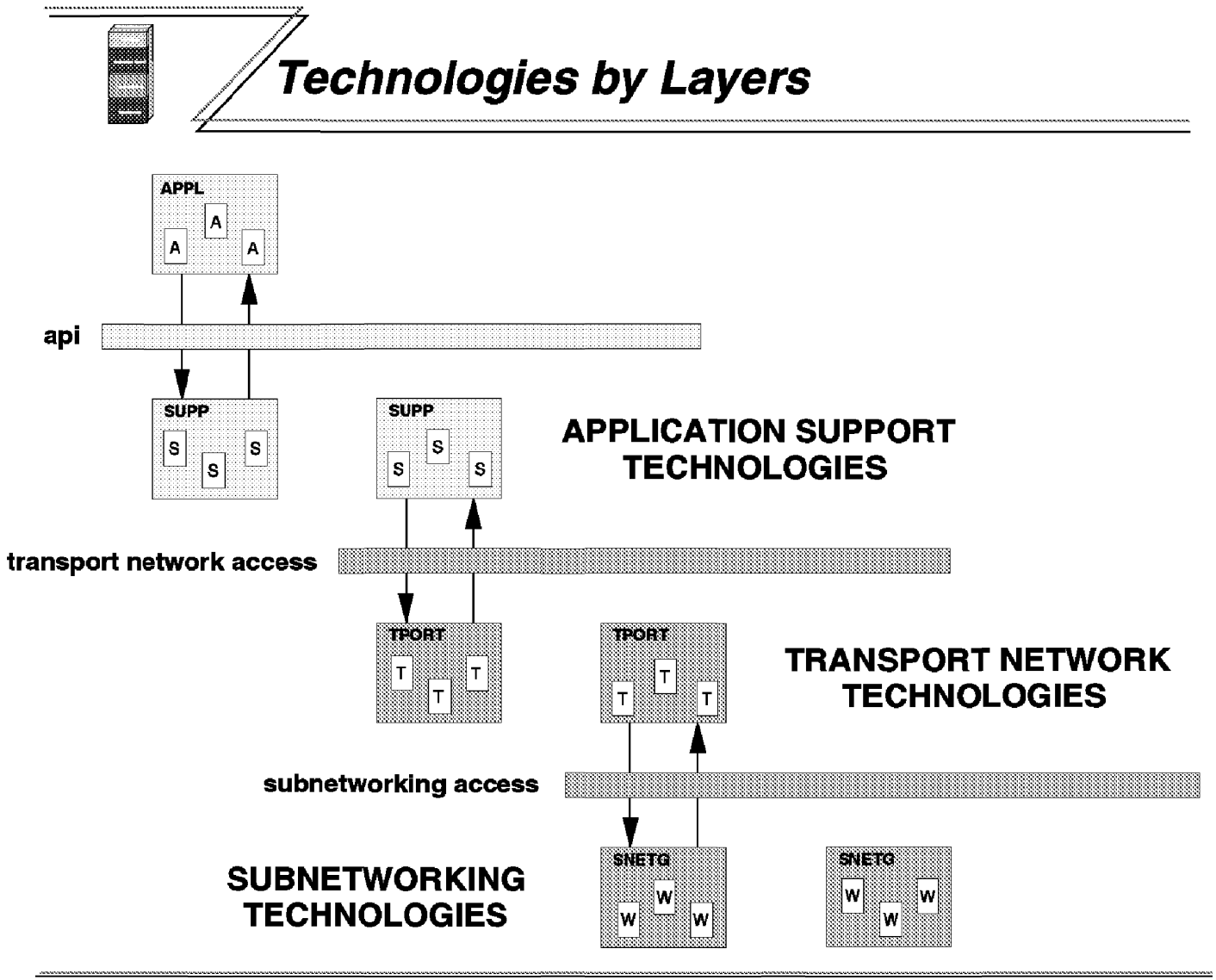



© 1993 IBM Corporation

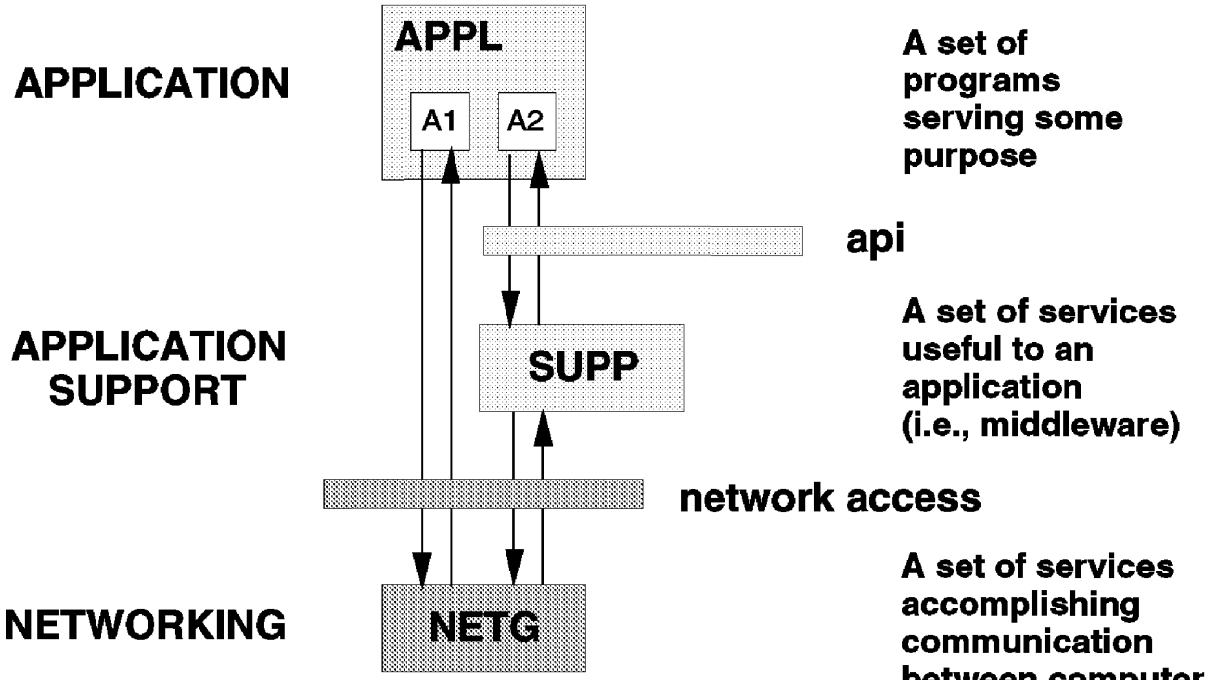
# Gateways




© 1993 IBM Corporation

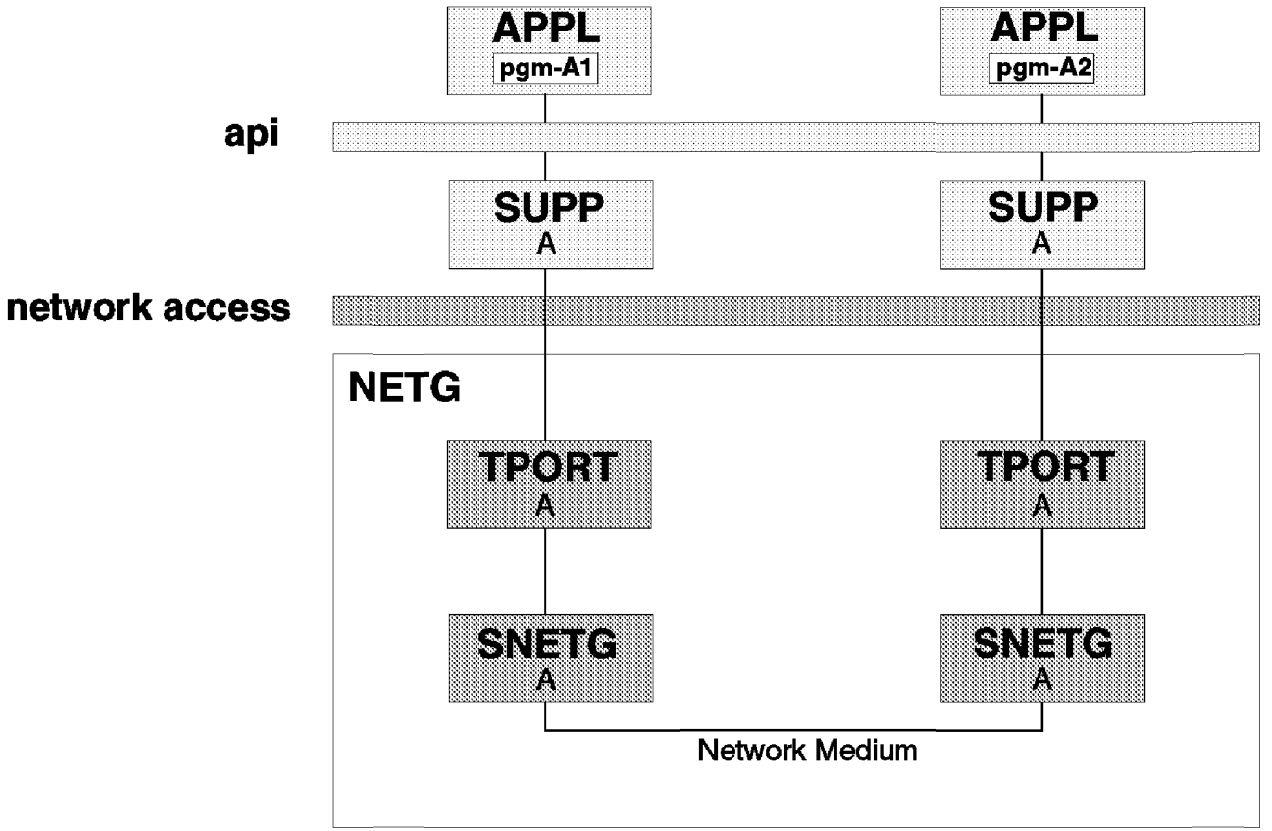

Technologies by Layers
APPL
A
A
A
api
SUPP
S
S
S
SUPP
S
S
S
APPLICATION SUPPORT TECHNOLOGIES
transport network access
TPORT
T
T
T
TPORT
T
T
T
TRANSPORT NETWORK TECHNOLOGIES
subnetworking access
SNETG
W
W
W
SNETG
W
W
W
SUBNETWORKING TECHNOLOGIES
© 1993 IBM Corporation

# Direct and Indirect Networking

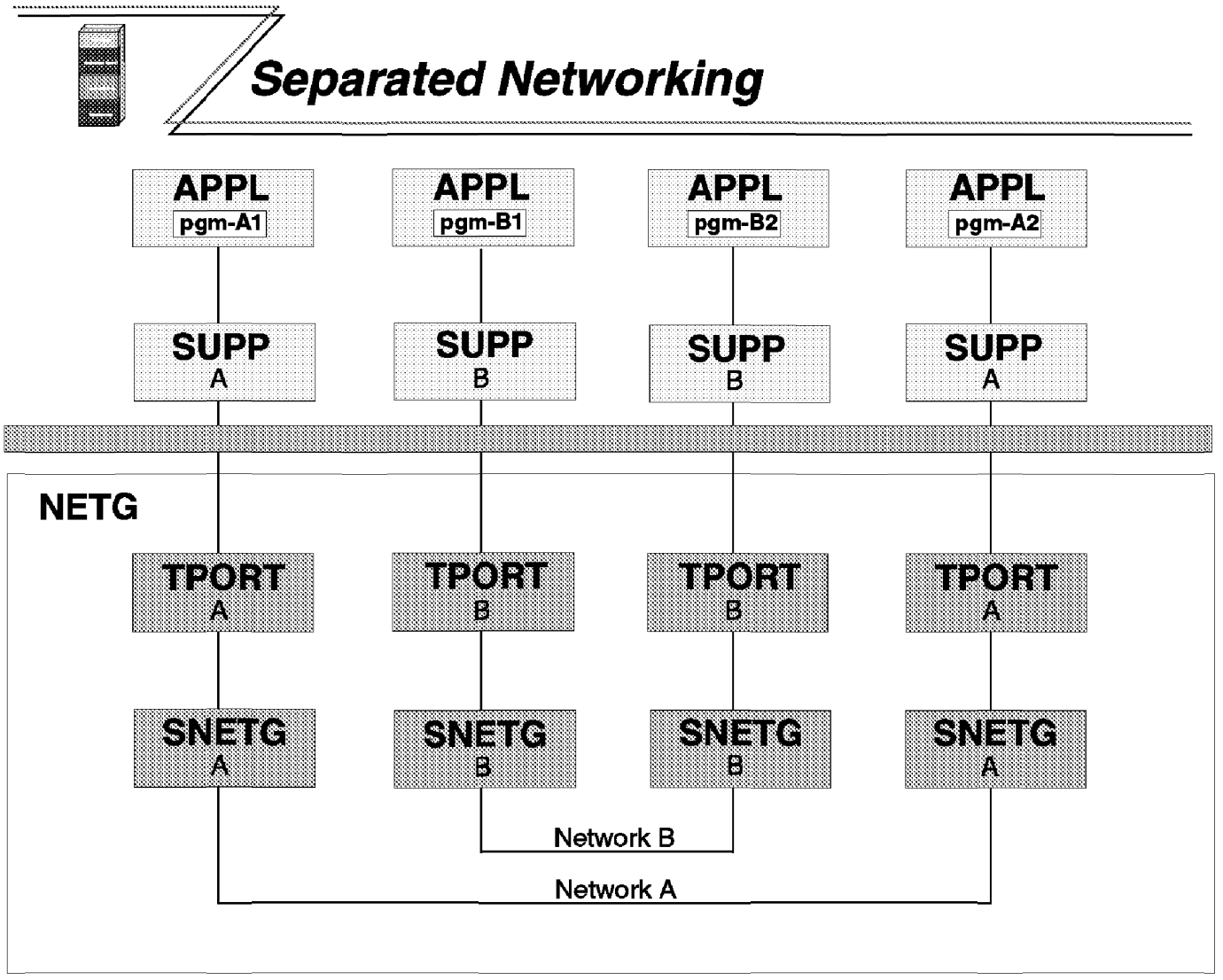



© 1993 IBM Corporation

# Simple Networking

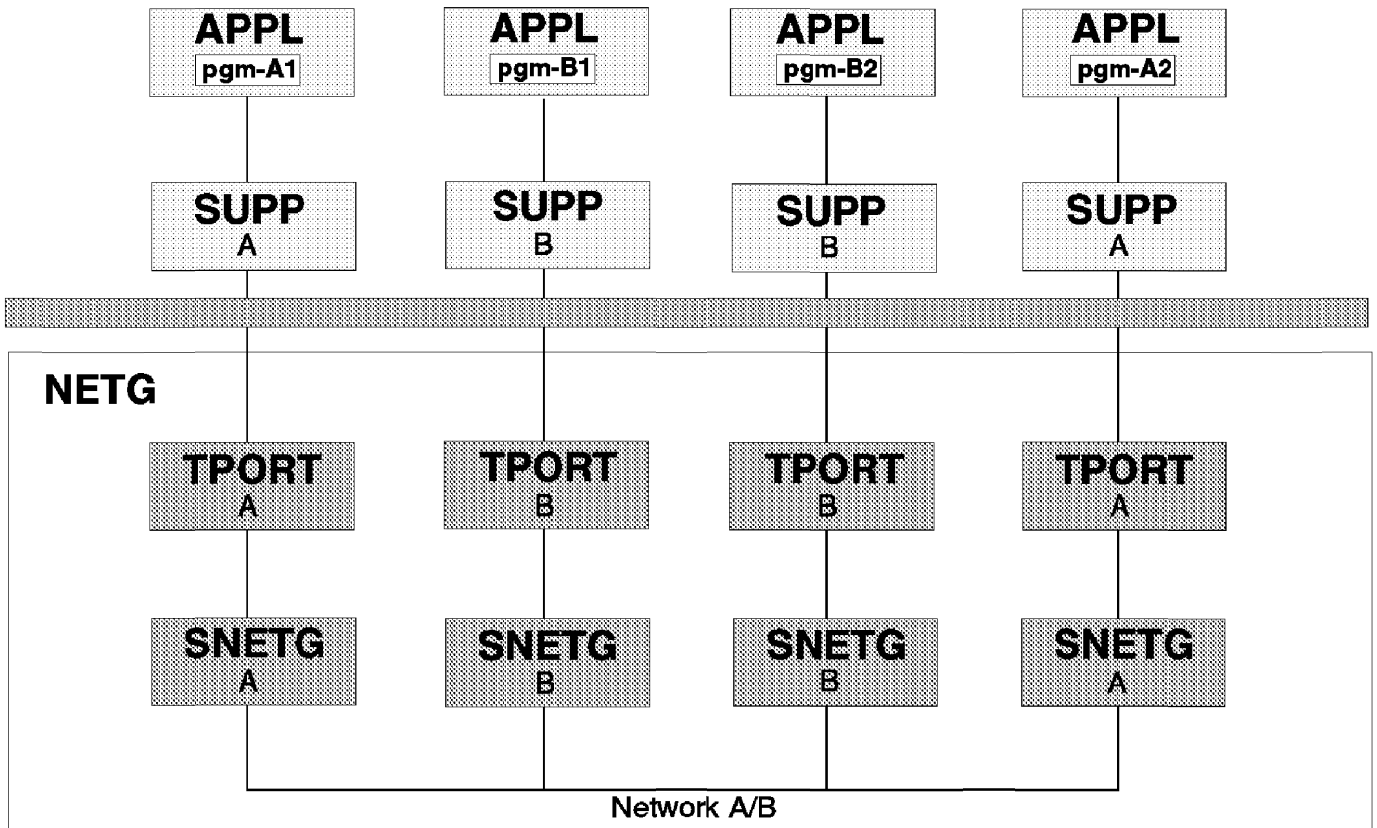



© 1993 IBM Corporation



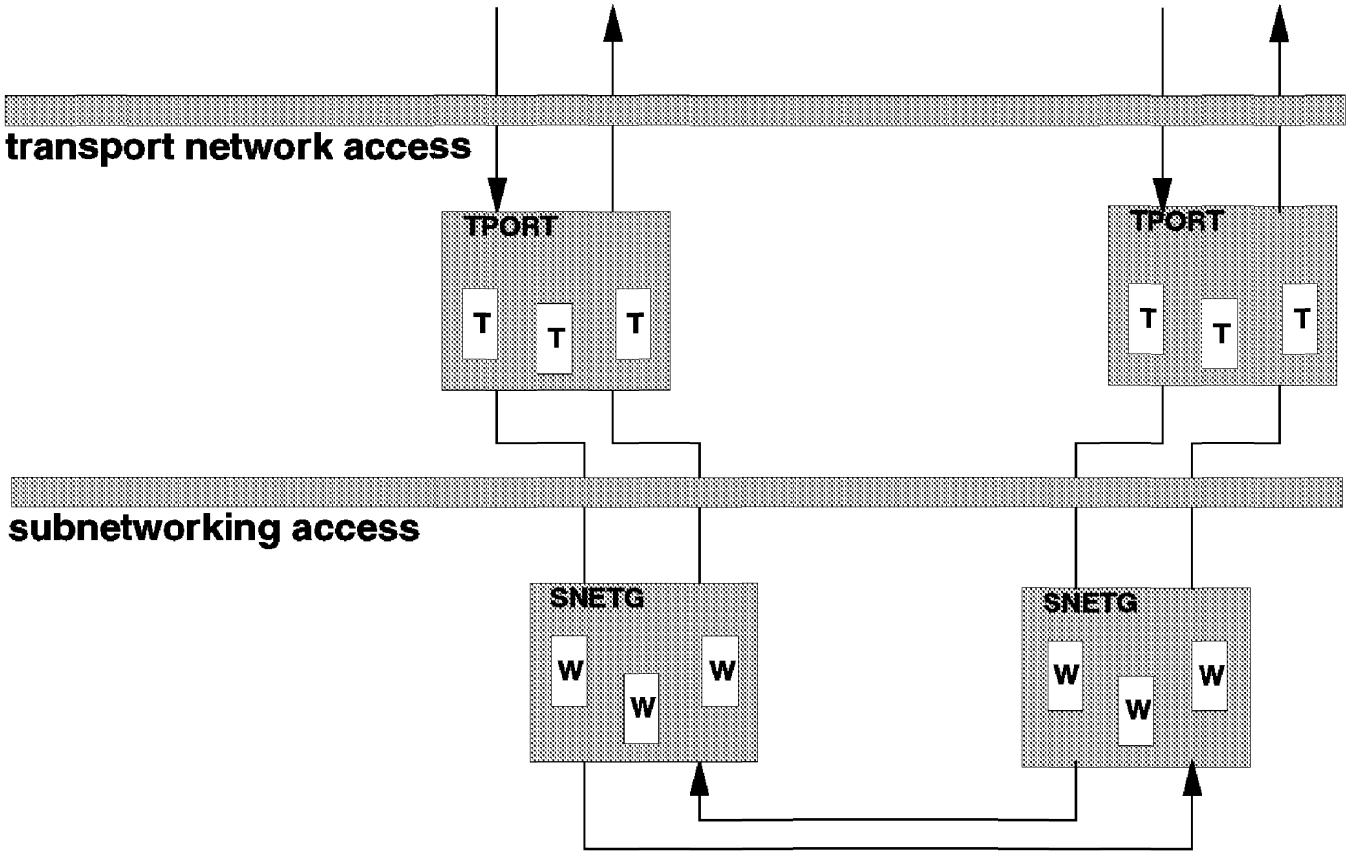

© 1993 IBM Corporation

# Shared Networking



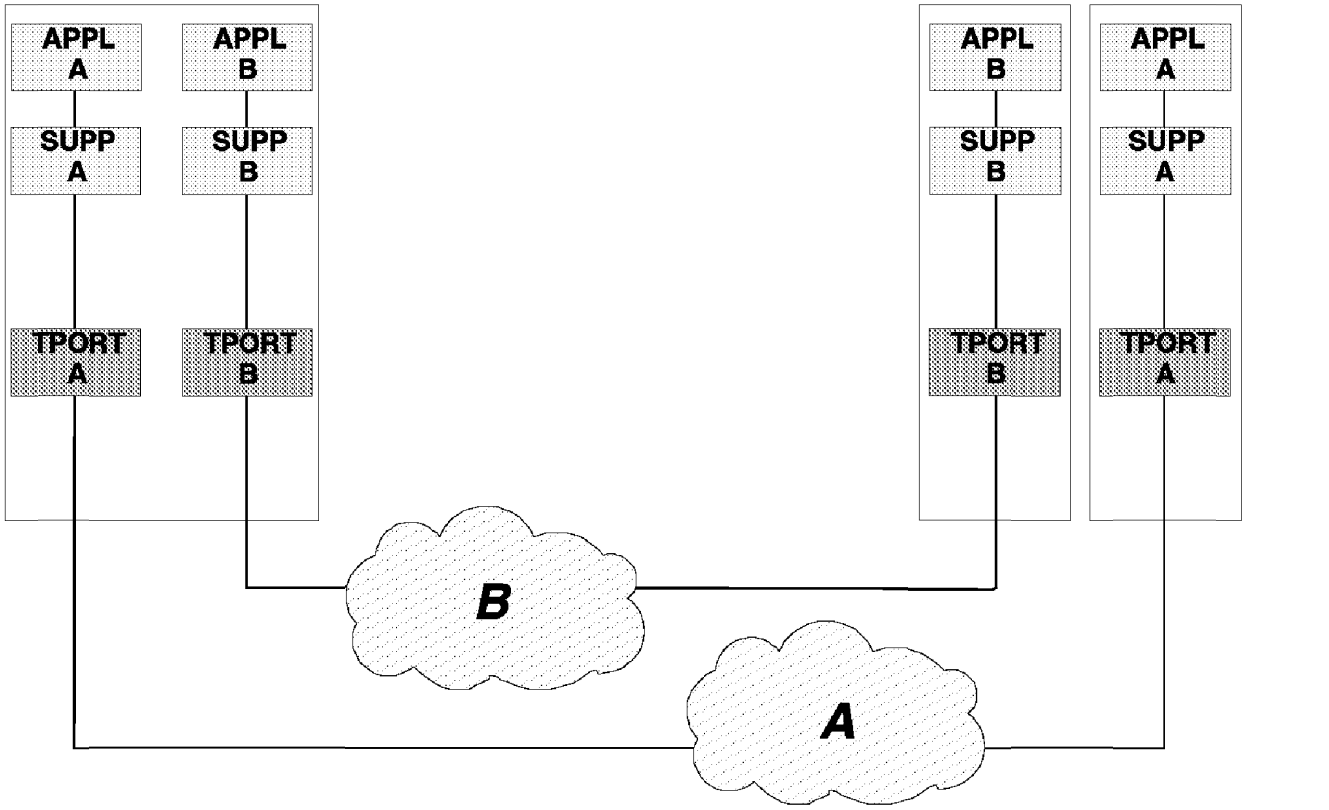

© 1993 IBM Corporation

# Subnetworking Technologies



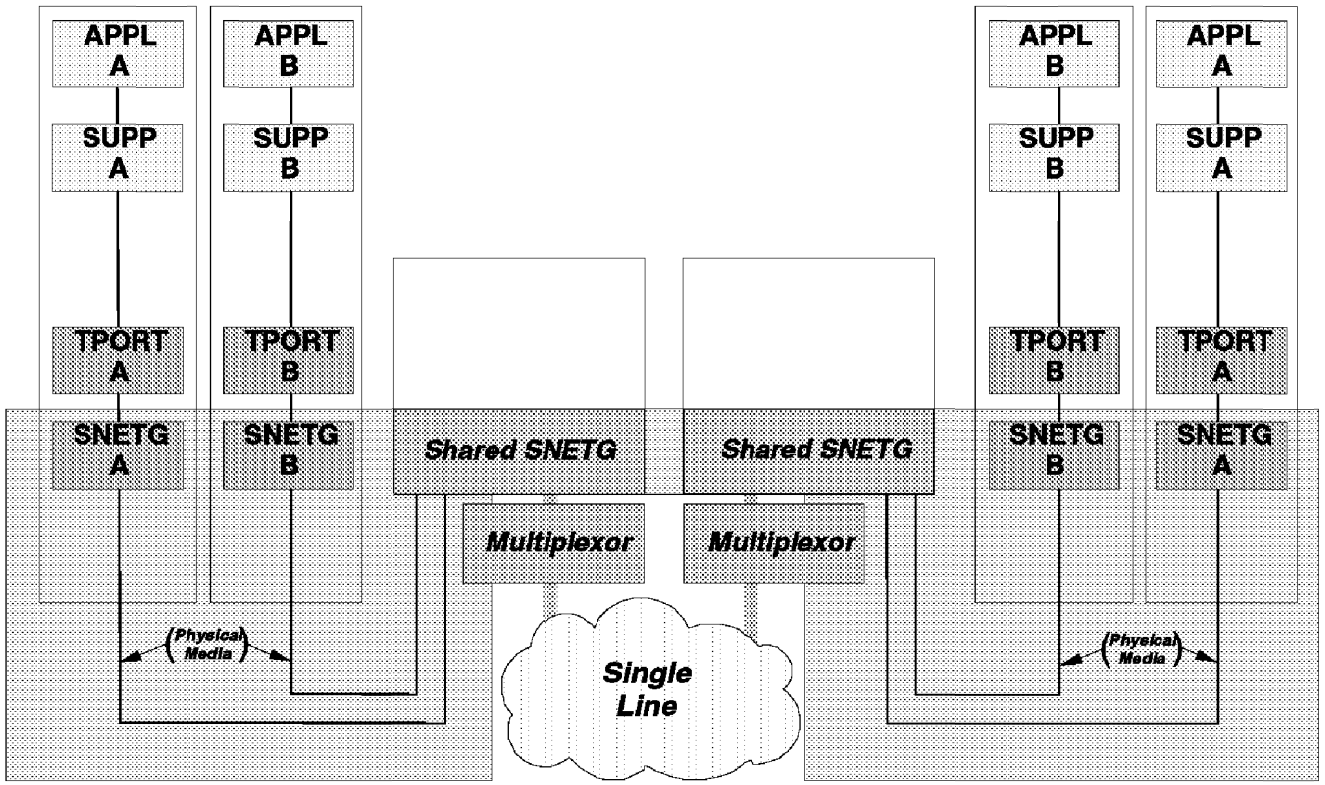

© 1993 IBM Corporation

# Separate Wide Area Networks (WANs)



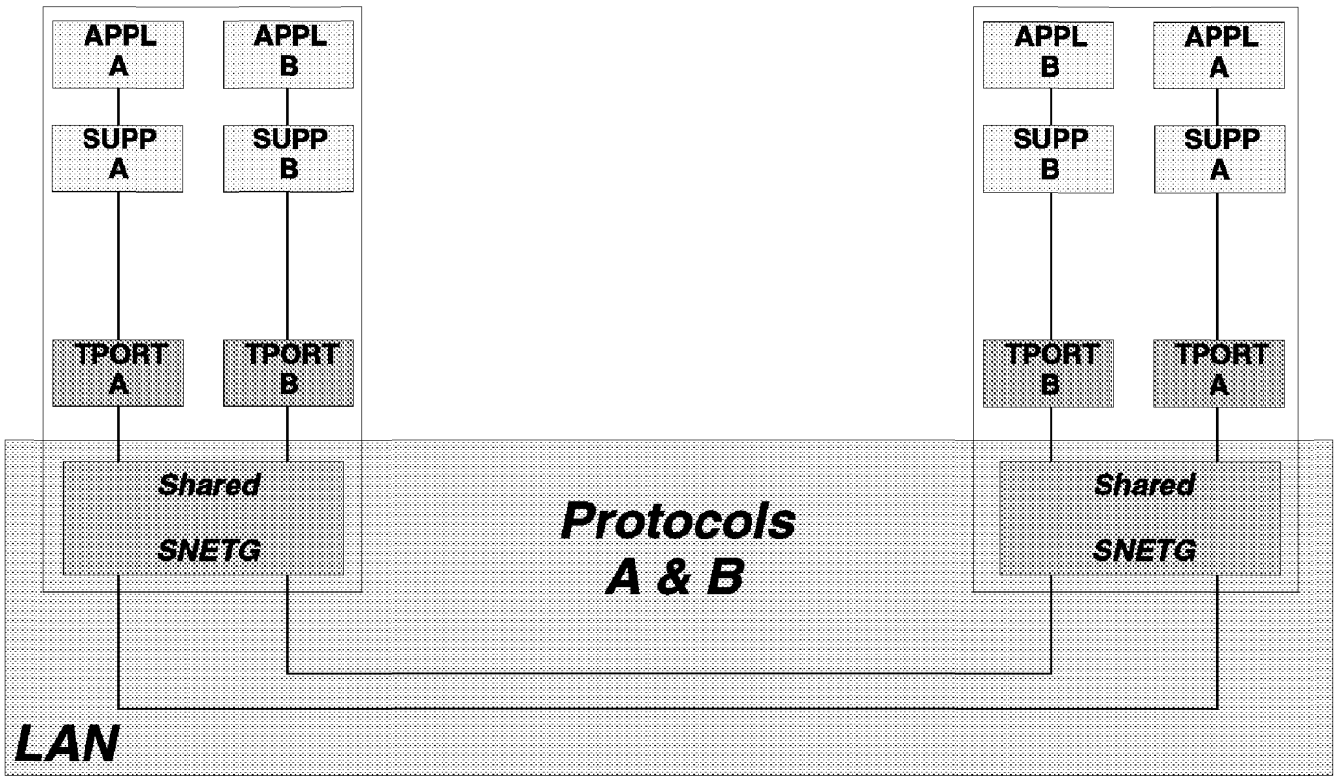

© 1993 IBM Corporation

# Multiplexor (Bandwidth Management)



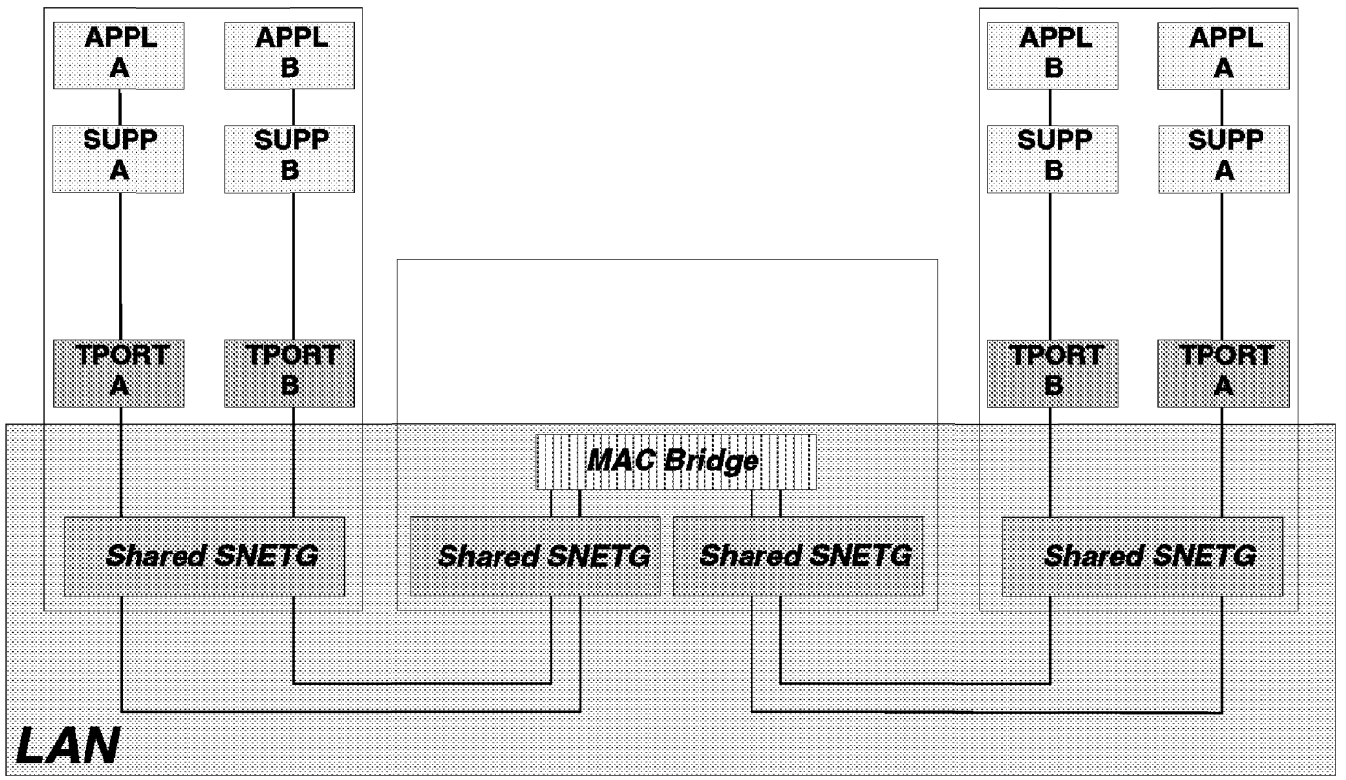

© 1993 IBM Corporation

# Shared Subnetworking: LANs



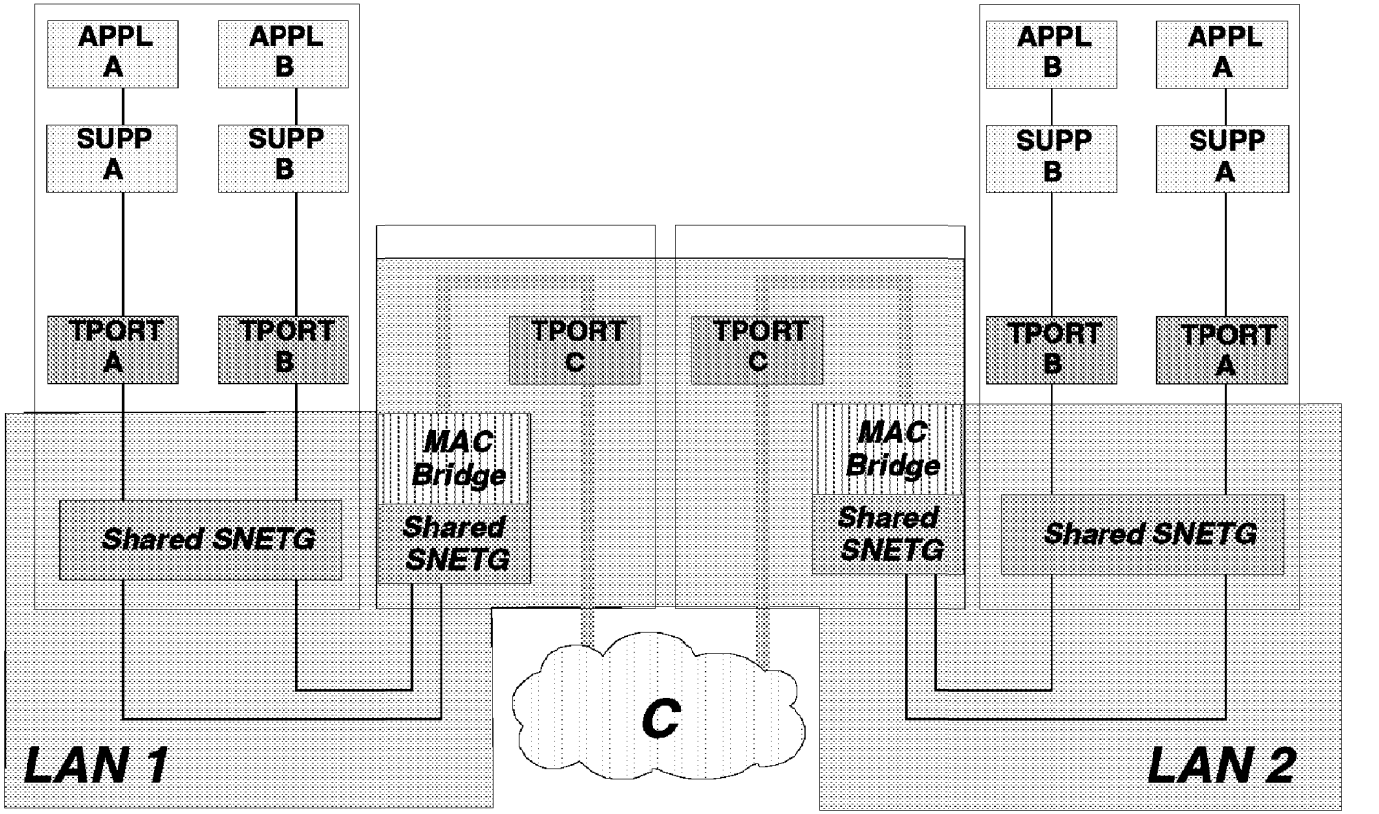

© 1993 IBM Corporation

# Local Bridge



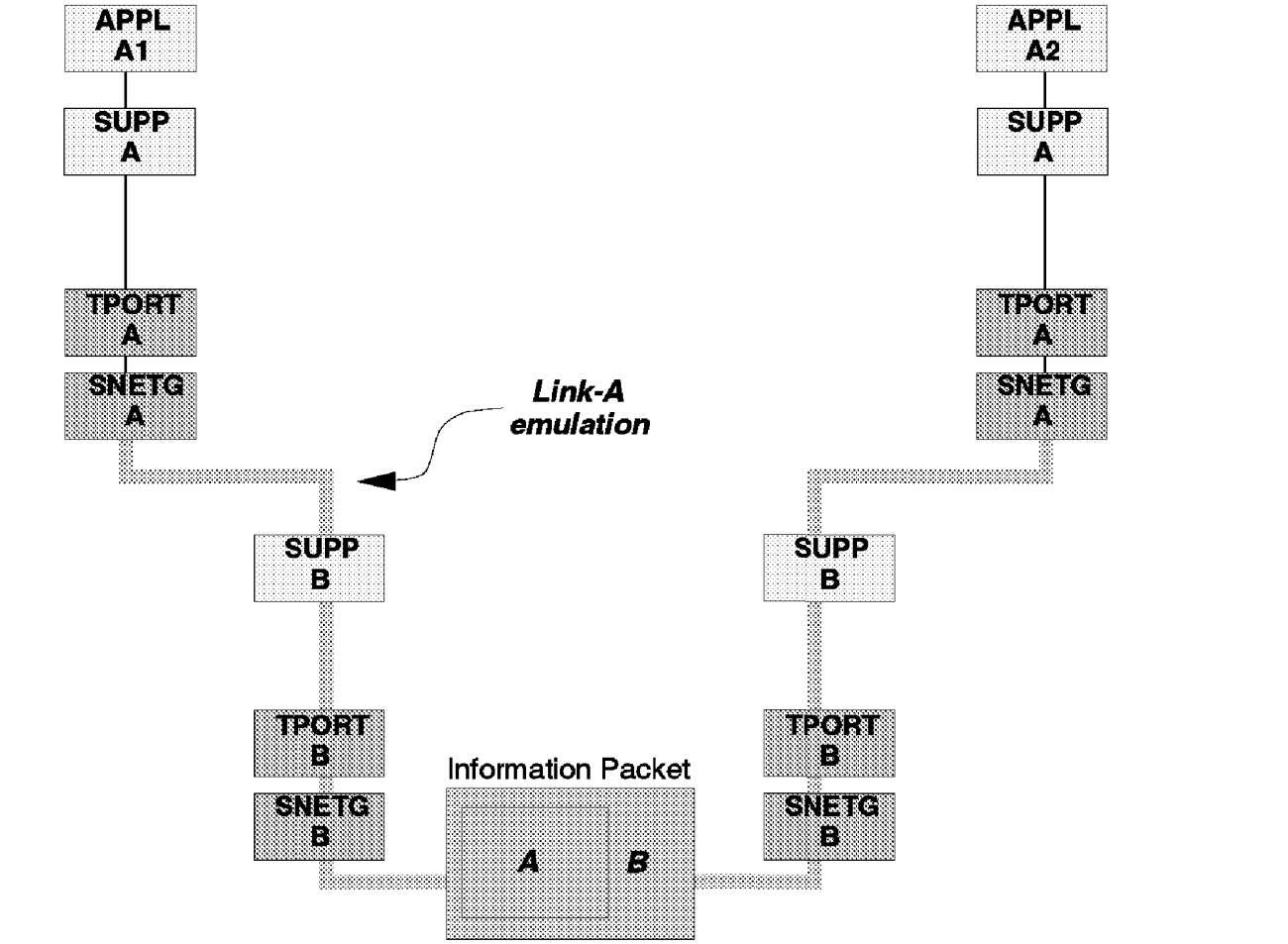

© 1993 IBM Corporation

# Remote (Split) Bridge



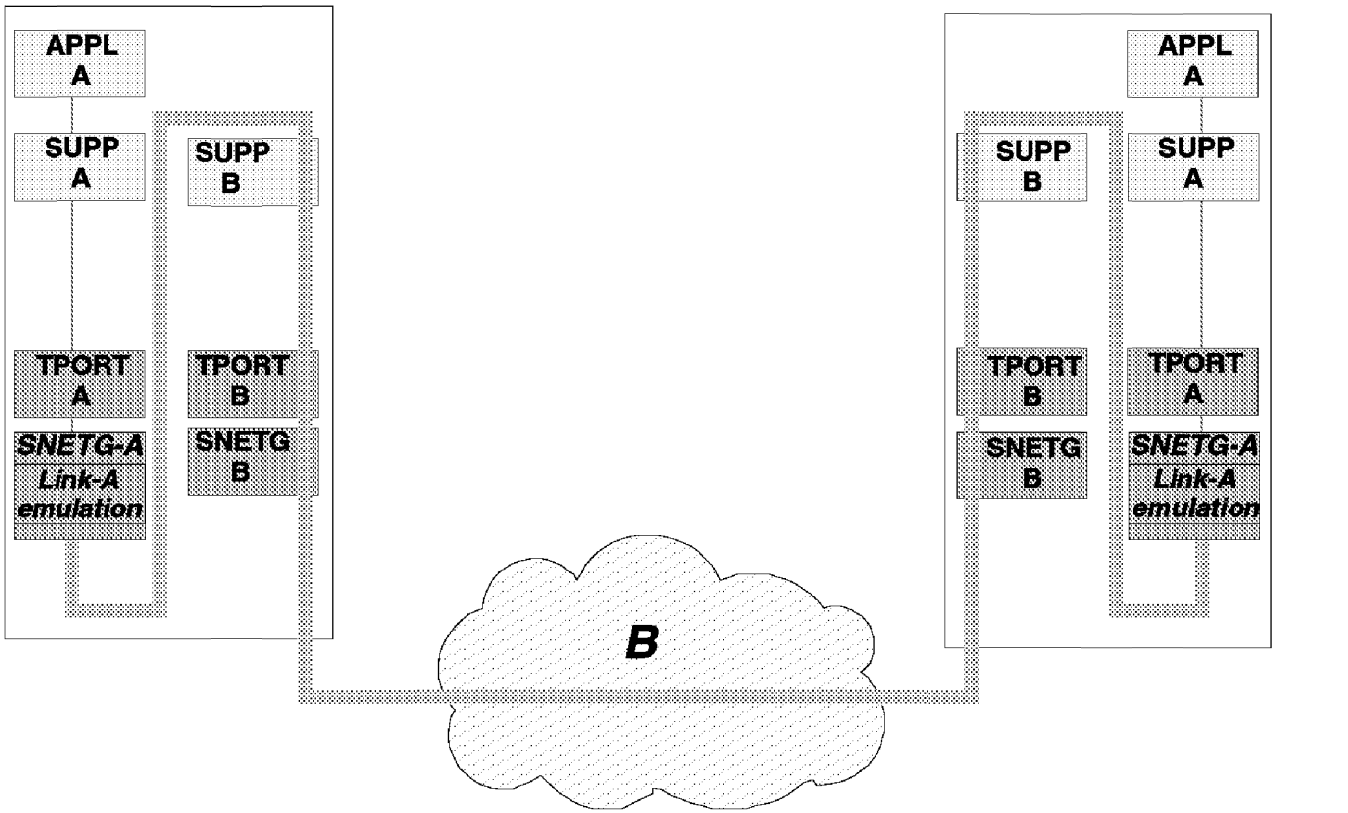

© 1993 IBM Corporation

# General Encapsulation Technique



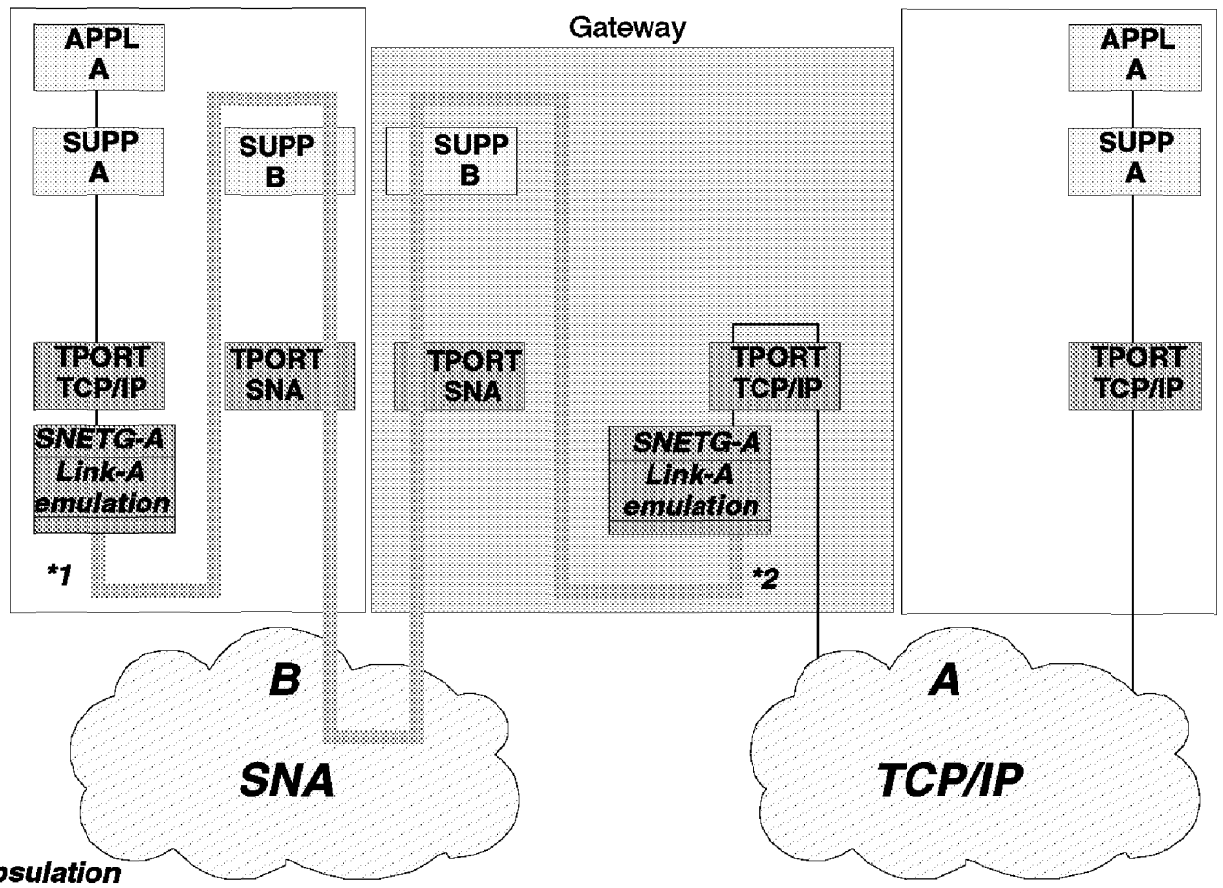

© 1993 IBM Corporation

# Encapsulation (End-to-End)




© 1993 IBM Corporation

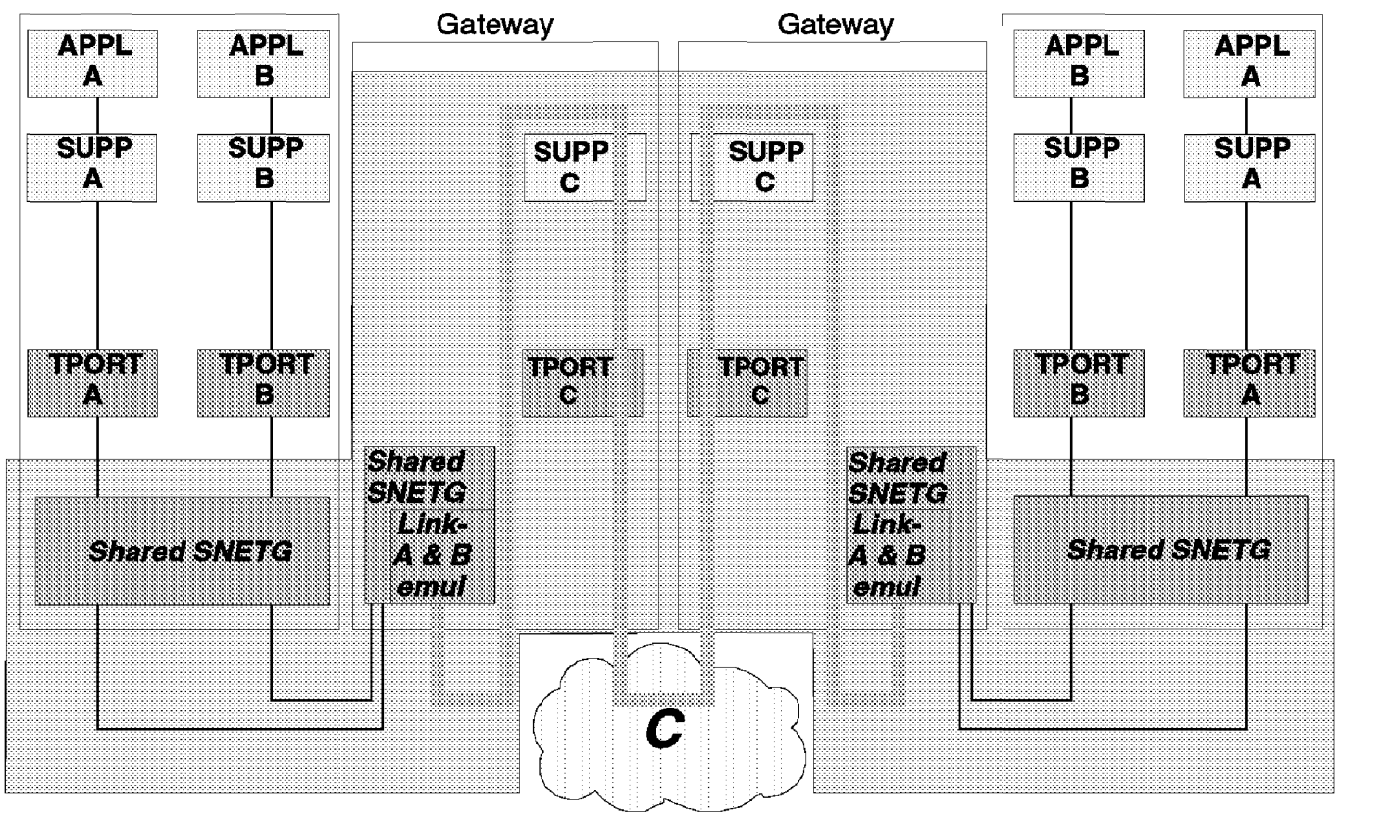
# Encapsulation at the Gateway




*1 Encapsulation
*2 De-Encapsulation

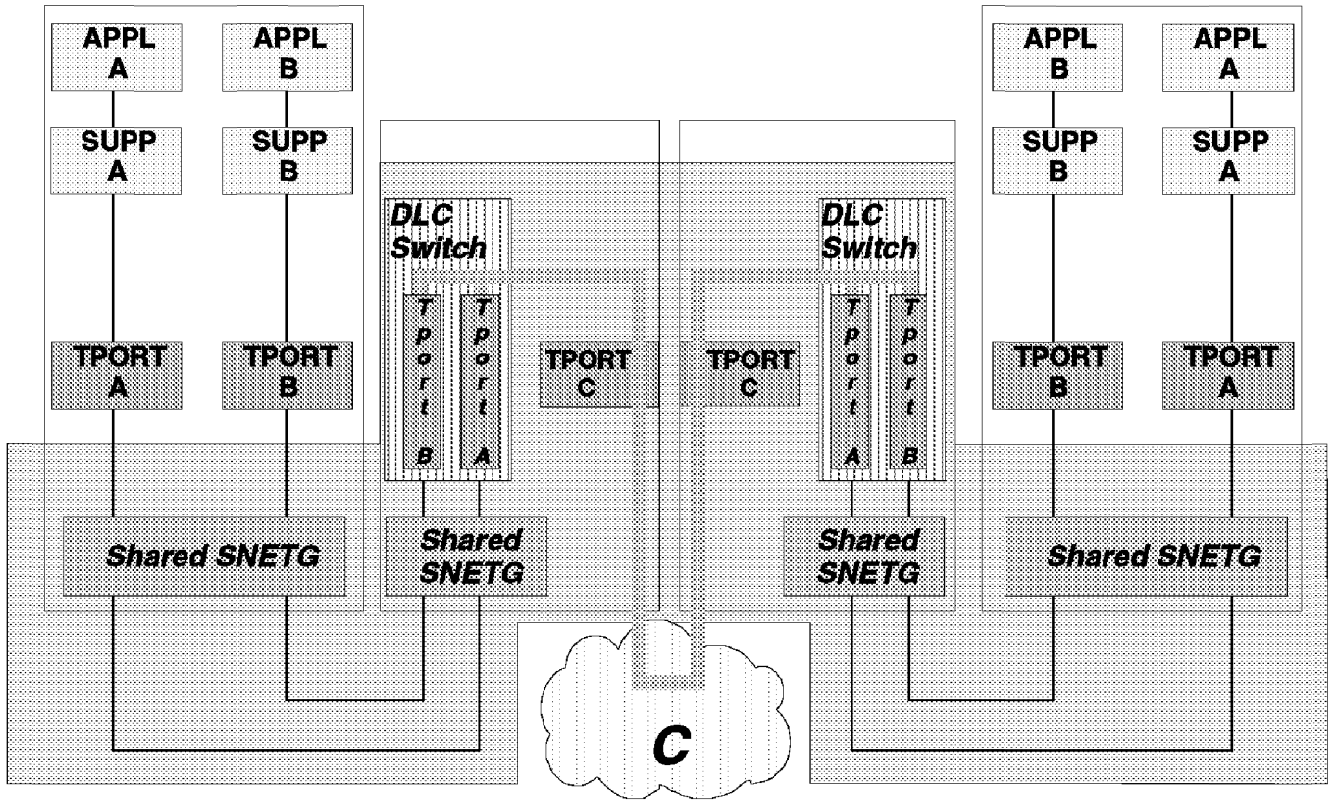
© 1993 IBM Corporation

# Double-Gateway Encapsulation



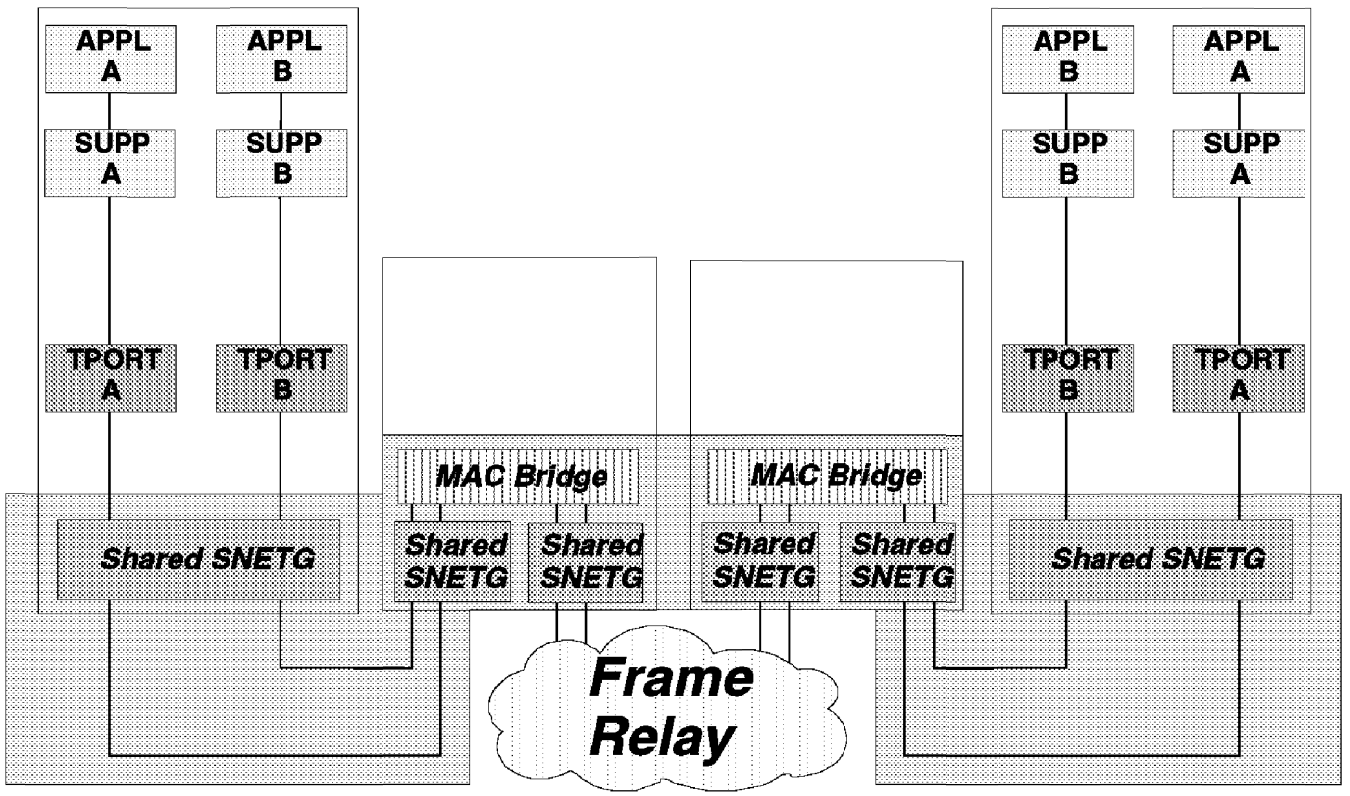

© 1993 IBM Corporation

# Data Link Switching (DLS)



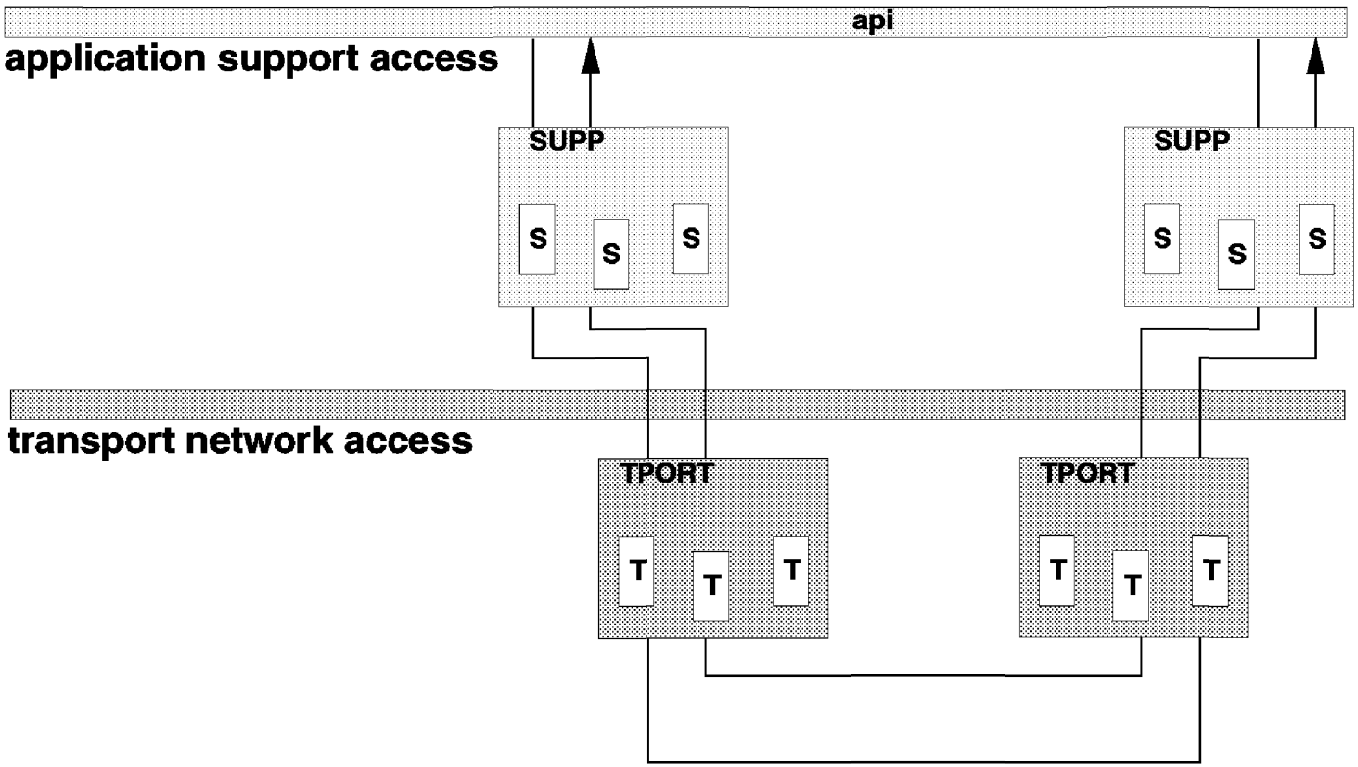

© 1993 IBM Corporation

# Frame Relay



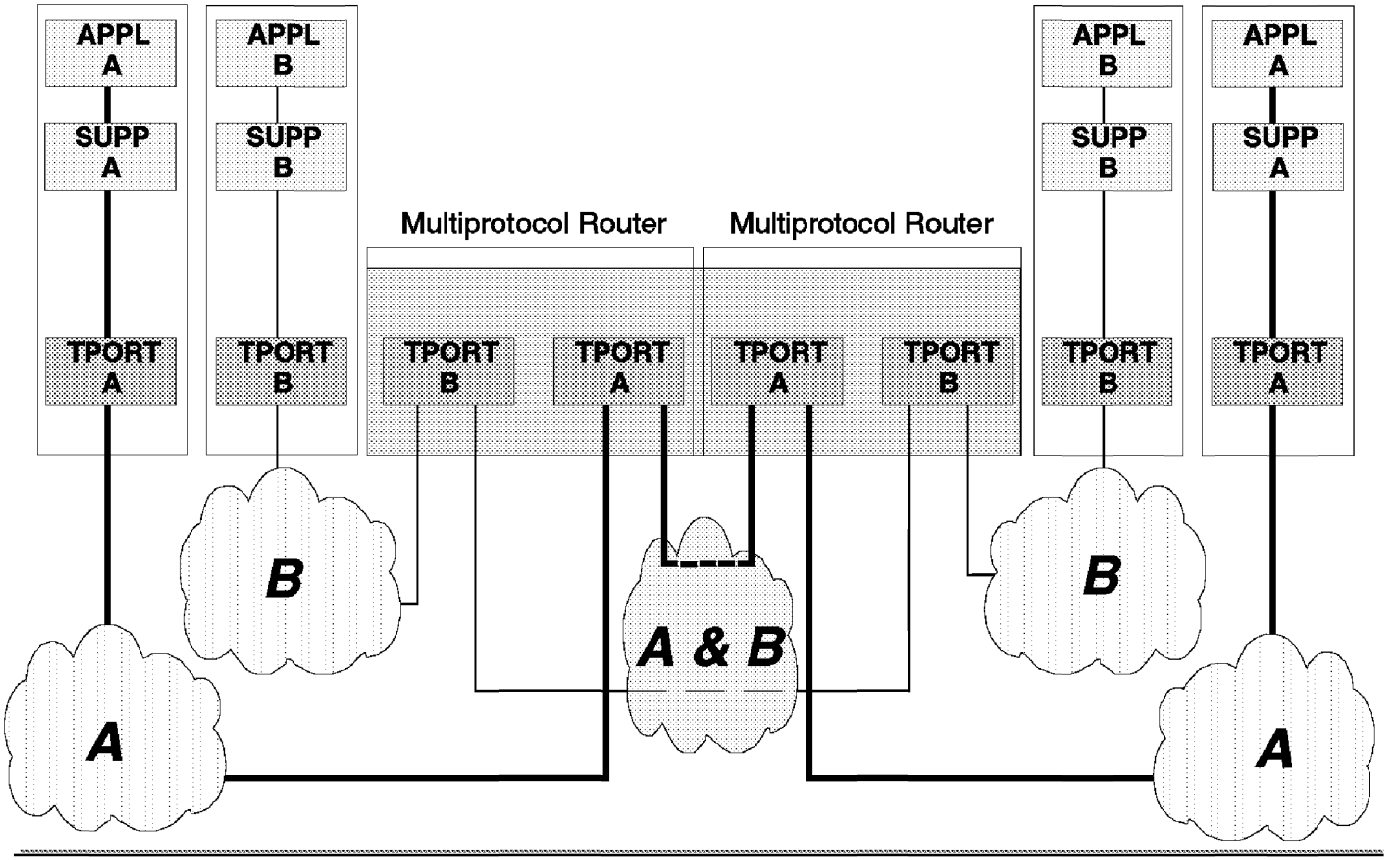

© 1993 IBM Corporation

# Transport Network Technologies



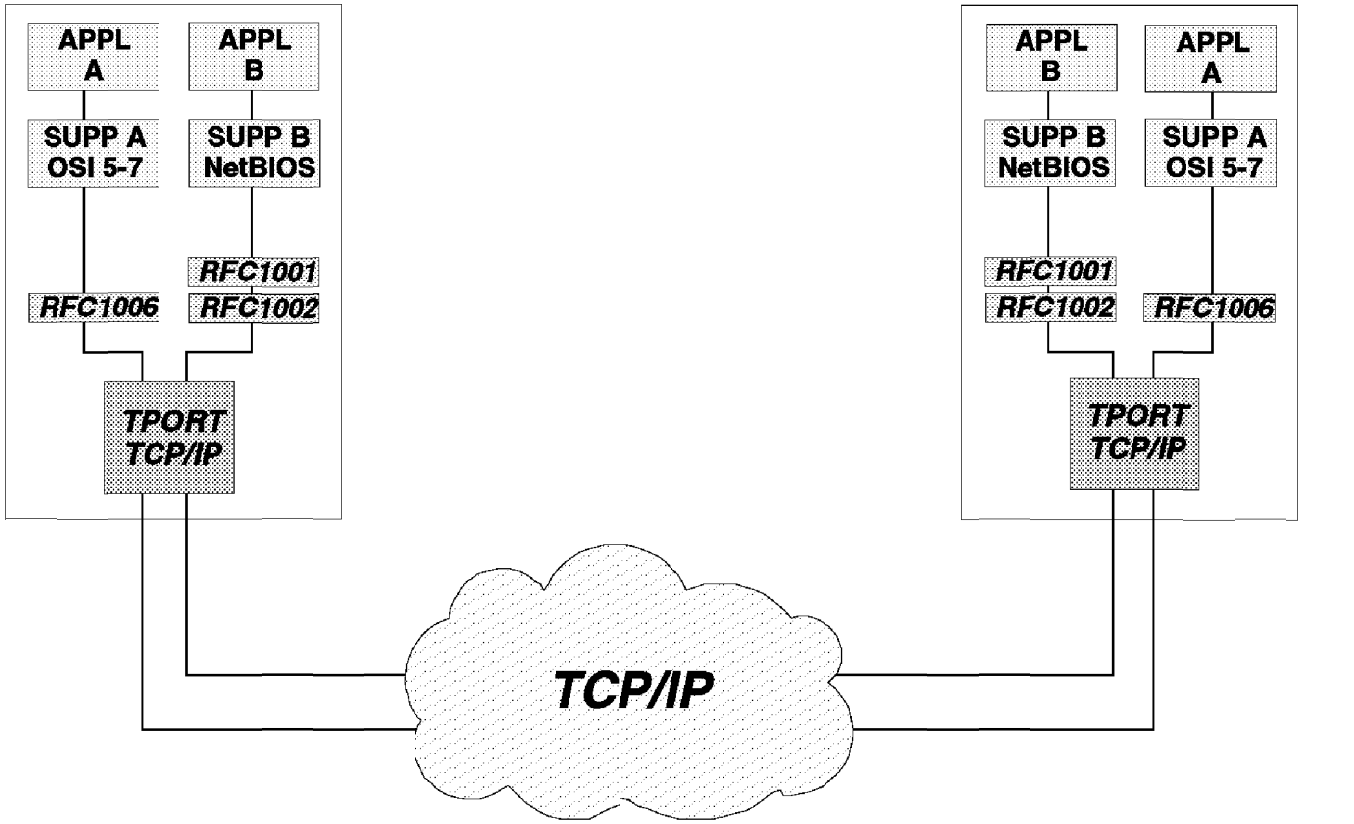

© 1993 IBM Corporation

# Native Multiprotocol Routers



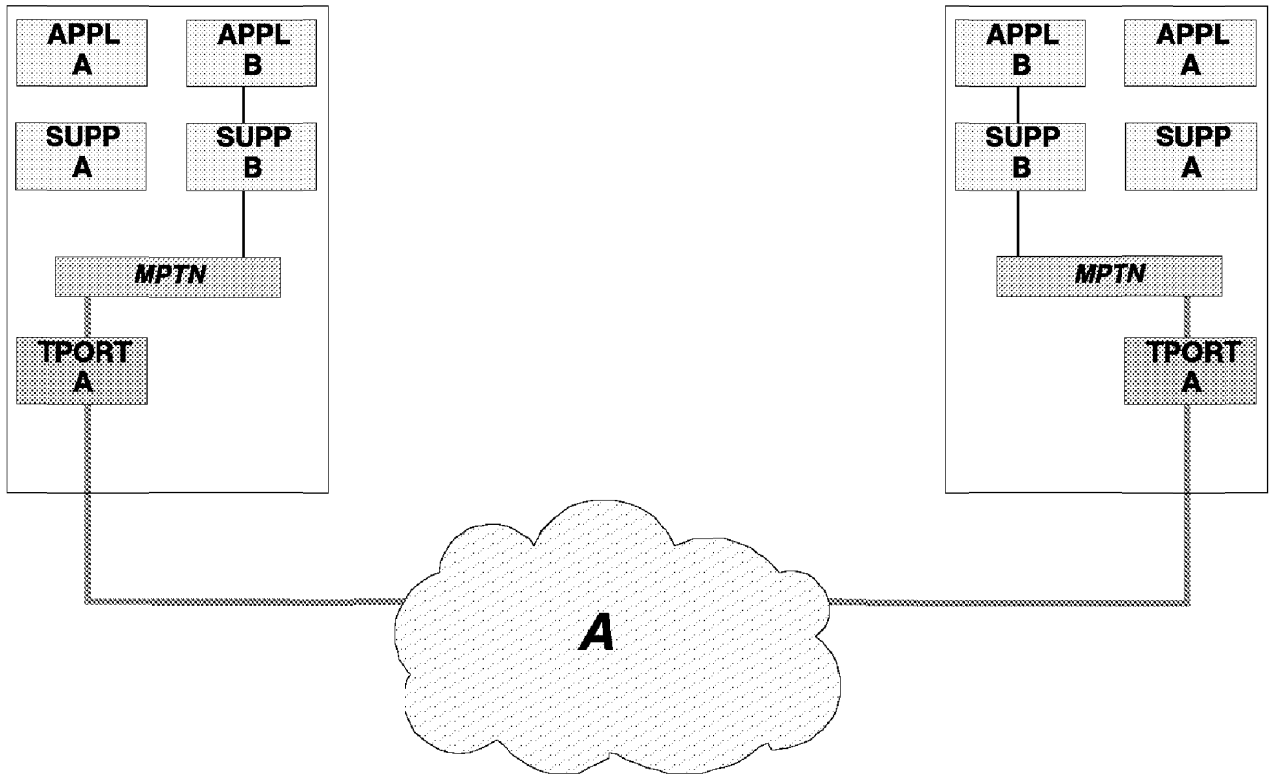

© 1993 IBM Corporation

# Mixed Protocol Standards -- RFCs



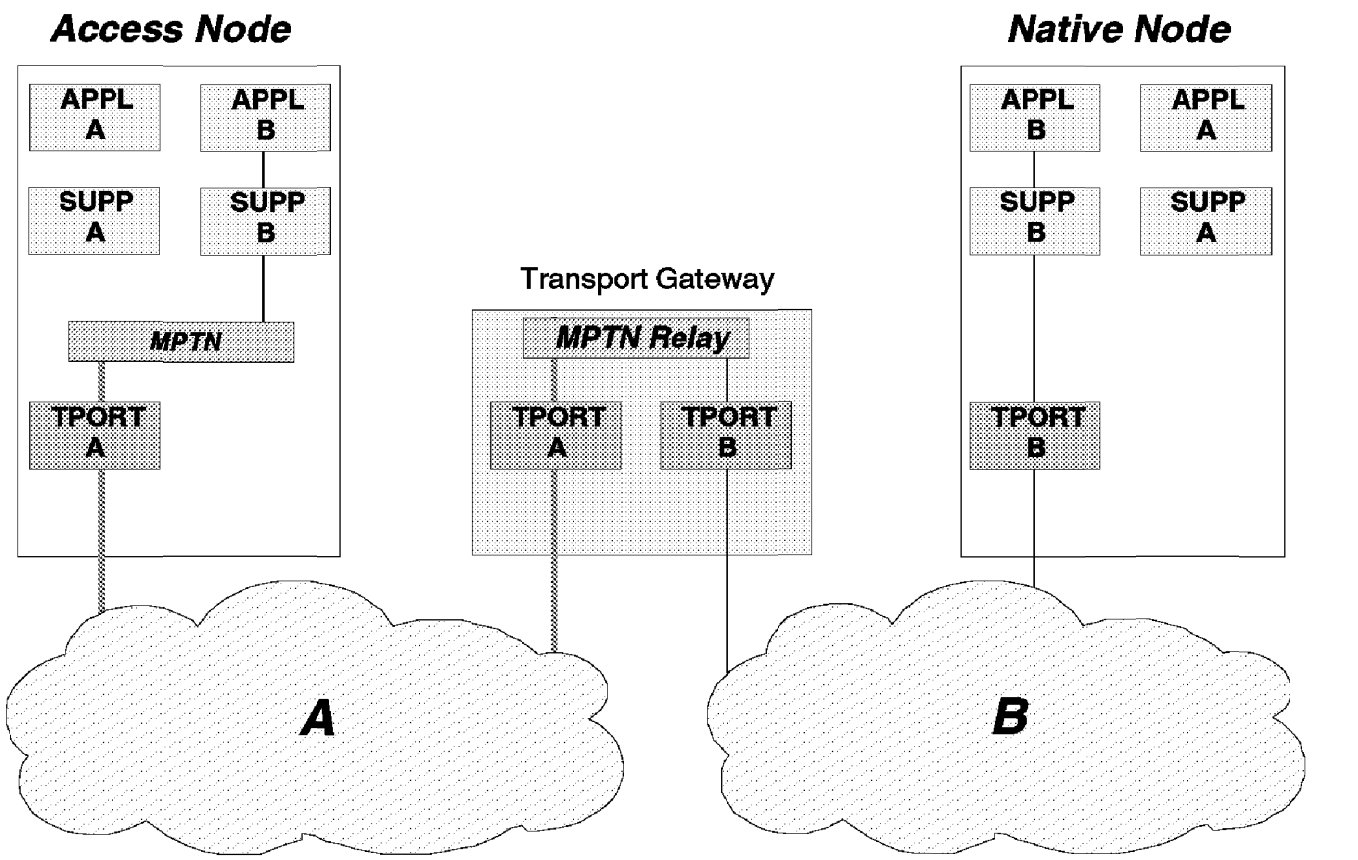

© 1993 IBM Corporation

# MPTN Access Nodes at End Points



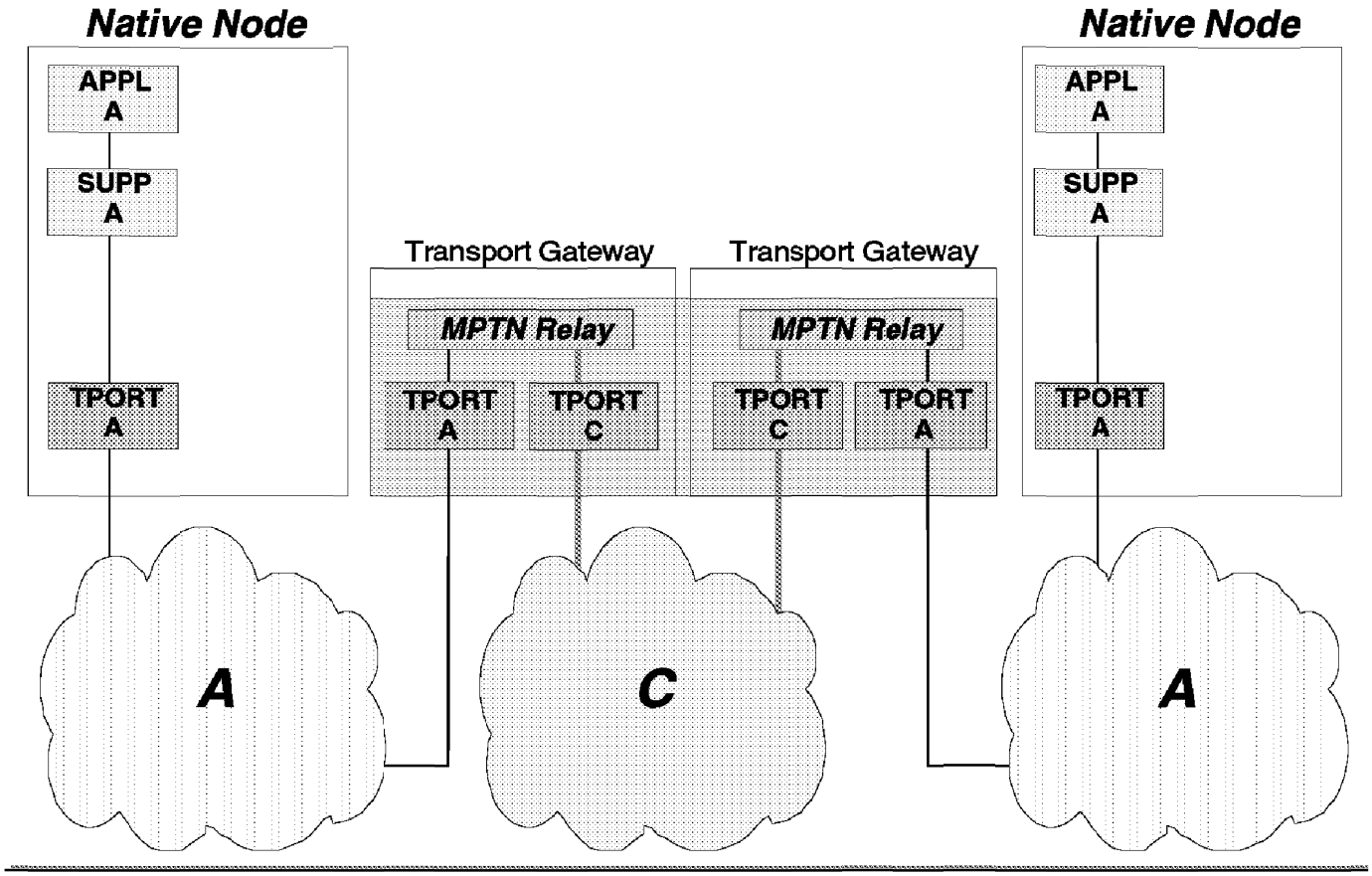

© 1993 IBM Corporation

# MPTN Gateway



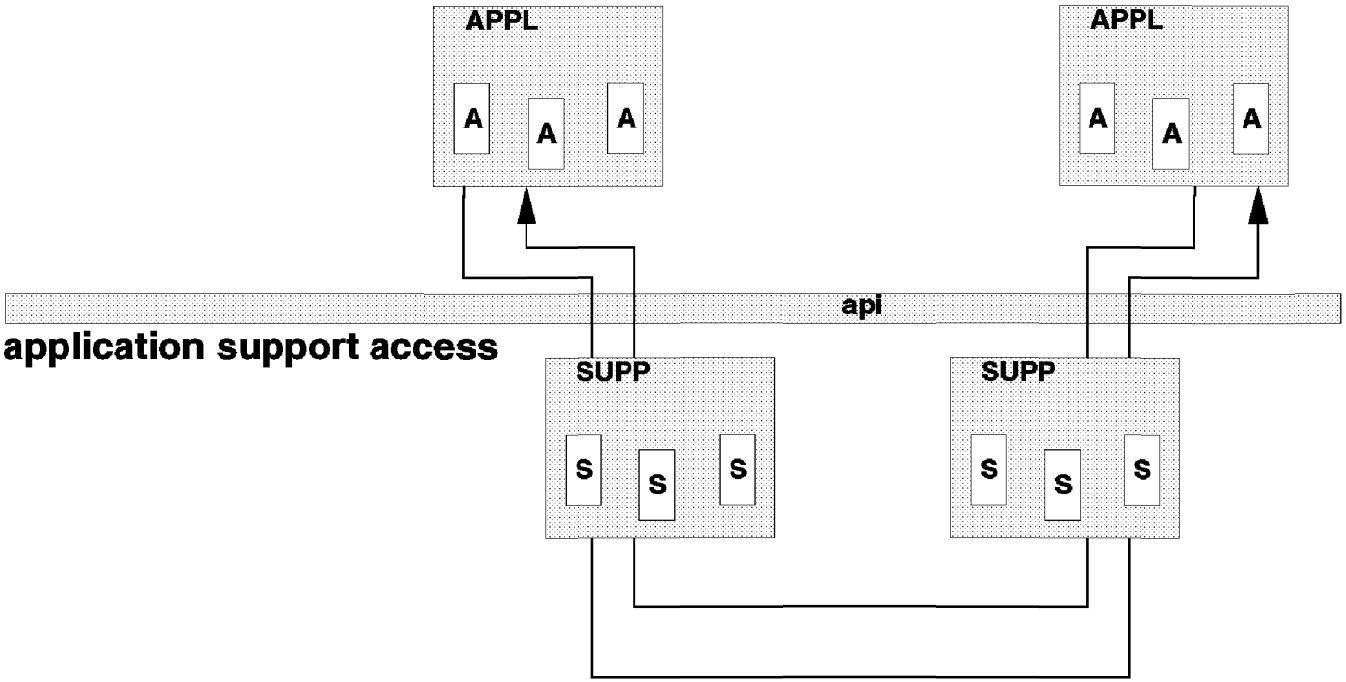

© 1993 IBM Corporation

# MPTN Gateways - Intermediate Network



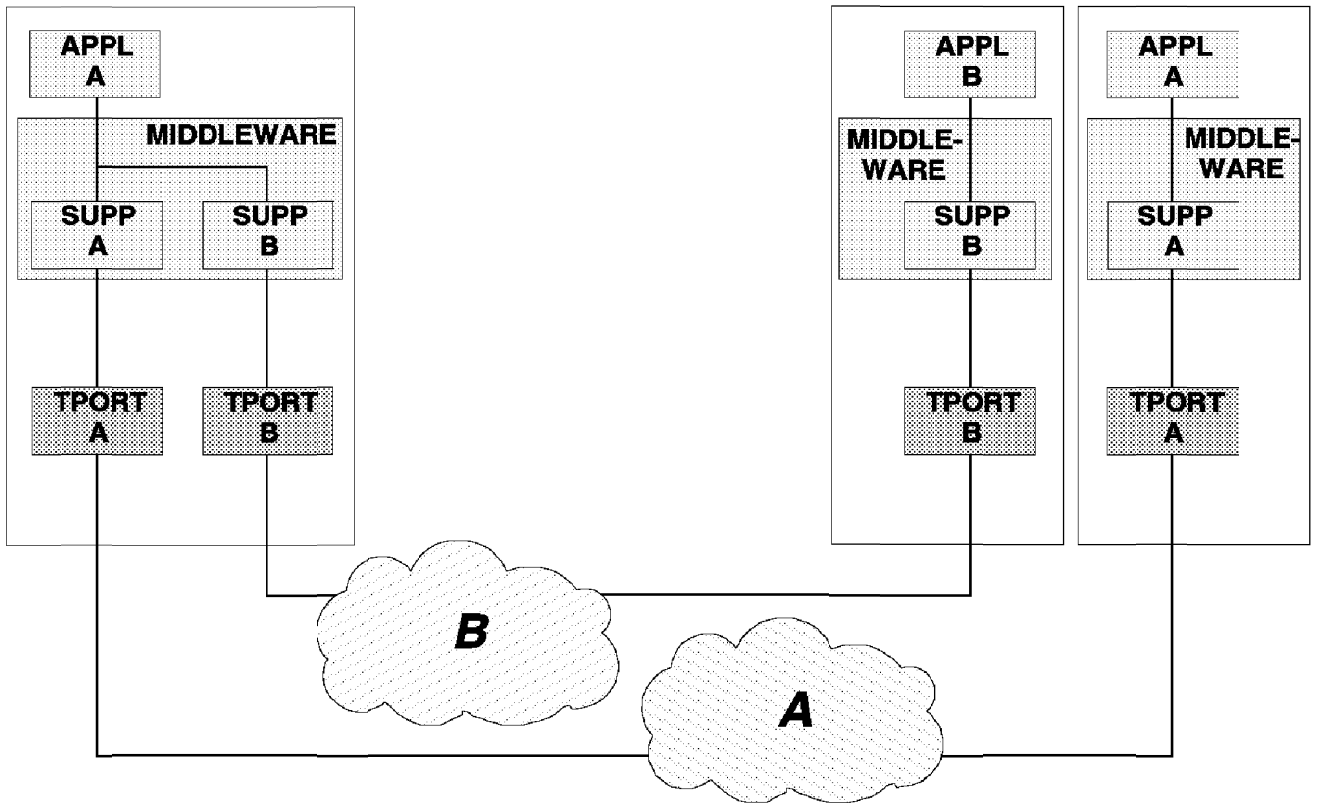

© 1993 IBM Corporation

# Application Support Technologies



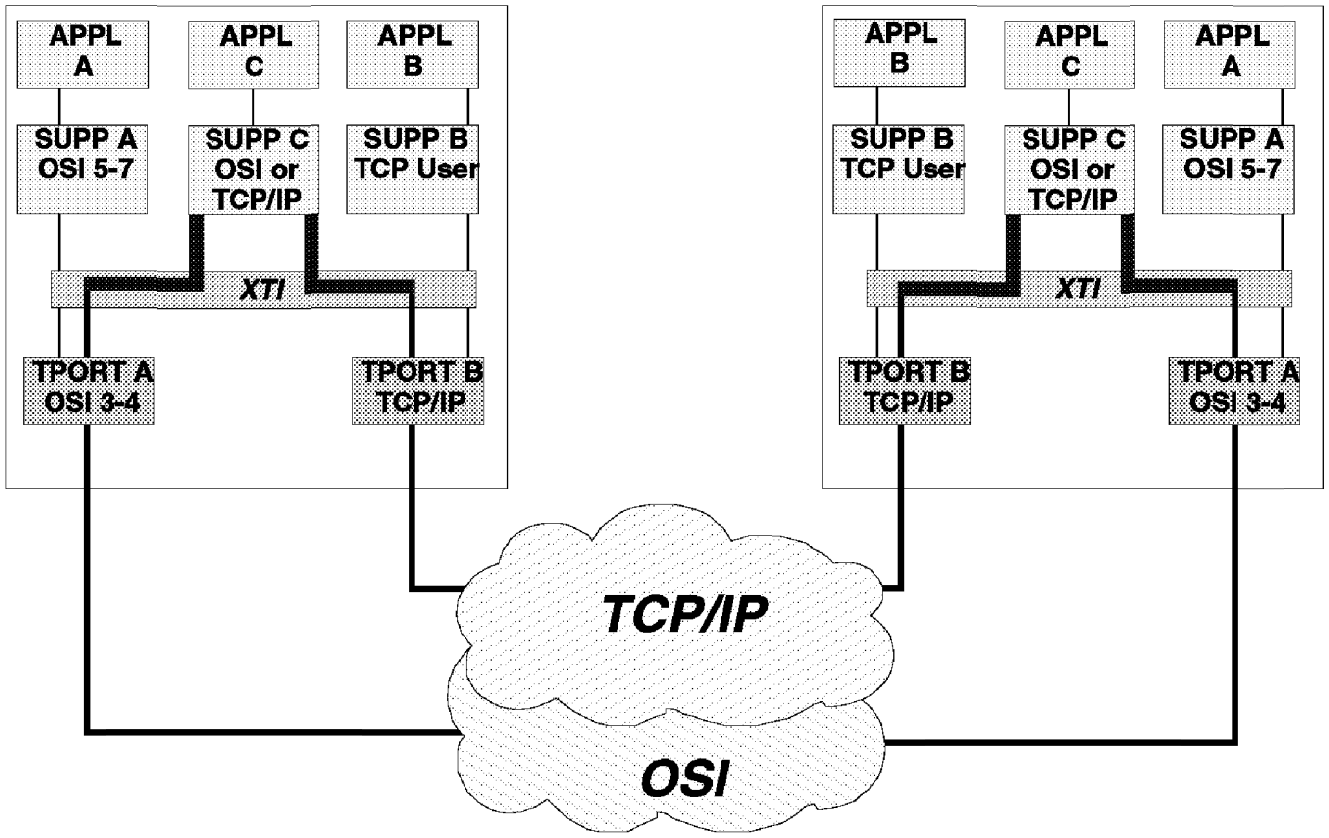

© 1993 IBM Corporation

# Middleware Technique



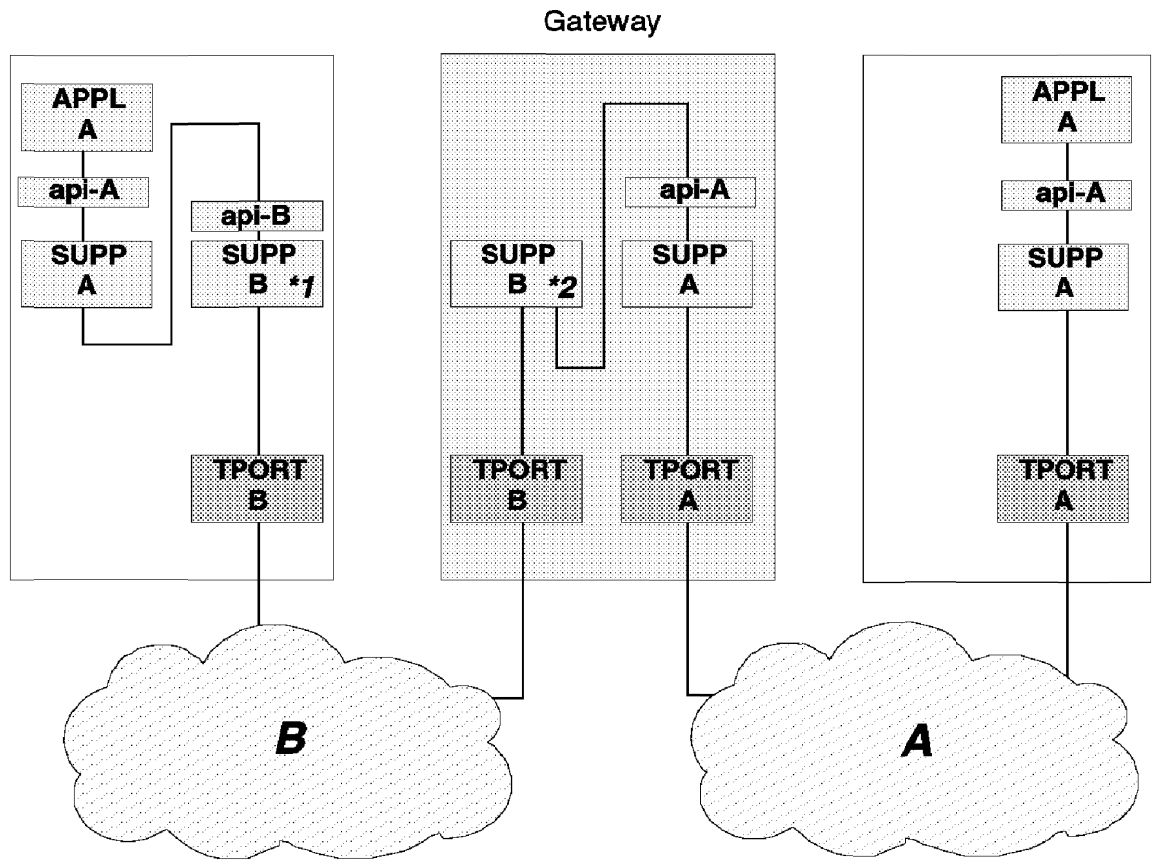

© 1993 IBM Corporation

# Selectable Transport Standard -- XTI




© 1993 IBM Corporation

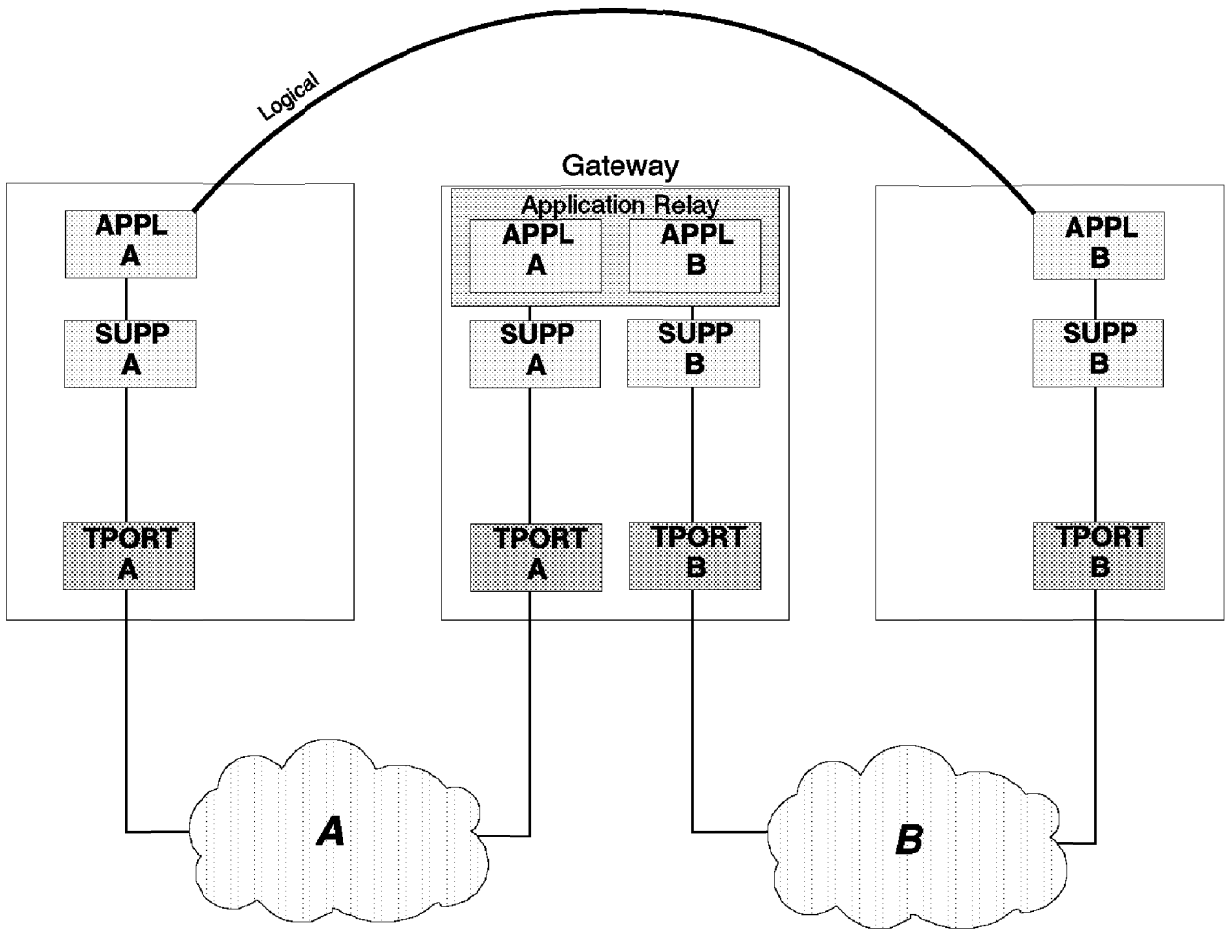
# Remote API




*1 Communications Client Role
*2 Communications Server Role

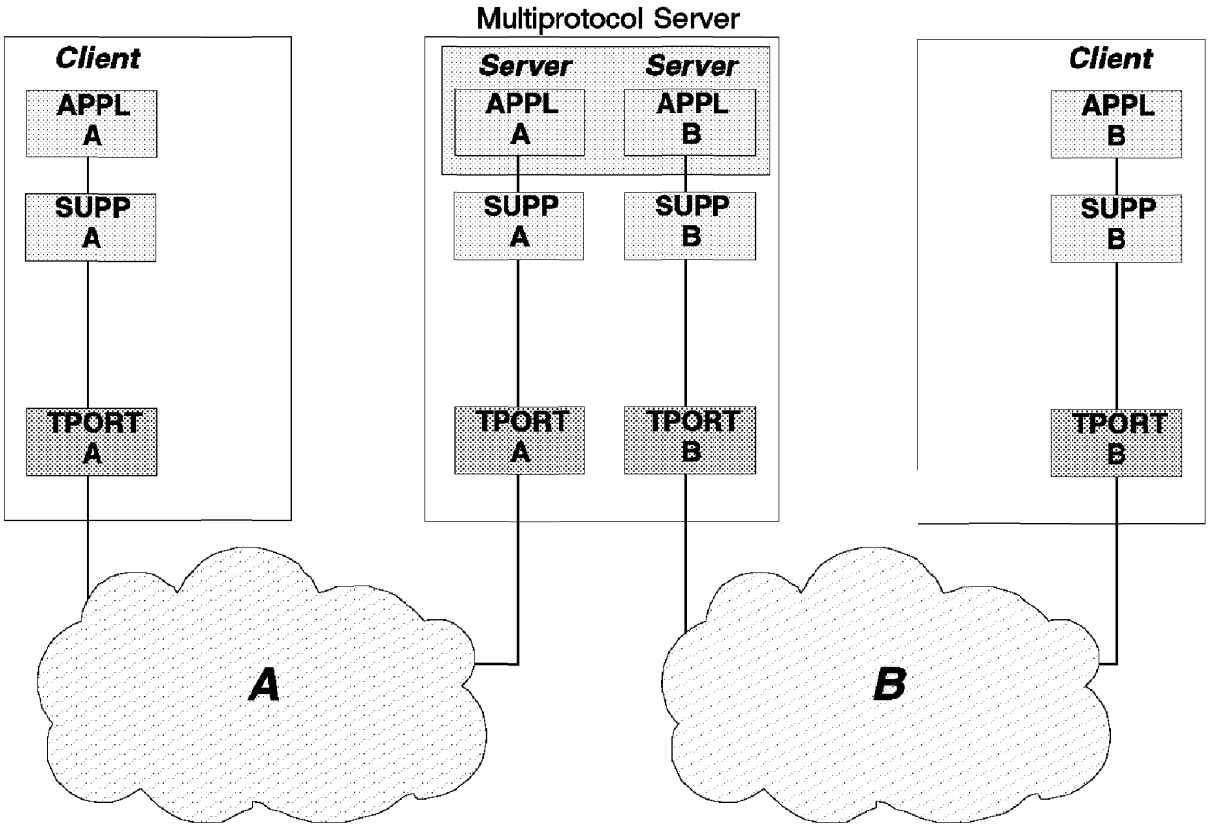
© 1993 IBM Corporation

# Application Gateway



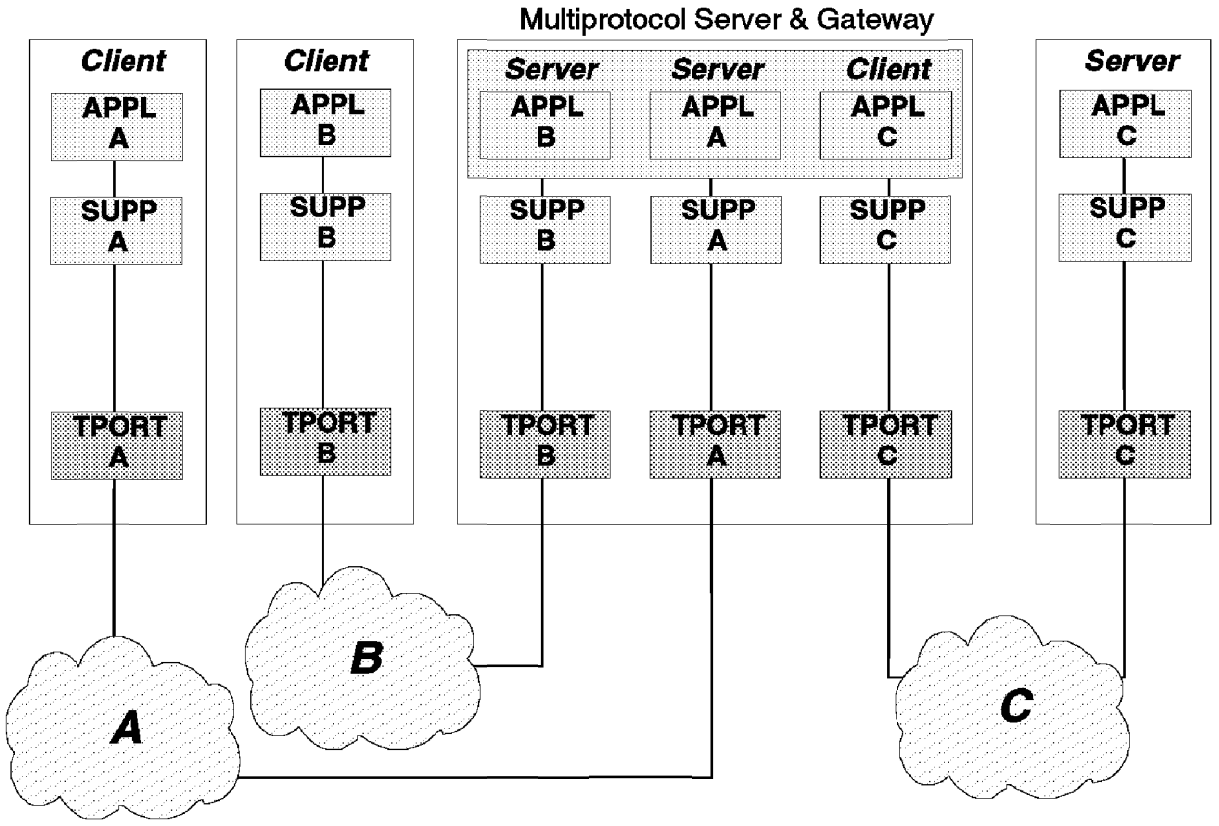

© 1993 IBM Corporation

# Multiprotocol Server

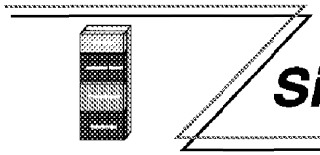



© 1993 IBM Corporation

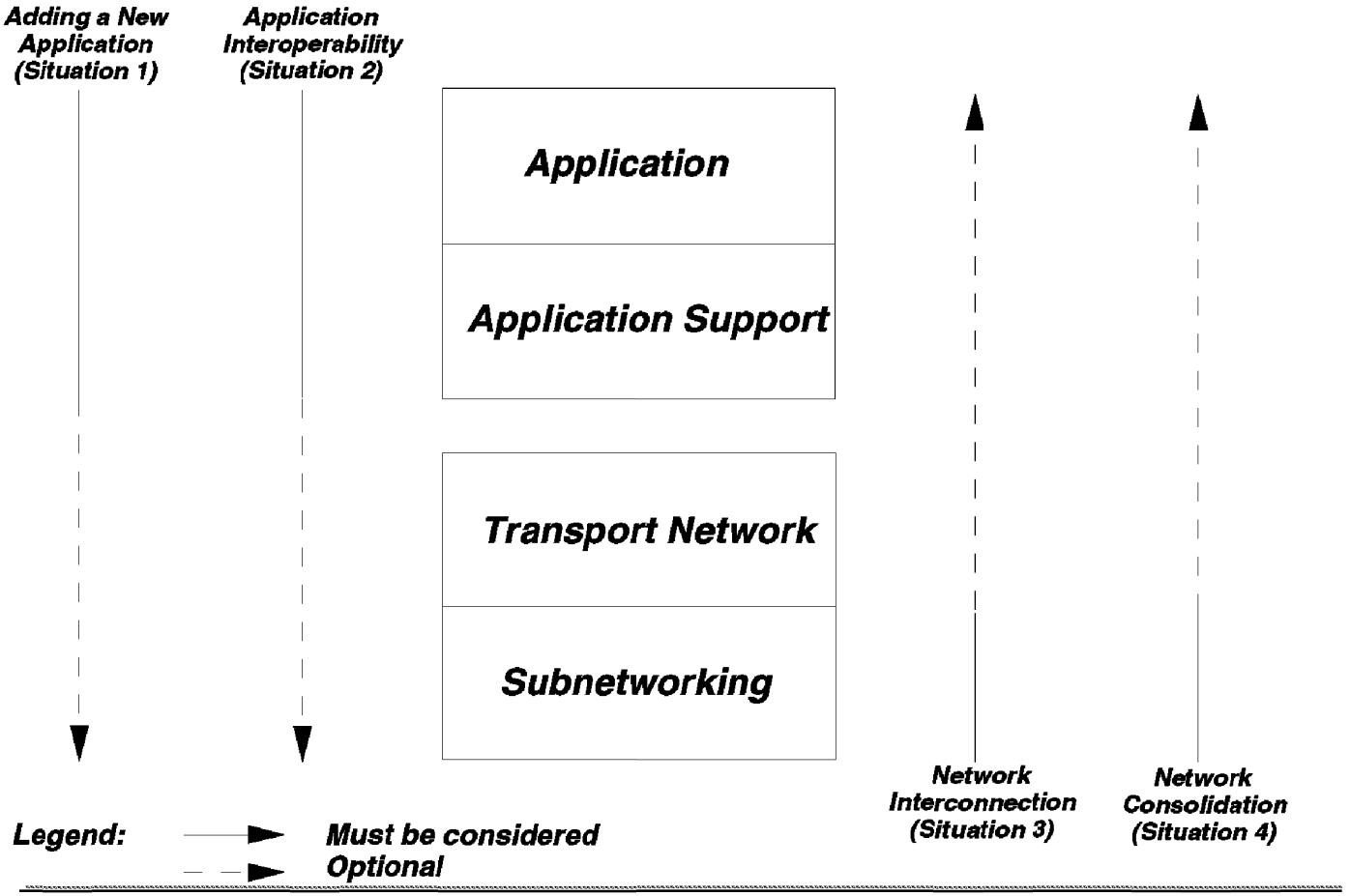
# Combination Server and Gateway




© 1993 IBM Corporation



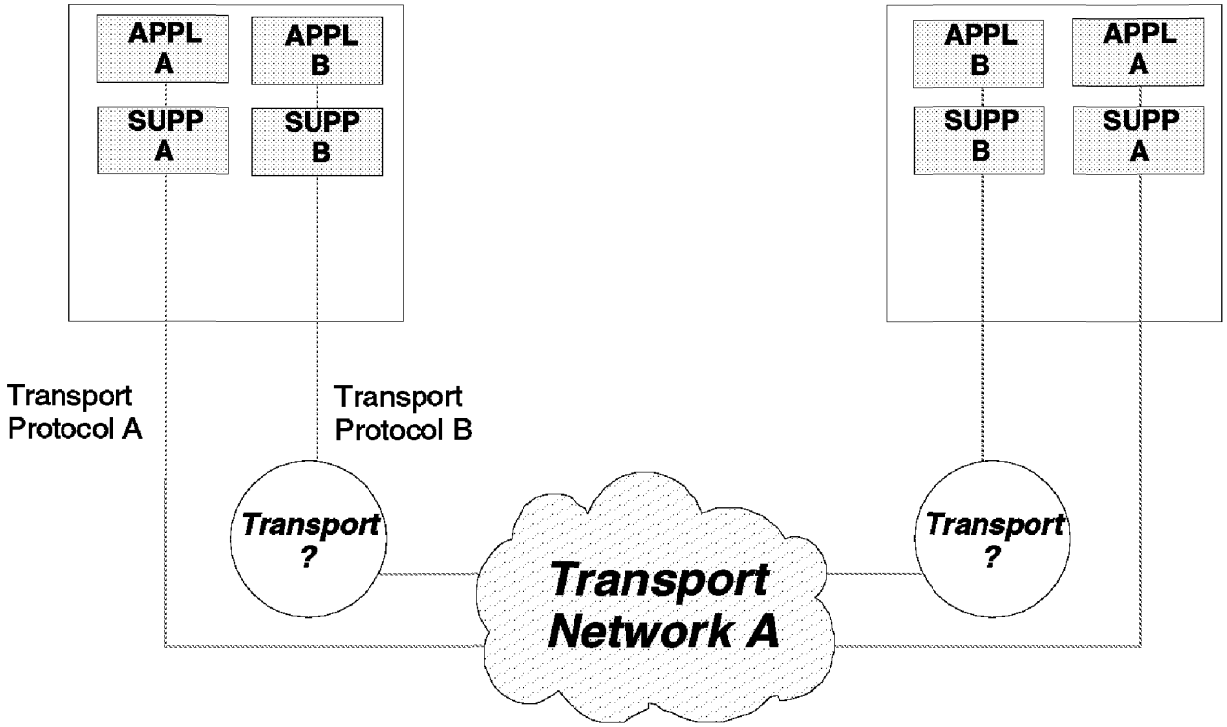
# Situations and Range of Consideration



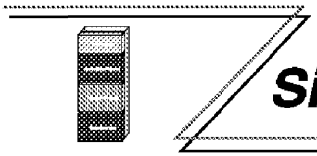

© 1993 IBM Corporation

# Situation 1A: Adding a New Application

***Adding a new application B to an existing network***




© 1993 IBM Corporation



# Situation 1A: Technology Choices

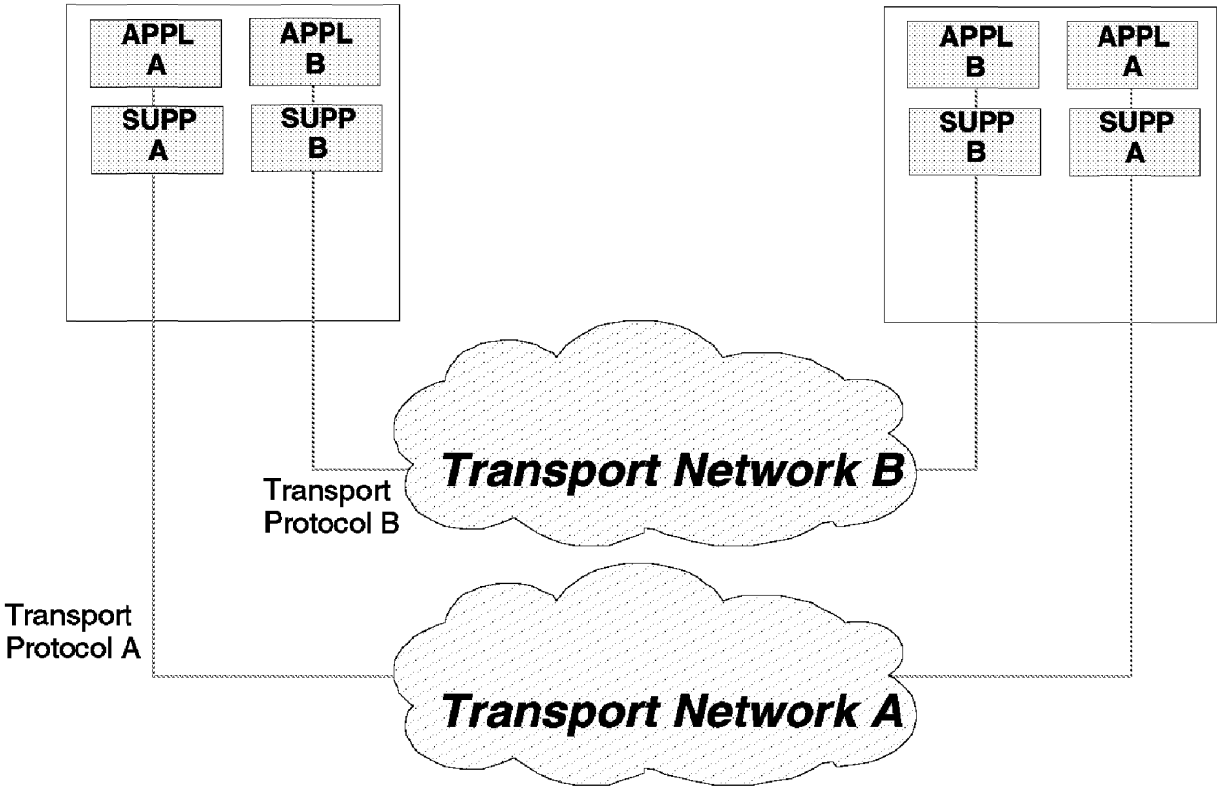
*Adding a new application to an existing network*

| | |
|---|---|
| *Application* | |
| *Application Support* | *Middleware*<br>*XTI* |
| *Transport Network* | *MPTN*<br>*Mixed-Protocol Standards* |
| *Subnetworking* | *Encapsulation* |

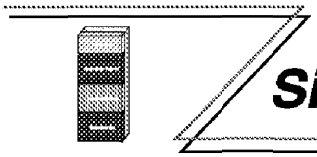
© 1993 IBM Corporation

# Situation 1B: Adding a New Application

***Adding a new application B via a separate transport network***




© 1993 IBM Corporation



# Situation 1B: Technology Choices

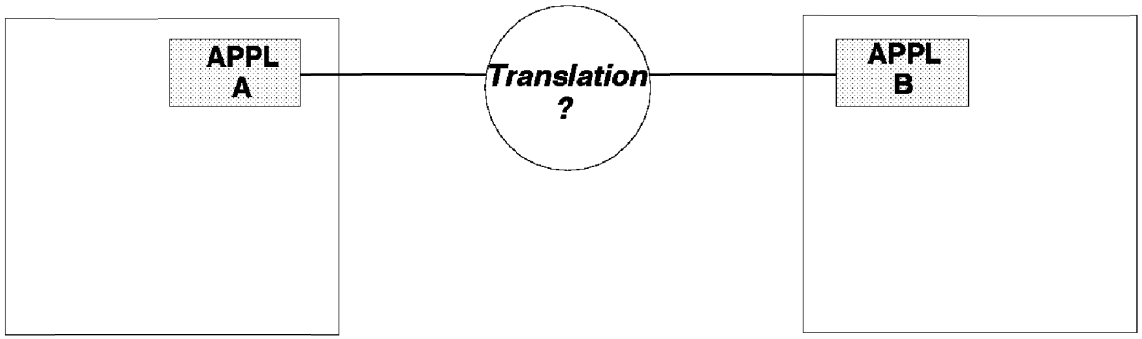
*Adding a new application via a separate transport network*

| | |
|---|---|
| *Application* | |
| *Application Support* | |
| *Transport Network* | *Multiprotocol Routers* |
| *Subnetworking* | *Remote Bridge, DLS, Packet Interface*<br>*LAN*<br>*Bandwidth Management (Multiplexor)*<br>*Separate WANs* |

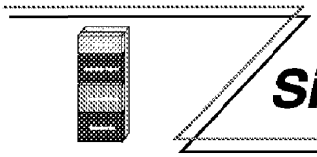
© 1993 IBM Corporation

# Situation 2A: Application Interoperability

***Application interoperability with application-level translation***




© 1993 IBM Corporation



# Situation 2A: Technology Choices

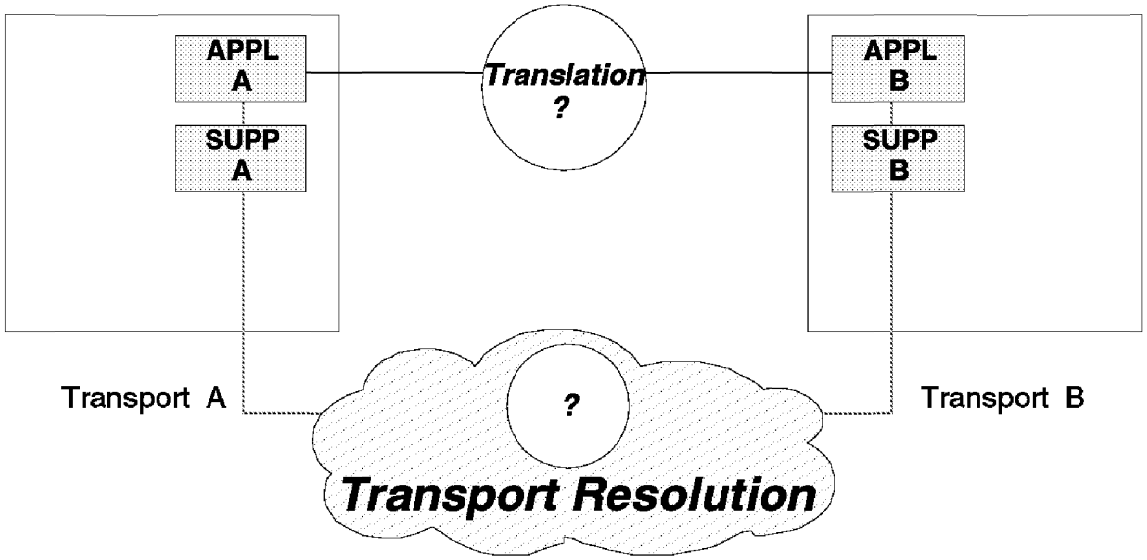
*Application interoperability with application-level translation*

| | |
|---|---|
| *Application* | |
| *Application Support* | *Multiprotocol Servers*<br>*Remote APIs*<br>*Middleware*<br>*XTI* |
| *Transport Network* | *MPTN*<br>*Mixed-Protocol Standards* |
| *Subnetworking* | *Encapsulation* |

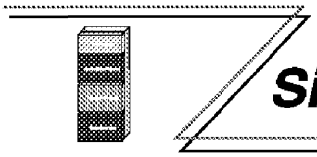
© 1993 IBM Corporation

# Situation 2B: Application Interoperability

***Application interoperability with application-level translation and transport resolution***

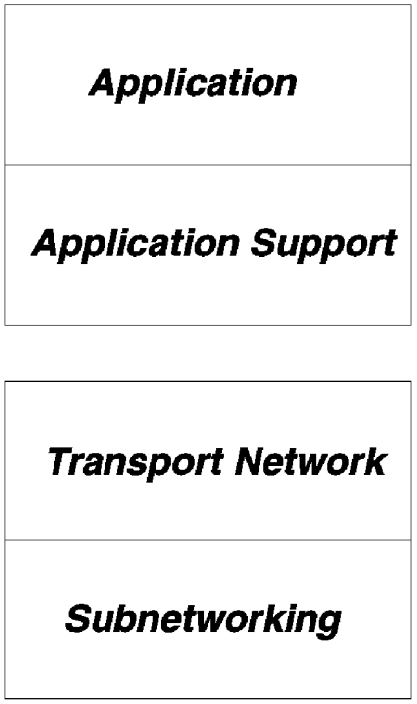



© 1993 IBM Corporation



# Situation 2B: Technology Choices

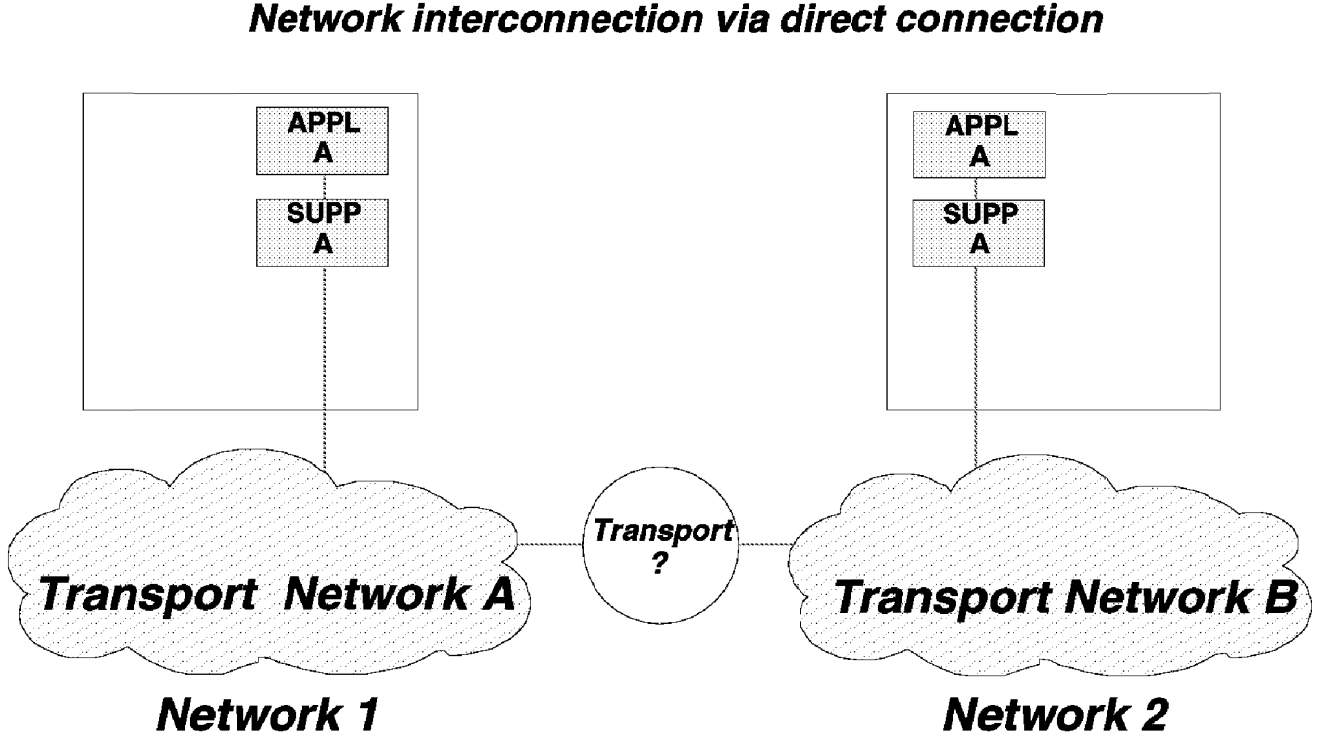
*Application interoperability with application-level translation and transport resolution*



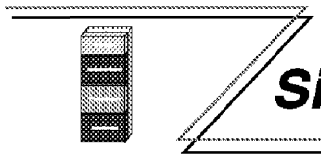

*Multiprotocol Servers*
*Application Gateways*

© 1993 IBM Corporation

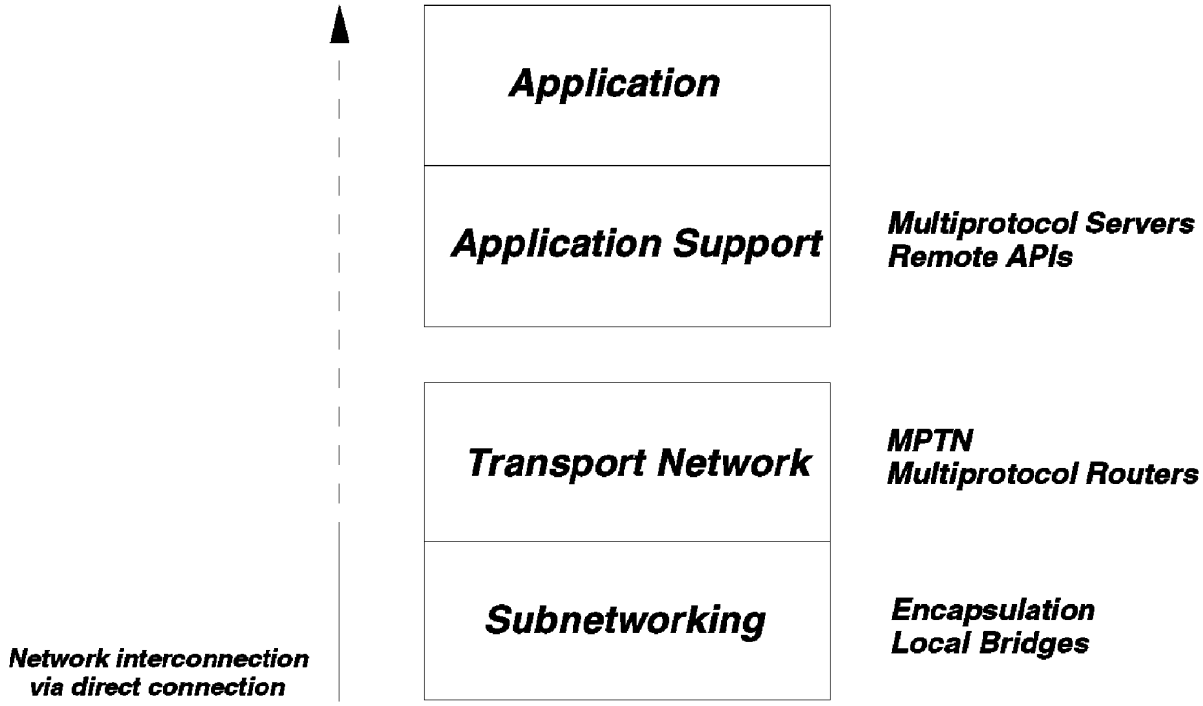
# Situation 3A: Network Interconnection




© 1993 IBM Corporation

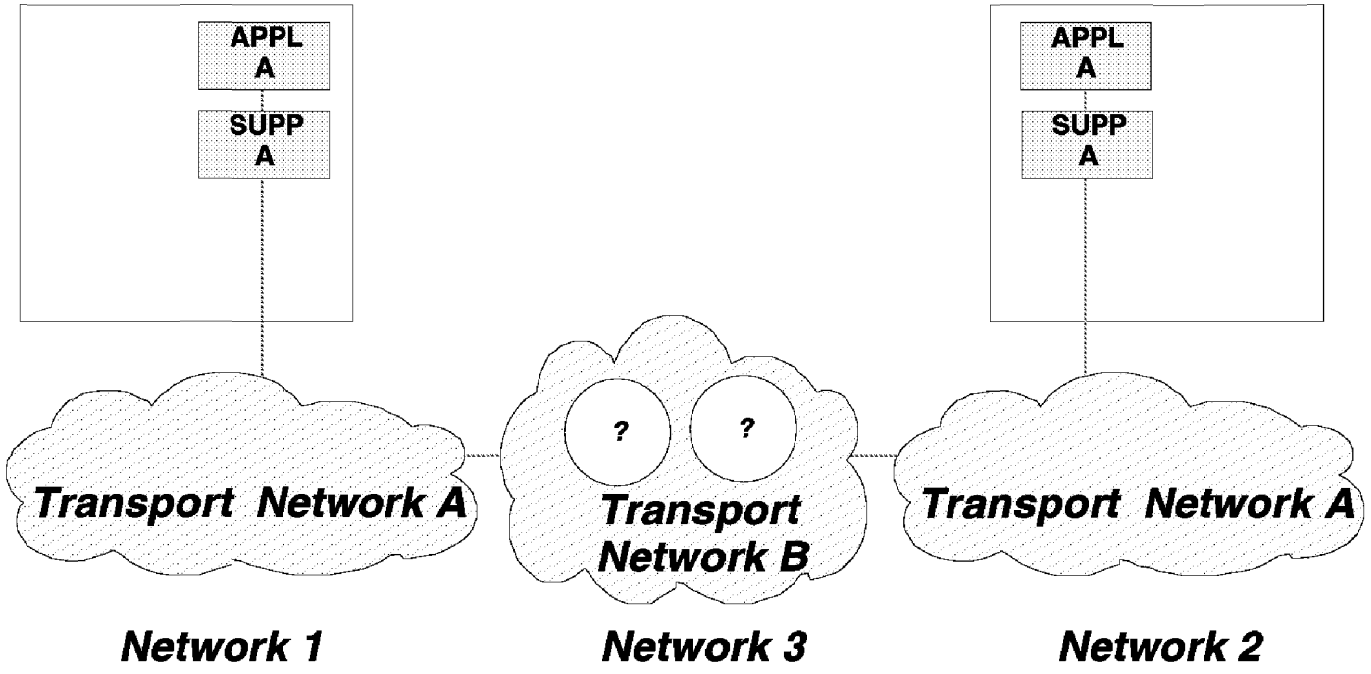


# Situation 3A: Technology Choices

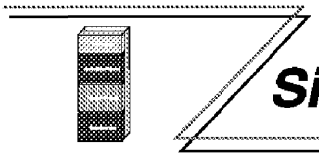



© 1993 IBM Corporation

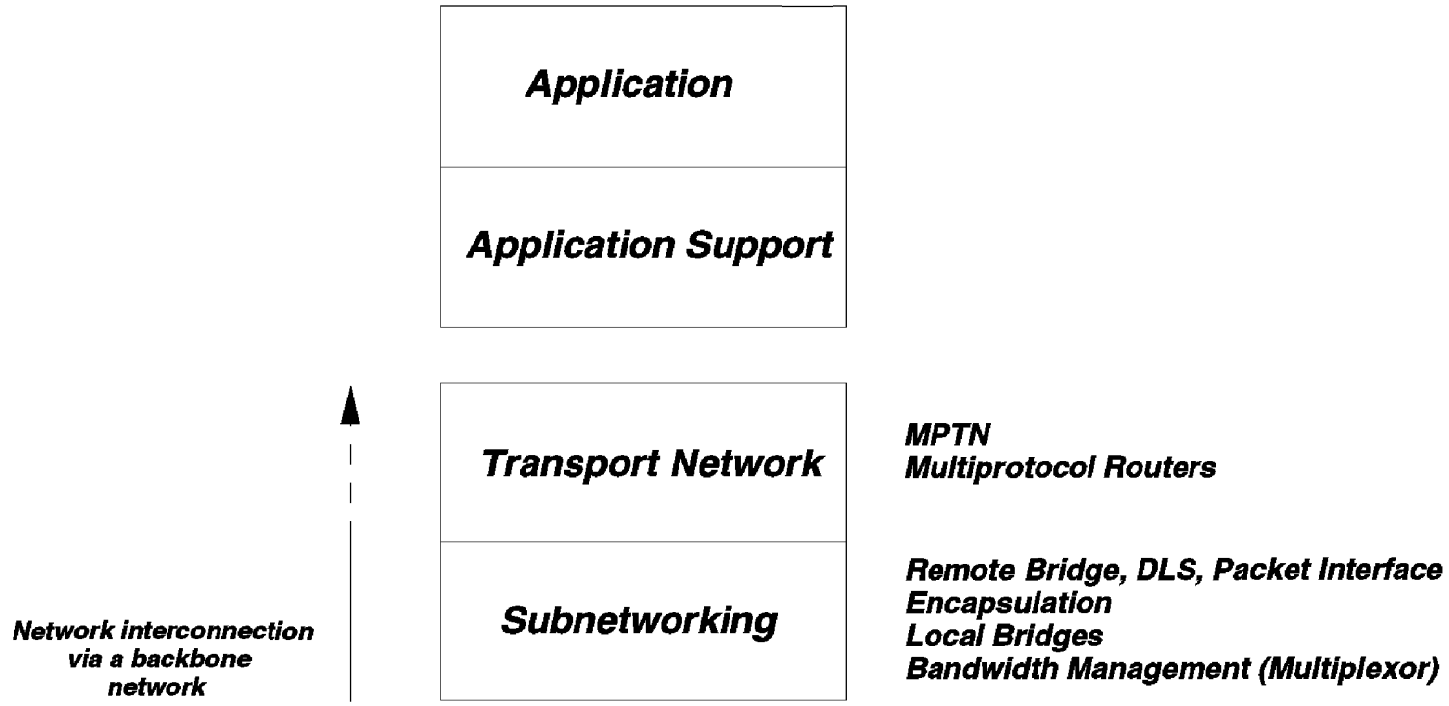
# Situation 3B: Network Interconnection




© 1993 IBM Corporation

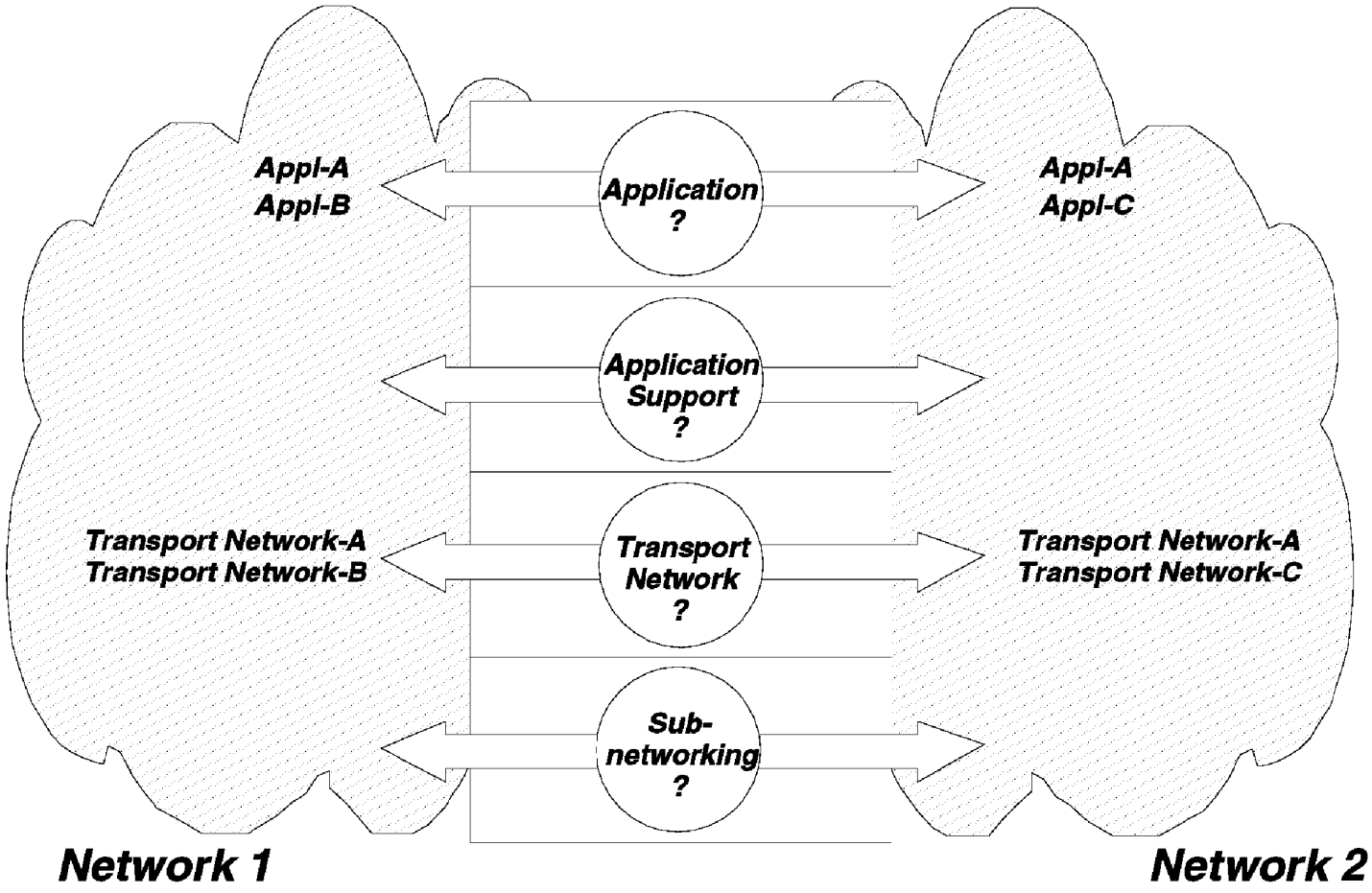


# Situation 3B: Technology Choices

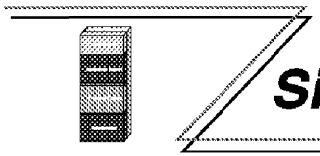



© 1993 IBM Corporation

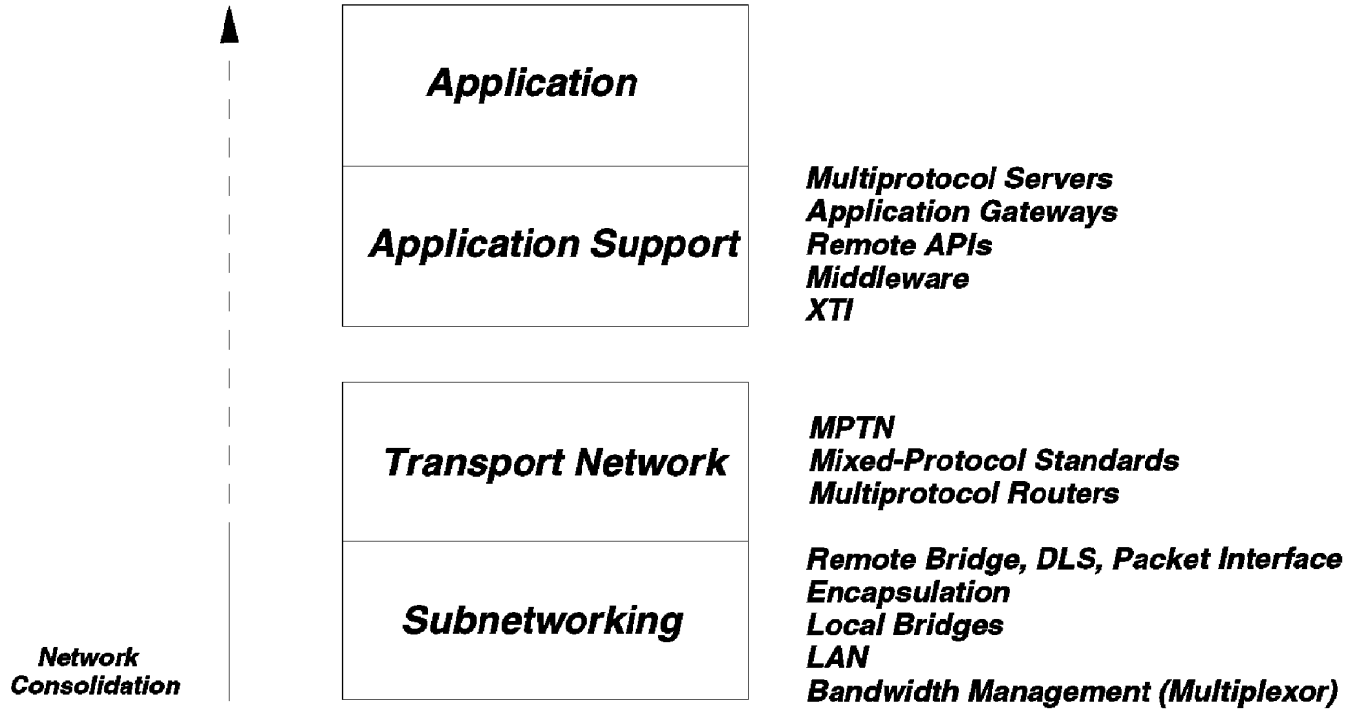
# Situation 4: Network Consolidation




© 1993 IBM Corporation



# Situation 4: Technology Choices




© 1993 IBM Corporation

# Technology Choices by Situation

**LEGEND:**
**X = Available Choice**

| | Subnetworking | | | | | | | | Transport Network | | | Application Support | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Separate WANs | Bandwidth Management (Multiplexor) | LAN | Local Bridges | Encapsulation | Remote Bridges | DLS | Packet Interface | Multiprotocol Routers | Mixed-Protocol Standards | MPTN | XTI | Middleware | Remote APIs | Application Gateways | Multiprotocol Servers |
| **Situation 1A**<br>**Adding a new application B to an existing network** | | | | | X | | | | | X | X | X | X | | | |
| **Situation 1B**<br>**Adding a new application B via a separate transport network** | X | X | X | | | X | X | X | X | | | | | | | |
| **Situation 2A**<br>**Application interoperability with application-level translation** | | | | | X | | | | | X | X | X | X | X | | X |
| **Situation 2B**<br>**Application interoperability with application-level translation and transport resolution** | | | | | | | | | | | | | | | X | X |
| **Situation 3A**<br>**Network interconnection via direct connection** | | | | X | X | | | | X | | X | | | X | | X |
| **Situation 3B**<br>**Network interconnection via a backbone network** | | X | | X | X | X | X | X | X | X | | X | | | | |
| **Situation 4**<br>**Network consolidation** | | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

© 1993 IBM Corporation

# Index

## Special Characters



## A



## B



## C



## D



## E



## F



© Copyright IBM Corp. 1994

## G



## H



## I



## J



## L



## M



## N

# O



# P



# R



# S



# T

## U



## W



## X

# ITSO Technical Bulletin Evaluation RED000

**Introduction to Networking Technologies**

**Publication No. GG24-4338-00**

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. _____

IBML®

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
Department 985, Building 657
P.O. BOX 12195
RESEARCH TRIANGLE PARK NC
USA 27709-2195

Fold and Tape **Please do not staple** Fold and Tape



Cut or Fold Along Line

IBML®

Printed in U.S.A.

GG24-4338-00