Multiprotocol Transport Networking: Eliminating application dependencies on communications protocols

by D. Pozefsky
R. Turner
A. K. Edwards
S. Sarkar
J. Mathew
G. Bollella
K. Tracey
D. Poirier
J. Fetvedt
W. S. Hobgood
W. A. Doeringer
D. Dykeman

The Multiprotocol Transport Networking (MPTN) architecture is a general solution that breaks the binding between distributed applications and communications protocols. The MPTN architecture enables existing applications to run unmodified over any communications protocol. In this paper, we first present the trends in networking that resulted in today's networks supporting multiple communications protocols. Next, we describe the classes of problems this support causes. The MPTN architecture is described and presented as a solution to many of these problems. We also present several alternative solutions to the multiple communications protocol problem and compare them to the MPTN solution. Last, we describe the IBM AnyNet™ family of products that implement the MPTN architecture.

Over the past two decades, numerous communications protocols have been developed. Examples of such protocols include Systems Network Architecture (SNA),[1] Open Systems Interconnection (OSI),[2] Transmission Control Protocol/Internet Protocol (TCP/IP),[3] Network Basic Input/Output System (NetBIOS),[4] Internetwork Packet Exchange (IPX**),[5] Digital Equipment Company architecture specifications for Networks (DECnet**),[6] and AppleTalk**.[7] Each communications protocol has its own advantages and disadvantages in terms of network performance, cost, security, ease of management, and availability of applications. Distributed applications and the application programming interfaces (APIs) they use, such as sockets for TCP/IP and CPI-C (Common Programming Interface–Communications) for SNA, are typically bound to run on a single communications protocol. This binding prohibits the independent selection of applications and communications protocols. Either the choice of applications is limited to those using currently installed communications protocols, or networks run multiple protocols to support all the applications that users require.

0018-8670/95/$3.00 © 1995 IBM

As the number of installed communications protocols increases, the complexity of managing the network and the consequent operational costs also increase. Further, the interactions between the protocols increase the complexity of performance tuning, bandwidth allocation, resource contention, and other network considerations.

The Multiprotocol Transport Networking (MPTN) architecture is a general solution that breaks the binding between an application and a communications protocol, enabling users to reduce the number of protocols installed in the network while all desired applications are still supported. The MPTN architecture solves two major problems:

1. Application dependence—An *MPTN Access Node* separates the application and application support from the communications protocol. This separation enables applications that were written for one protocol to run over any other protocol. One very important benefit of MPTN is that existing applications do not need to change. For example, existing, unmodified TCP/IP applications written to a sockets interface can run over an SNA network.
2. Mixed protocol networking—*MPTN Transport Gateways* concatenate networks running different protocols so that they function like a single network. The MPTN architecture supports any configuration using MPTN gateways, including multiple MPTN gateway hops and parallel MPTN gateways.

This paper begins with some general trends in networking and the types of problems that have developed. It then presents an overview of the MPTN architecture and offers some networking scenarios where MPTN can be used. After the functions that make up the MPTN architecture are described, the MPTN solution is applied to real networking problems. The paper concludes with a comparison of MPTN to other multiprotocol solutions and a brief description of AnyNet*, IBM's family of products based on the MPTN architecture.

General trends in networking today. In the 1980s, the growth of local area networks (LANs) brought networking decisions in an enterprise down to the divisional and departmental levels. The application mix became the determining factor driving the decision about which communications protocols were to be supported on the LAN. In the late 1980s and early 1990s, there was an explosion of intracompany communications requirements between LANs and hosts, and between the applications required at the level of the division and department. At the same time, cross-company business ventures were growing with their communications needs. These trends increased the demand for global communications, both within an enterprise and between enterprises. Interdepartmental communications, merging of networks, and increased communication with suppliers, vendors, and business partners required support of a wide variety of communications protocols.

As a result, enterprises today are dependent on multiple vendors and technologies. Buyers and vendors of both hardware and software are looking to standards and consortiums for interface definitions that allow major networking software and hardware components to be essentially interchangeable. The Open Blueprint*, described in another paper in this issue,[8] is a framework that provides a structured perspective of these network components.

Classes of problems introduced. The networking trends identified above have resulted in three types of problems for customers:

1. Adding applications that use a new communications protocol—Consider a company that is running a TCP/IP network, but whose changing business needs make it necessary to add a new SNA application to the network. Traditionally, this addition would require that SNA protocols be supported in order to enable this new application. However, multiple networks significantly increase the cost of administration and management. Companies prefer to maintain only one network for the following reasons:
   - It is less expensive to maintain one network compared to the cost of maintaining parallel networks.
   - Configurations are less complicated, and maintenance costs are reduced.
   - When multiple protocols use the same physical network, the network resources cannot be fairly allocated among users.

   As illustrated in Figure 1, the preferred situation is to be able to administer the single TCP/IP network while running both TCP/IP and SNA applications.
2. Connecting unlike networks—Consider a service company that has an SNA network that is

Figure 1 Adding new applications to the enterprise network

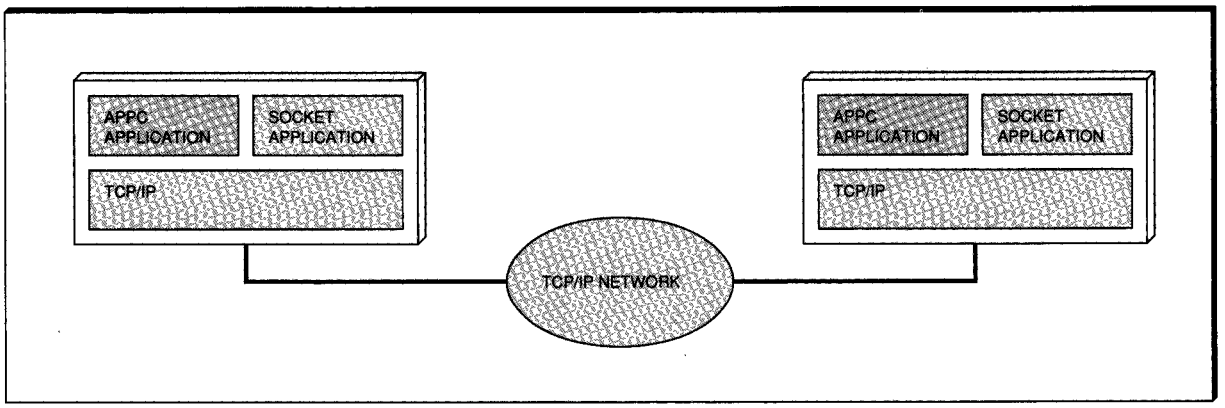
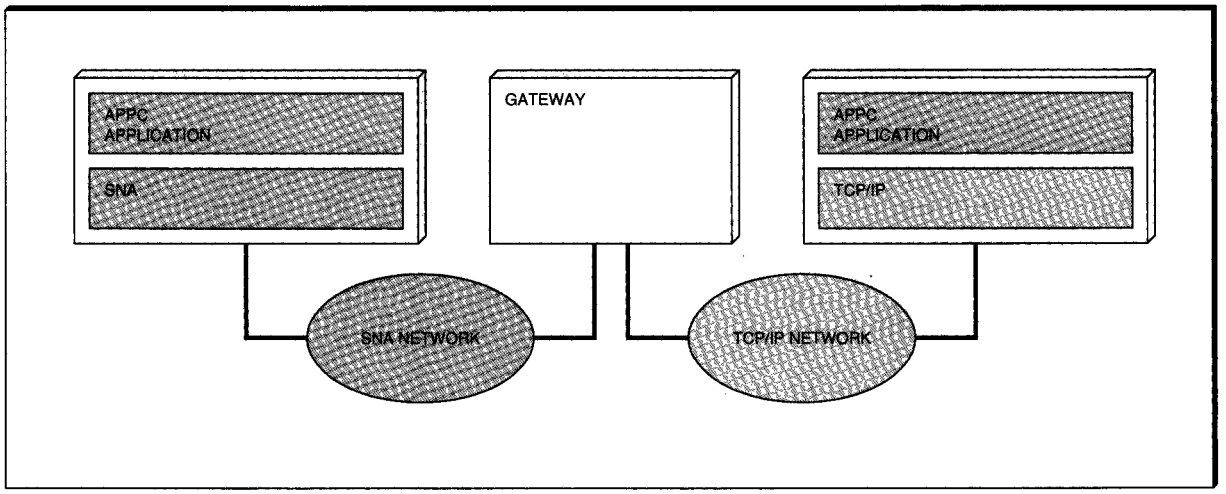


Figure 2 Connecting unlike networks using a gateway



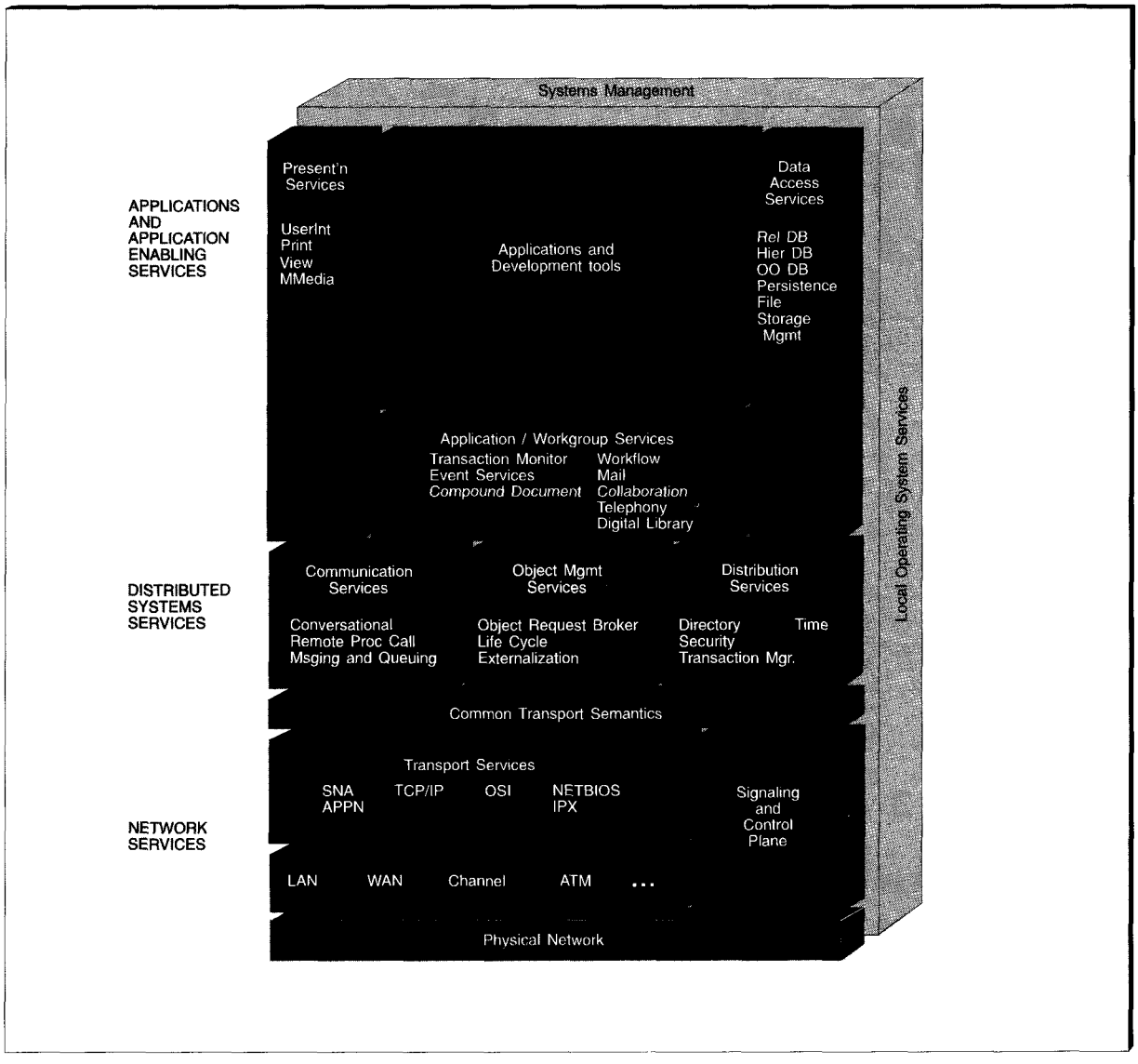
used to communicate with its clients. It acquires a new client who only has a TCP/IP network. The acquisition of the client company, although desirable financially, is problematic with respect to network communications over the two different protocols. The service company wants to run several SNA applications at the client's site, but the client does not want to change its network or maintain parallel networks. The service company would like to connect to the client's network through a gateway and run the SNA applications as shown in Figure 2.

3. Connecting LANs through a backbone network—Consider a company that has operations in New York and Los Angeles. The sites are connected by a backbone network running SNA. The company has LANs at each location running NetBIOS. When people from New York visit the Los Angeles site, they would like to be able to access their NetBIOS servers across the backbone from a workstation in Los Angeles. The company would like to connect the LANs through the backbone network.

MPTN overview

The Open Blueprint provides a framework for addressing the networking challenges of today and tomorrow, including network integration, applica-

Figure 3 Open Blueprint



tion freedom, systems and network management, and investment protection in times of rapid change. MPTN is a single architecture that addresses all of these challenges, providing a unified approach for solving multiple problems.

Within the Open Blueprint, MPTN function can be found in the layer labeled "Common Transport Semantics" (CTS). As seen in Figure 3, this layer is a common boundary within virtually all contemporary protocol stacks. This boundary separates the lower layers of a protocol stack, including the transport layer, from the upper layers. Conceptually, this CTS layer allows applications that reside above the upper layers of a protocol stack to be independent of the lower protocol stack layers.

MPTN performs two basic functions. First, it enables application independence. That is, it allows applications to be separated from the underlying transport stack without any change to the applications. A machine that supports this function is

called an MPTN access node. Application independence solves many of the problems resulting from the tight binding between applications and communications protocols. Specifically:

- With MPTN access node capability, the communications protocol running on a network does not dictate what applications can be used on that network. Instead, applications can be chosen on the basis of their own merits.
- Similarly, MPTN allows the communications protocol to be chosen on its merits. The need to run a particular set of applications does not require the use of a specific communications protocol.
- With MPTN, the installed communications protocol does not determine the API used for developing new applications. The application programmer is thus free to choose an API based on its capabilities, instead of being forced to use the API that is bound to the communications protocol running on the network.

Note that the MPTN access node function only supports matching applications to communicate across a different communications protocol. In this context, the term *matching* means that the applications are written to the same (or compatible) APIs. Applications written to different APIs are generally connected by an application gateway program, for example, a mail gateway.

The second function performed by MPTN is network concatenation. This function is provided by an MPTN gateway. An MPTN gateway connects two (or more) networks running different protocols, and allows applications in one network to communicate with matching applications in another network. Multiple MPTN gateways may be found in the path between communicating applications.

An MPTN gateway not only handles the case in which the communicating partners are MPTN access nodes, but also handles partners that contain no MPTN function, or *native nodes*. Two important gateway configurations are a single MPTN gateway between partners on different networks, typically where one partner is an access node and one is not, and two MPTN gateways connecting native nodes across a network running a different communications protocol. An MPTN gateway extends the reach of applications across dissimilar networks, and therefore solves many of the problems resulting from network isolation. Specifically, the network concatenation function of MPTN eliminates:

- The inability to share data across network boundaries
- The duplication of application function, data, and management in different network-specific applications
- The need to install duplicate or parallel networks to achieve application connectivity

The complete MPTN architecture is ambitious and would be difficult to implement in one step. Instead, pieces of the architecture, such as access nodes and gateways for specific application transport combinations, are being implemented as individual products. Details of the product family will be given later in the section on AnyNet products. The following subsections provide an in-depth discussion of the details of the MPTN functions as defined by the architecture.

A functional description of MPTN. MPTN solves the problem of application independence by separating the application and application layers from the transport protocol by defining a generic transport-layer interface, the *transport layer protocol boundary* (TLPB). The subsections that follow describe in greater detail how the TLPB provides application independence and how problems associated with the notion of a generic transport-layer interface, namely the requirement for *protocol compensations* and *address mapping*, are solved in the MPTN architecture. These solutions form the basis of an MPTN access node. Later subsections describe how an MPTN gateway builds on the above concepts to provide network concatenation. The management of a heterogeneous network consisting of MPTN access nodes, native nodes, and MPTN gateways is also described.

The transport-layer protocol boundary. True protocol independence of applications is provided by the TLPB, a fundamental component of the MPTN architecture. The TLPB defines a unique transport-layer interface that includes transport services that are commonly provided by such protocols as TCP/IP, SNA, OSI, NetBIOS, and IPX. Products that implement the MPTN architecture modify the part of the native protocol implementation that normally makes transport-layer requests. These modifications convert API requests for communications services into TLPB service requests. The TLPB requests are then mapped to transport-layer services, using any available transport layer that can be used to reach the communications partner. The application itself remains unchanged.

Figure 4 The transport-layer protocol boundary and associated terminology

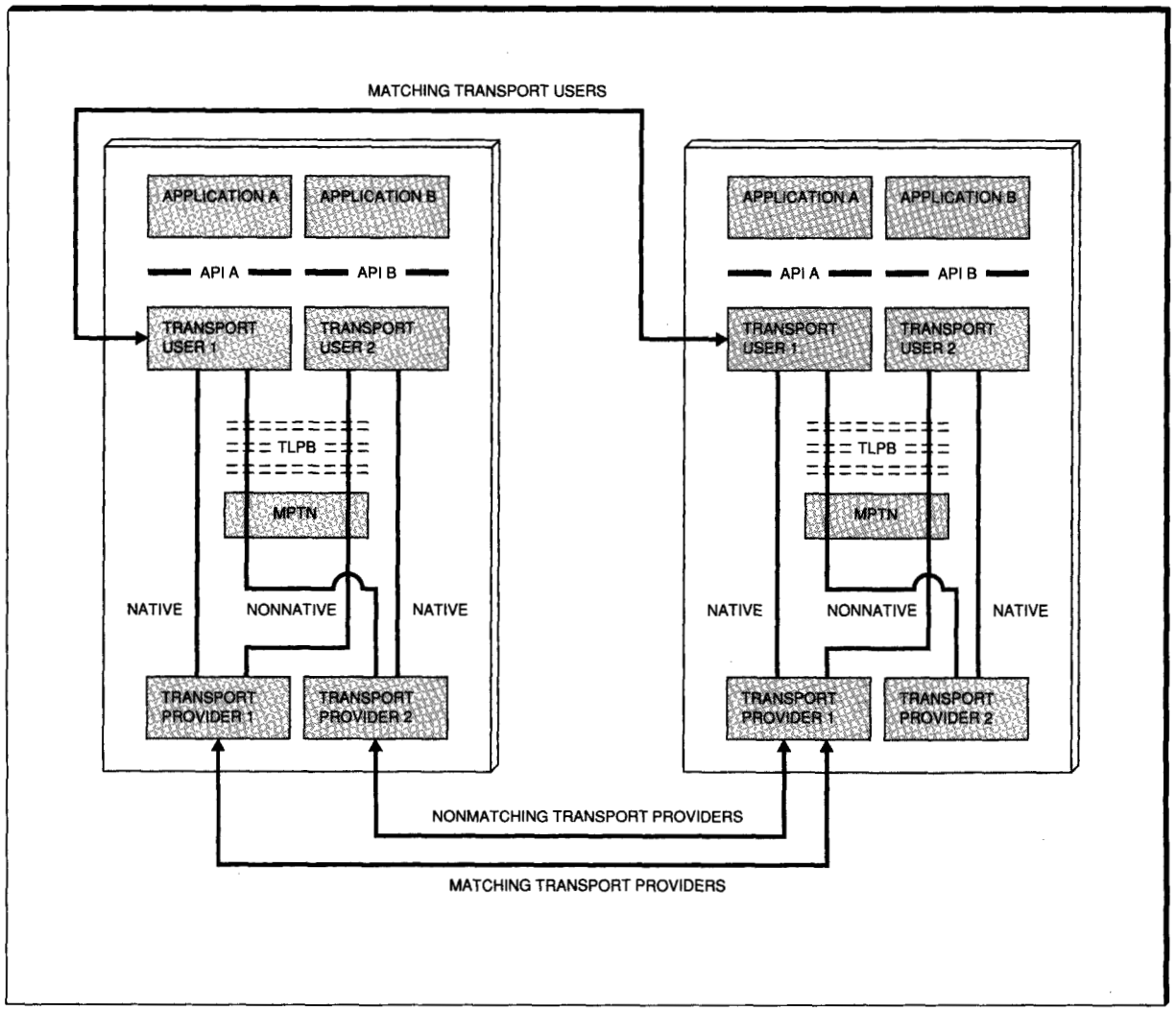
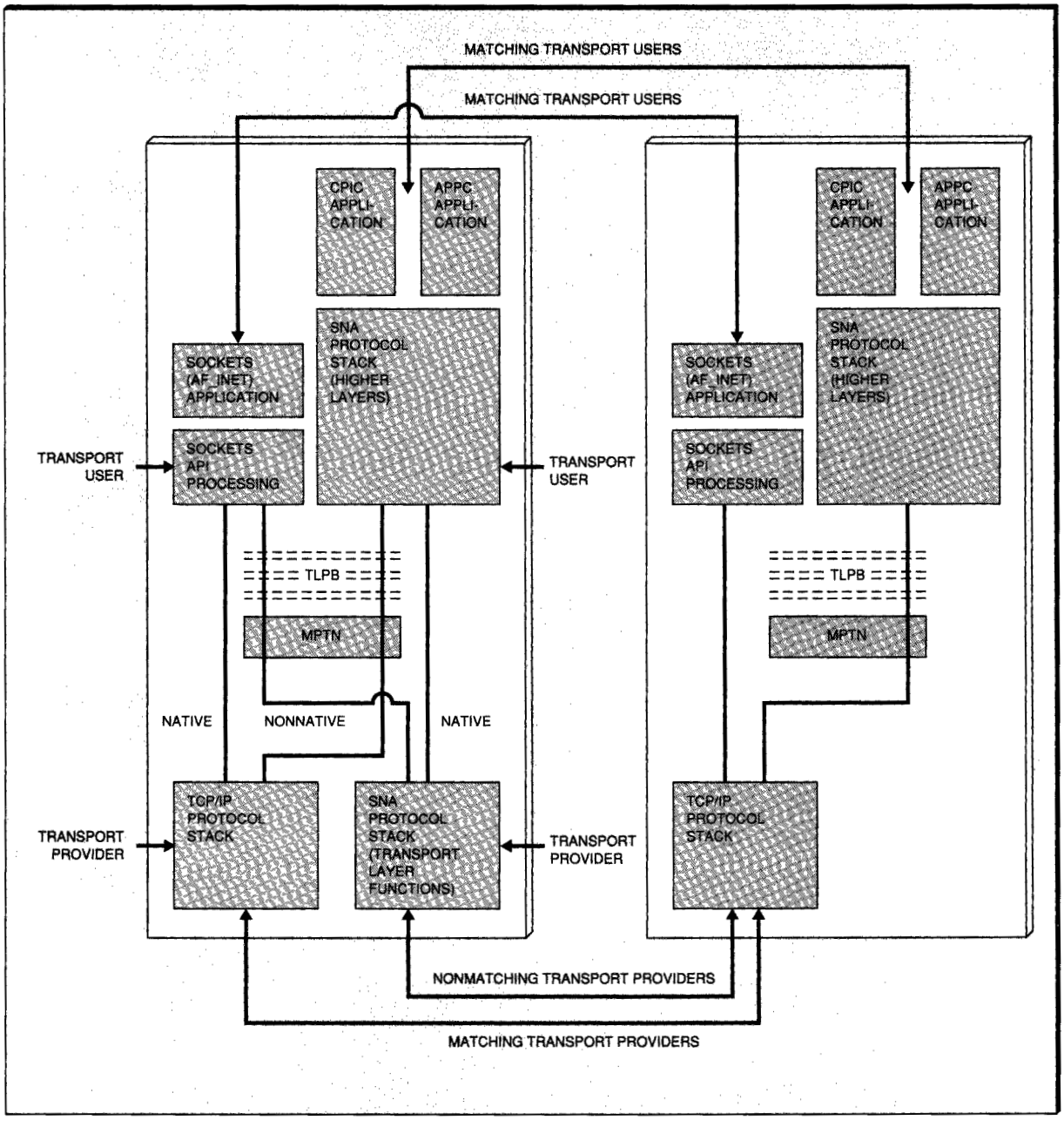


Figure 4 illustrates how the TLPB is used and also introduces some terminology that appears later in the paper. The figure shows that the TLPB separates communication subsystem components into those that use its functions and those that provide the underlying transport-layer services that the TLPB offers.

In the MPTN architecture, any component of a communications system that normally requests services from its native transport layer, requests those same services from the TLPB instead. Such a component is called the *transport user*. For a transport layer API such as sockets or NetBEUI (NetBIOS End User Interface), the transport user is the API processing layer. For an API such as CPI-C, which exposes higher-layer SNA functions, the transport user is the part of the SNA stack above the transport layer. The application and the communications API are above the transport user as shown in Figure 4. A protocol stack that is used to provide the TLPB services is called the *transport provider*. Figure 5 illustrates these concepts with some specific examples.

Figure 5 Examples of transport users and providers



When the address format and the services requested by a transport user are fully supported by a given transport provider, the user is said to be *native* with respect to the transport provider. For example, in Figure 5, the TCP/IP protocol stack (transport provider) is native to the sockets API processing layer (transport user) for applications that use IP addresses. When operating natively, transport users do not make TLPB calls. When the address format or a required transport service is not supported by the transport provider, the transport user is said to be *nonnative* with respect to that

transport provider. Transport users invoke the TLPB to operate nonnatively.

Transport users that use the same address format and request the same set of TLPB services are said to be matching transport users. Transport providers that use the same formats and protocols are said to be matching transport providers. The term *nonmatching* is defined conversely. The MPTN gateway allows transport providers to be nonmatching—allowing concatenation of networks running different protocols. However, transport users must be matching. For example, in Figure 5, whereas MPTN allows two Advanced Program-to-Program Communication (APPC) applications (or an APPC and a CPI-C application) to communicate with each other over any communications protocol, it does not allow a sockets application to communicate with an APPC application. The TLPB-based approach to the application independence problem raises the following questions, all of which are addressed by the MPTN architecture:

1. What (smallest) set of transport-layer services should be provided in the TLPB for it to be useful for all current APIs? The next subsection provides a high-level overview of the TLPB.
2. When a given transport-layer service of the TLPB is not natively available in the transport provider's protocol stack, a *protocol compensation* must be provided for that service. What is the set of protocol compensations associated with the TLPB? How large is that set? The subsection on generalized protocol compensations addresses those questions.
3. A distributed application, written to a given communications protocol, uses the address formats defined for that protocol. When such an application is run nonnatively over a different communications protocol with different address formats, an address used by a transport user to identify its partner must be mapped to an address that the transport provider can use to locate the node where that partner resides. What are the solutions to this problem? The MPTN architecture defines three different *address mapping* techniques to solve this problem. These are described in the subsection on address mapping.

Transport-layer services of the TLPB. To support existing applications written to a wide variety of APIs, the TLPB must provide all basic transport-layer services associated with currently available transport protocols. The services provided by the TLPB are summarized below.

Connection-oriented transport services. Connection-oriented transport services consist of *connection setup, data transfer, connection termination,* and *session outage notification*. A connection guarantees that all data arrive in order without duplication, loss, or corruption. All connections are assumed to be full-duplex, allowing each side to send and receive data, or to terminate the connection, independently of the partner.

The connection setup service is initiated by a TLPB request to set up a connection to a partner with a given address. The partner in turn typically makes TLPB requests to register its address and to declare itself as being ready to accept connection setup requests. When a connection is set up, connection data may be sent with the connection request. Figure 6 gives additional details on the MPTN connection setup.

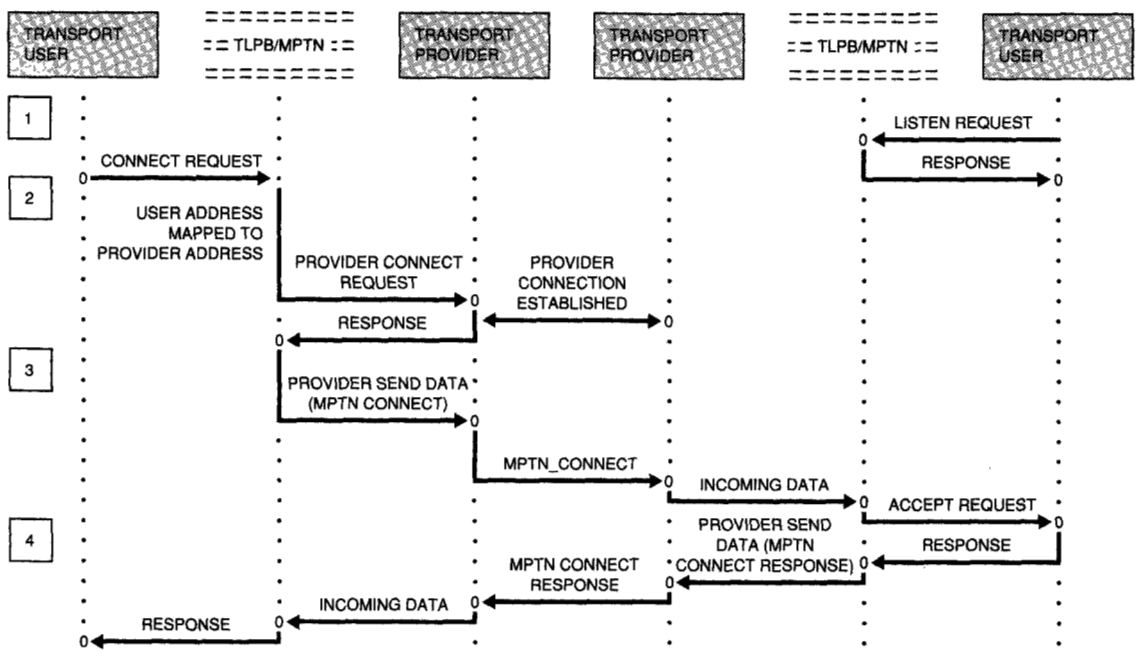
The TLPB service user may request that the connection be *stream-oriented* or *record-oriented*. A stream-oriented connection does not preserve the boundaries of the units in which data are sent. A record-oriented connection preserves record boundaries, i.e., every unit of data received corresponds to the data sent.

The TLPB data transfer service may be specified as *normal* or *expedited*. Normal data are subject to the flow control mechanism of the native transport. Expedited data bypass the native flow control. The TLPB service guarantees that expedited data will arrive at the partner *no later* than normal data sent after the expedited data. The TLPB optionally provides a marking service, wherein the expedited data are marked showing where the data are located in the data stream relative to the normal data.

A full-duplex connection can be terminated by either side. *Termination data* may be exchanged between the transport users when a connection is terminated. Two aspects of connection termination are supported by TLPB services:

1. A termination request may be classified as *simplex* or *duplex*. A simplex termination request is used to close only one end of the full-duplex connection (allowing the partner to continue to

Figure 6 MPTN protocol for connection establishment



ANNOTATION

1 The establishment of an MPTN connection between two MPTN access nodes begins when an application makes itself available, and another application requests the establishment of a connection. The transport user transforms the native connection setup operation to a TLPB operation.

2 The MPTN components map the address provided by the transport user to an address usable by the selected transport provider, and establish a transport provider connection with the partner MPTN access node using the transport provider address obtained from the mapping.

3 An MPTN-specific format is then sent as the first data packet on the provider connection. This format is processed by the MPTN components in the receiving node. This format contains, among other MPTN-specific information, any connection data sent by the transport user.

4 If the transport user on the receiving node processes the connection request without error, a positive response is returned to the initiating MPTN component, together with any connection reply data. If the connection reply data are processed successfully, the application is notified that a connection has been set up.

When the partner can only be reached via one or more MPTN gateways, the connection setup protocol described above is performed between the access node and the first gateway in the path. The gateway in turn proceeds to set up connections with the next gateway in the path if the partner is more than one hop away from it, or to a partner MPTN access node or native node in an adjacent network that it is attached to. This is described in more detail in the subsection "Interaction Between Gateways."

send data). A duplex termination request closes both ends of the connection simultaneously.

2. A termination request can also be either *abortive* or *orderly*. Abortive termination is used if data in the pipe need not be delivered before the connection is terminated. Orderly termination guarantees that data in the pipe will reach the partner before connection termination.

A transport user specifies both aspects of session termination—whether the termination is simplex/duplex, and abortive/orderly—thereby giving rise to four types of connection termination.

In session outage notification, when a full-duplex connection is set up using the TLPB, the MPTN mechanism notifies the transport user when the connection partner becomes unreachable. When it is detected that the partner is unreachable, the connection is terminated.

Connectionless transport services. Connectionless transport (or datagram) services only guarantee data integrity, i.e., *if* the data reach the partner, the data will not be corrupted. However, the data may be lost in transit, the same data may be delivered more than once, and data may not reach the receiver in the same order in which the data were sent. The TLPB supports two kinds of connectionless data transfer services: *unicast* and *multicast*.

A unicast datagram is sent to a single destination. A multicast datagram is sent to all members of a group, the destination address representing the name of the group that each member has joined. The unicast services of the TLPB allow the receiver of a unicast datagram to register its address and to declare that it is ready to receive datagrams sent to that address. The sender of the datagram makes a request to send a datagram to a given destination address. Multicast services allow a transport user to join a group with a given (multicast) address, and allow a transport user to send a single datagram to all members of a given group within a given subnetwork. A special case of multicast is *broadcast* services, which allows a single datagram to be sent to all transport users in a given subnetwork.

Generalized protocol compensations. Since the TLPB provides a diverse set of services to each transport user, most transport providers cannot provide all of the services that might be requested by a transport user. For each TLPB service that is absent in the native protocol, a compensation mechanism is defined in the architecture. The compensation enables a transport provider to simulate the given TLPB service, using existing features of the native protocol.

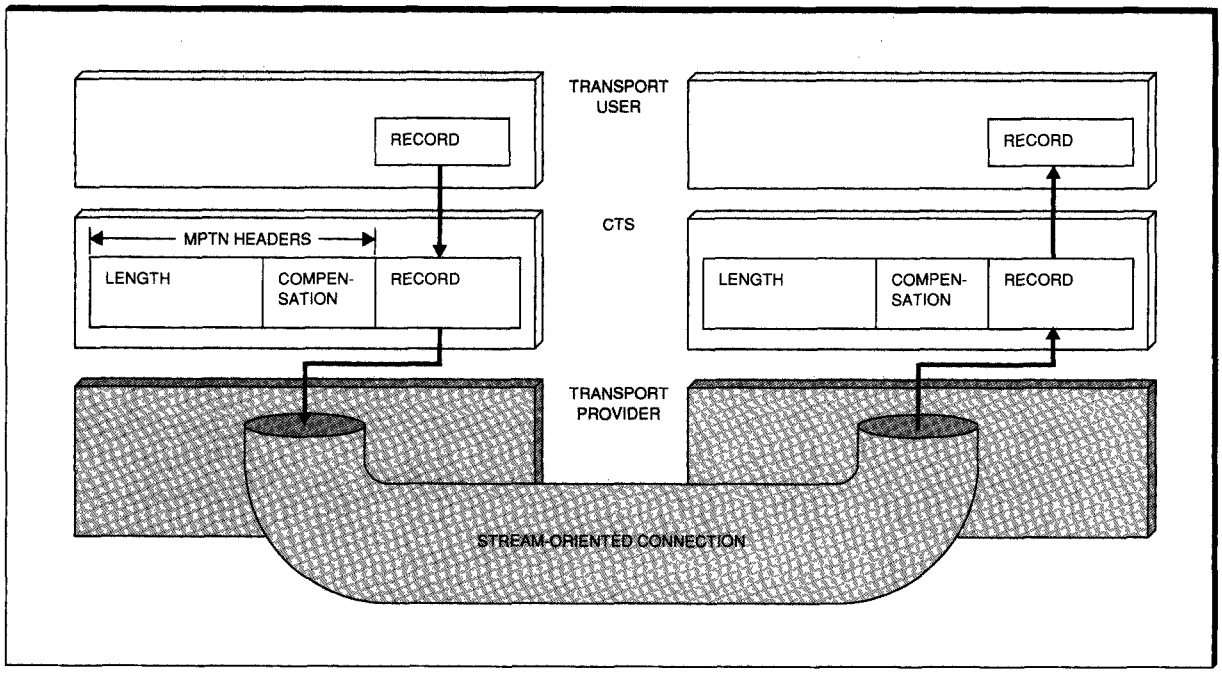
Since the TLPB provides a fixed set of services, only a limited number of compensations need to be defined in the architecture, one for each service of the TLPB. These compensation techniques can then be used in a potentially unlimited number of transport-user-transport-provider protocol combinations that may be implemented based on the architecture. This usage simplifies development of a family of products using the MPTN architecture. This approach should be contrasted with customized solutions such as RFC (Request for Comments) 1001/1002[9,10] for running NetBIOS applications over TCP/IP, and RFC 1006[11] for running OSI TP4 (Transport Protocol Class 4) applications over TCP/IP—where the issue of protocol compensations is implemented differently for each new pair of nonmatching transport user and provider pair.

An example of a simple compensation is sending record-oriented data on a connection.

Techniques for compensation. Some compensations are conceptually simple, requiring a short header to be inserted before the data by the MPTN component in the sending end. The header is interpreted by the receiving end and removed before the data are passed to the transport user. Other compensations are conceptually more complex, requiring the use of timers, parallel data paths, etc. Some illustrative examples are given below. More details about different compensations defined in the MPTN architecture are given in References 12, 13, and 14.

One example of a simple compensation is sending record-oriented data on a connection. When the transport user is record-oriented and the transport provider is stream-oriented, a four-byte length field and a one-byte compensation header identify the record boundaries on the data stream (see Figure 7). Another example is when both the transport user and transport provider are record-oriented, but the transport user sends a record that is larger than the provider allows, the record can be broken up into segments, and a one-byte

Figure 7 Simple compensation: Record-oriented user over stream-oriented provider



compensation header is sufficient to identify the segments.

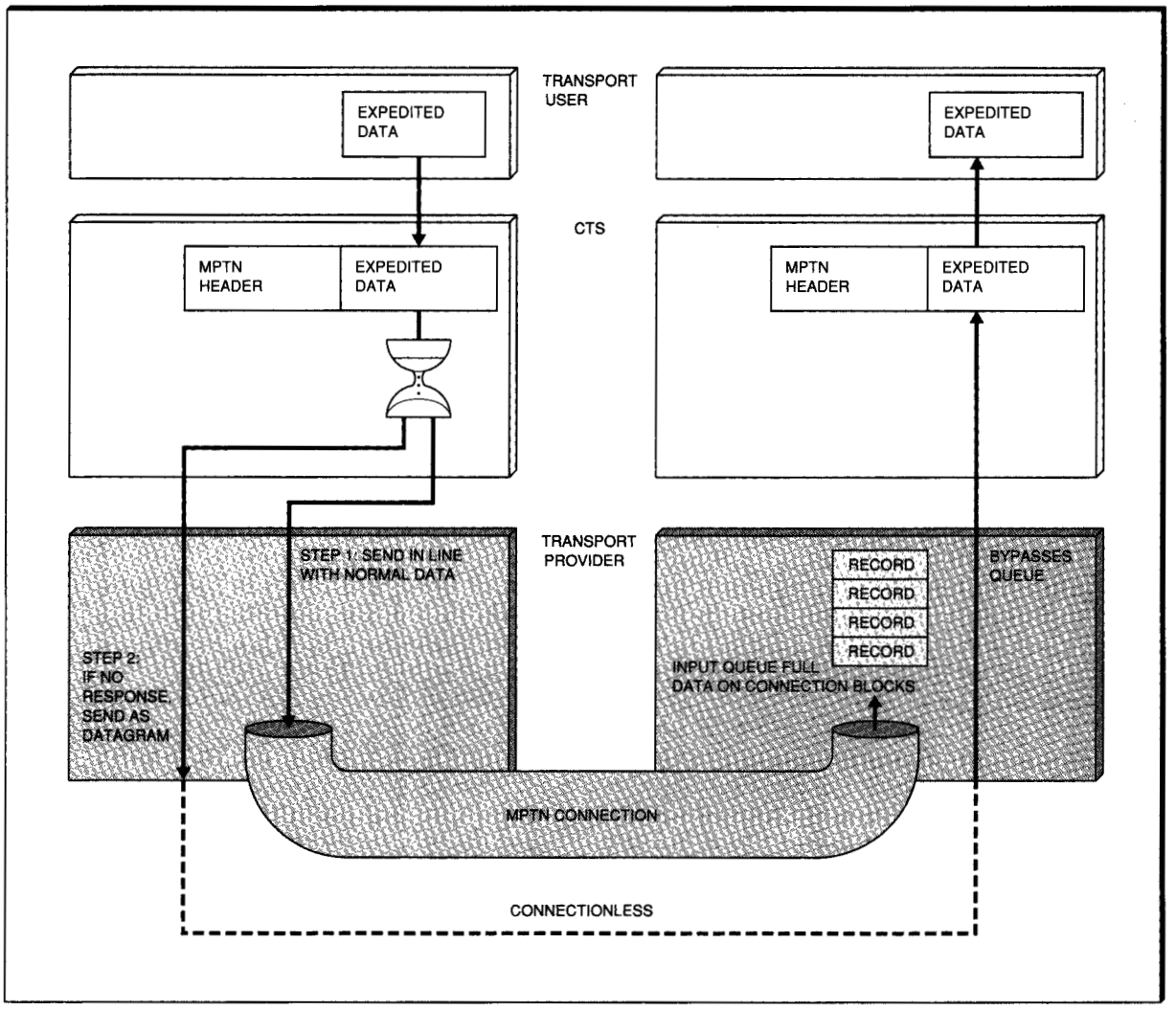
Connectionless data from the transport user, when sent using the connectionless services of a transport provider, also involve adding architecturally structured headers. In this case, the headers are more complex because each unit of connectionless data must contain the names of end points in the transport user's address format, enough state information to segment and reassemble the datagram if necessary, etc.

Some compensation mechanisms are quite complex, requiring implementation of sophisticated network features using whatever services the native transport protocol provides. Some of these compensations include expedited data, session outage notification, and connectionless data over a connection-oriented protocol.

Expedited data from the transport user should bypass queued data anywhere in the MPTN connection. When the transport provider does not provide the ability to send user-expedited data on the connection, the MPTN compensation mechanism defines a protocol for sending the expedited data using an alternate path. The expedited data are first sent on the connection (which is subject to congestion) with a header identifying the data as such. If the partner does not acknowledge the expedited data within a short period of time by sending an expedited-data-acknowledgment header, the data are sent again, this time using a special MPTN datagram called an out-of-band (OOB) datagram. An OOB datagram identifies the connection for which expedited data are being sent and a sequence number; its data portion consists of both the MPTN expedited data header and the data themselves. This ensures that the data can bypass any connection level flow control that is in effect. In either case, the MPTN components on the receiving end use the information in the OOB datagram to route the data around any queued data. See Figure 8.

Session outage notification is used to notify a transport user if the underlying transport provider connection fails. When the transport provider does not provide session outage notification natively, the MPTN components compensate by keeping track of the time since the last activity with the next node in the MPTN connection (which could be another

Figure 8 Complex compensation: Expedited data



access node or an MPTN gateway). If this time interval exceeds a configurable value, the MPTN components send a special MPTN datagram (called a keepalive datagram) to the next node in the connection. If that node responds, the interval timer is reset. However, if that node does not respond after repeated attempts, the transport user is informed that all sessions to that partner node have failed.

Finally, if the transport provider is only connection-oriented, datagrams themselves may have to be implemented using connections. It can be done using short-lived connections (one connection per datagram) or long-lived connections (multiple datagrams sent over the same connection).

Determining which compensations are necessary. There are slightly different procedures for determining which compensations will be needed for a given data transfer, depending on whether the transfer is connection-oriented or connectionless. For a connection-oriented transfer, the transport user that initiates the connection request (as well as the transport user that accepts the connection request) begins by specifying the set of TLPB services it will

require for the new connection. These services are called the transport user characteristics. When a TLPB connection request is issued, MPTN uses address mapping (see the next subsection) to choose a transport provider that can reach the requested partner. On the basis of the transport characteristics available from that transport provider, MPTN determines the compensations that will be needed to provide the required TLPB services. One of the first items sent in the MPTN connection setup message is this list of necessary compensations. If the partner does not verify in its response that the necessary compensations are available, the MPTN connection setup fails.

For connectionless data transfer, the final part of this procedure is eliminated. The transport user specifies required services, MPTN chooses a provider and determines the required compensations, and the datagram is sent. No checking is done to determine if the compensations needed are acceptable to the partner.

Address mapping. The address mapping problem arises because the MPTN framework allows distributed applications to run unmodified over nonnative transport protocols, and transport user addresses cannot be understood by nonnative transport providers. To enable nonnative communications, an address used by a distributed application to identify its partner, in a format supported by the protocol the application was written to, must be mapped to a transport provider address that identifies the node where that partner resides. The problem is conceptually similar to one that exists in LANs, where network-layer addresses must be mapped to MAC addresses in order to establish LAN connectivity (e.g., the IP Address Resolution Protocol[3]).

Most transport protocols structure addresses into two parts, a node address that uniquely identifies a node in a network (e.g., a fully-qualified logical unit [LU] name in SNA, an IP address in TCP/IP), and a local address that identifies an application within a node (e.g., a transaction program [TP] name in SNA, a port number in TCP/IP). To ensure scalability, all MPTN address mapping solutions map node addresses only.[15] A well-known transport provider local address is used to identify the MPTN function in the node (e.g., a well-known TP name for the SNA transport provider, a well-known port number for the TCP/IP transport provider). Complete transport user addresses flow on connection setup and datagram headers to enable the request to be routed to the correct transport user and application.

MPTN supports three methods for address mapping, ranging from a simple solution (in terms of implementation complexity) that has limited applicability, to a completely general but more complex solution. They are: algorithmic address mapping, use of a protocol-specific directory, and the MPTN address mapper. The advantages and limitations of each of these methods are described in the following subsections.

MPTN supports three methods for address mapping.

Algorithmic address mapping. In this approach, an algorithm is used to generate a transport provider address from a transport user address. In the node that is ready to accept a connection or datagram, the transport user address that is registered to MPTN is algorithmically converted to a transport provider address, which is then registered with the transport provider in a protocol-specific manner. When a connection or datagram needs to be routed to a given transport user address, the same algorithm is applied to obtain the corresponding transport provider node address, and the well-known local address is appended to it to generate the complete transport provider address of the partner node. Figure 9 shows an example of a sockets application that uses IP addresses. This IP address is mapped algorithmically to an SNA LU name by the MPTN component.

The advantages of this scheme are that it is fully distributed and does not require any additional network flows. However, this technique is only applicable if the address space of the transport provider is larger than that of the transport user. Furthermore, the transport provider protocol must be flexible enough to allow more than one address per node in a network, since the algorithmically

generated address will typically not be the same as the address that was originally allocated to that node. This condition rules out the use of the algorithmic address mapping technique for TCP/IP transport providers, for example, since a node in a TCP/IP network can only have one IP address per network adapter. The sockets over SNA products (see the section on AnyNet products) use algorithmic address mapping.

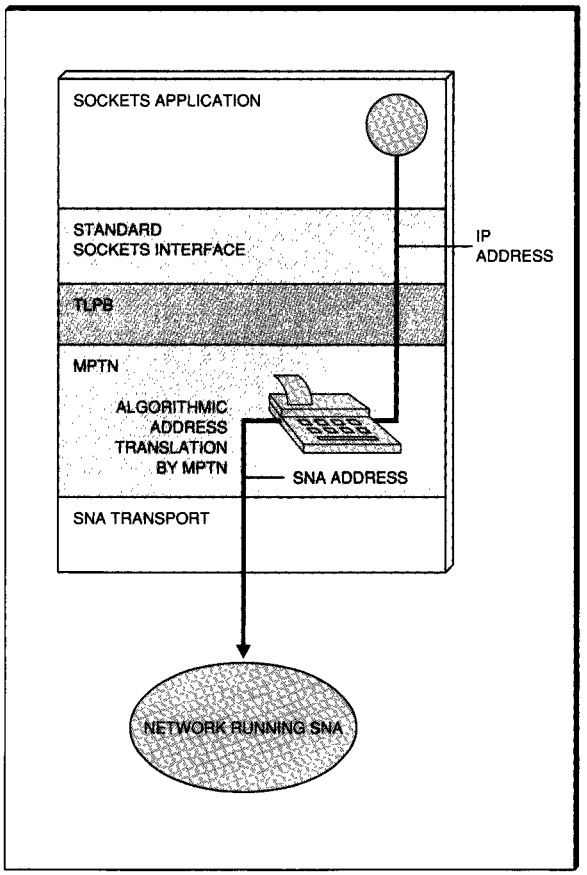
Protocol-specific directories. A transport provider directory that is flexible enough to accept registrations of nonnative names can be used to provide MPTN address mapping. A node that is ready to accept connections or datagrams registers its transport user address to MPTN. The mapping between that transport user address and the transport provider address of that node is then registered to the directory using transport-provider-specific flows. To establish a connection or forward a datagram to a transport user address, the directory must be queried in a protocol-specific manner in order to determine the corresponding transport provider address.

Figure 10 shows a network configuration with a protocol-specific directory used to map addresses. The only MPTN flows are between access nodes, but the connection setup requires flows from the access node to the protocol-specific directory. These flows are not formats designed according to the MPTN architecture but the formats required by the directory.

For example, the NetBIOS over SNA product (see section on AnyNet products) uses an SNA-specific directory for registering the mappings between NetBIOS and SNA (LU) names. Another example is the SNA over TCP/IP products (see section on AnyNet products) which use the TCP/IP Domain Name System (DNS) to define the mappings between SNA LU names and IP addresses. Each SNA LU name is converted to ASCII, put into the form LU.Netid, and prepended with a standard extension to yield a valid domain name. The mapping between the domain name and the IP address of the node where that LU resides is maintained in the DNS database. The above are current implementations; different address mapping techniques may be used as the products evolve.

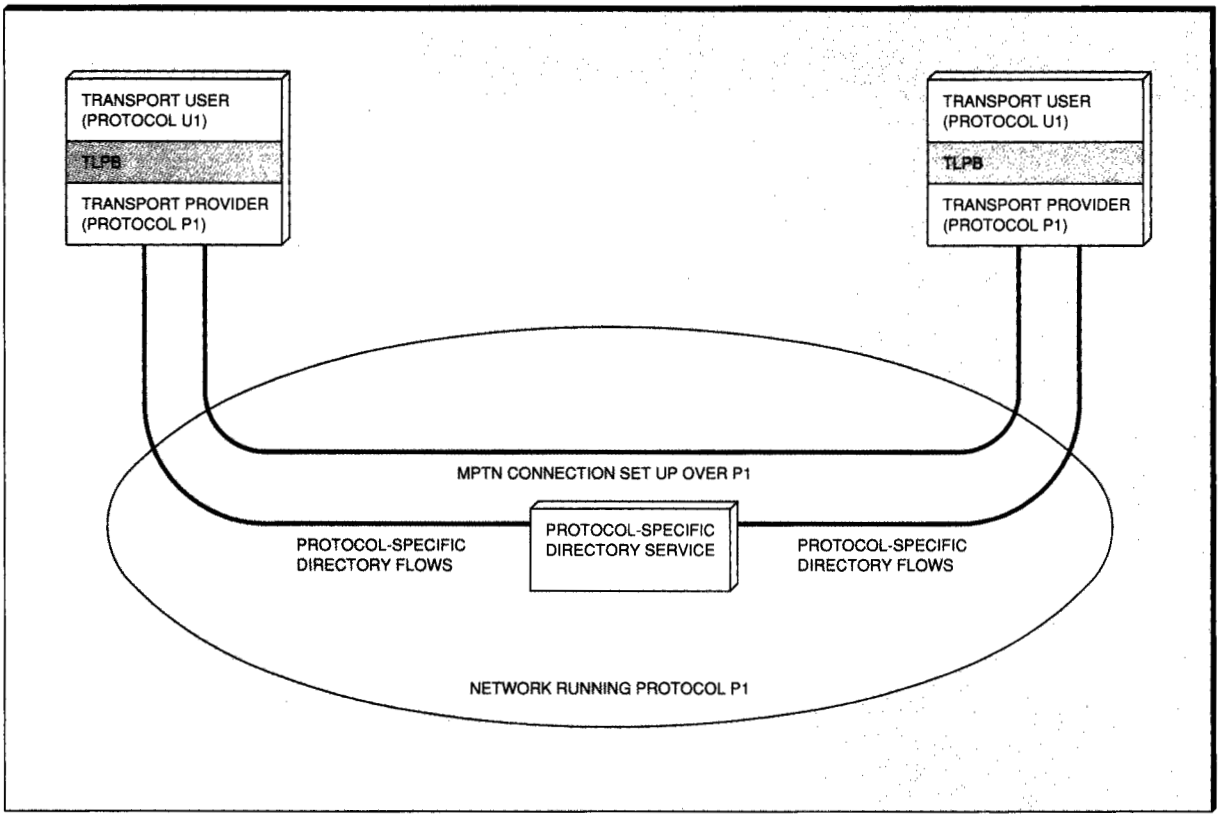
This scheme is more flexible than algorithmic mapping because, first, there are no restrictions on the relative length and syntax of the user and provider addresses. Second, no new protocols need to be defined for address mapping. The main drawback of this scheme is that it is only feasible in transport provider networks that support a sufficiently flexible directory service. Furthermore, if protocol-specific mechanisms are used, the address mapping solution will be different in each transport provider network. An additional drawback of this scheme is that certain directories do not allow dynamic updates and can be only statically configured by a network administrator (e.g., the DNS). With such a scheme, an address mapping for a transport user will exist even if the node where it resides is not currently active, and if an address is moved to another node, the configuration will have to be manually updated.

Figure 9 Example of algorithmic address mapping[12]



MPTN address mapper. The MPTN address mapper is a protocol-independent function that supports

Figure 10 MPTN access nodes and a protocol-specific directory [14]

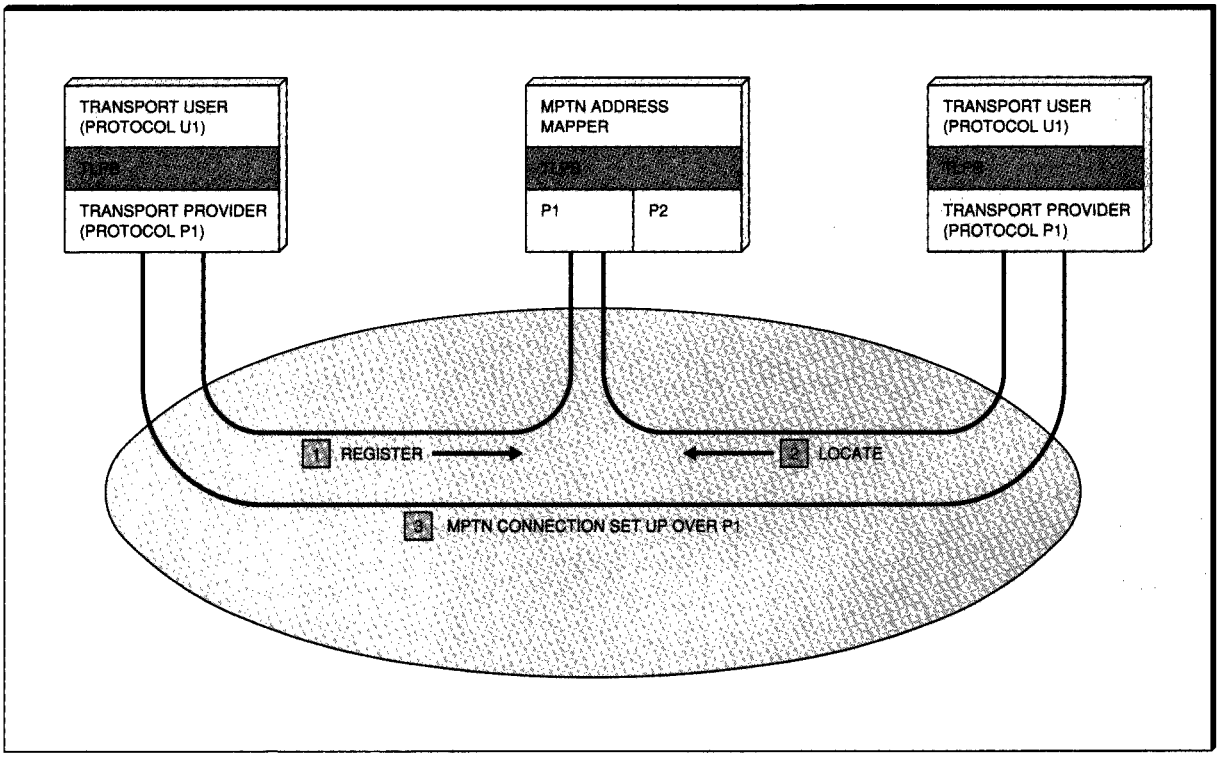


address resolution and dynamic name registrations. When a transport user address is registered by an application, that user address and the corresponding transport provider address are registered with the MPTN address mapper, using special MPTN network flows. To establish a connection or forward a datagram to a specified user address, the address mapper is queried in order to determine the corresponding transport provider address. Figure 11 shows a network configuration with an address mapper. The flows between the access nodes and the address mapper are flows specified according to the MPTN architecture.

The MPTN address mapper provides the most flexible method for address mapping. It can support mappings between any type of user and provider addresses, and it is independent of the transport provider network(s). Therefore, regardless of the mix of transport users and providers in use in a particular environment, the MPTN address mapper provides a single, uniform solution for address mapping. Furthermore, the MPTN address mapping protocols are dynamic so that no static definitions are required, and the address mappings will correctly reflect the availability and location of applications. Finally, the address mapper allows wild card registrations, which an MPTN gateway can exploit to advertise reachability to a set of end points, based on a network identifier that is a common part of a group of node addresses.

For fault-tolerance or performance reasons, more than one MPTN address mapper may be used in a single transport provider network. The address mappers register mappings in a common distributed database so that they all provide a current and consistent view of the user-to-provider address mappings. As long as at least one address mapper continues to function, the address mapper service will not be interrupted. For more details on the address mapper, see Reference 16.

Figure 11 MPTN access nodes and the MPTN address mapper [19]



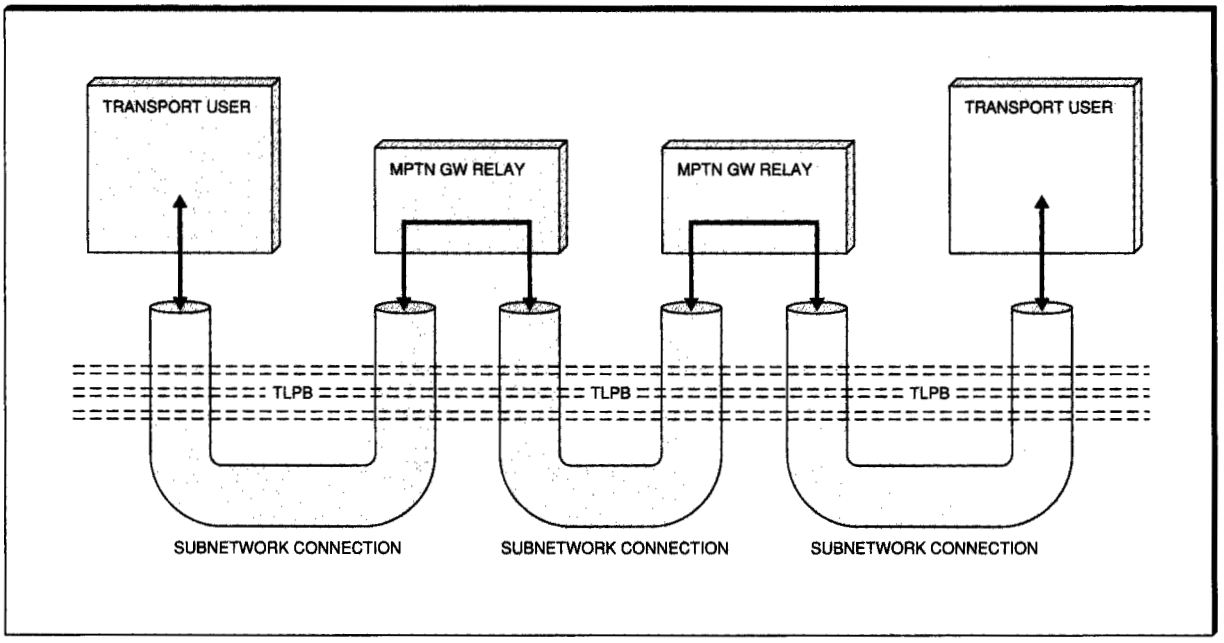
MPTN transport gateway. An MPTN network consists of individual *subnetworks*, each of which runs a single transport protocol. (Two or more subnetworks may share the same physical networking hardware). An MPTN gateway exists at the boundary between two subnetworks running different protocols and allows communication to take place across them. MPTN gateways perform subnetwork concatenation, thereby facilitating the construction of a global MPTN network from a collection of heterogeneous subnetworks.

An end-to-end MPTN connection consists of a series of subnetwork connections concatenated by relays in MPTN gateways, as depicted in Figure 12. Gateways relay user data between related subnetwork connections. Datagrams are relayed end-to-end in a similar fashion. MPTN gateways provide network concatenation at the transport layer, which implies that the subnetwork transport and lower-layer protocols are terminated at the gateway.

Gateway configurations. An MPTN transport gateway can be accessed by end systems that have no MPTN function (native nodes), as well as by MPTN access nodes. This access allows applications running natively (e.g., a sockets application running on TCP/IP) to interoperate with a peer running nonnatively (e.g., a sockets application running on an SNA network). MPTN gateways allow native nodes to access the global MPTN network. MPTN gateways can be used in very flexible ways to concatenate subnetworks. Two configurations have been found to be very fundamental in customer environments, and the rest of the discussion on gateways will focus on these configurations.

The most basic configuration consists of a single gateway that concatenates one subnetwork containing native nodes and another subnetwork containing access nodes. One example of such a configuration would be a gateway connecting a subnetwork consisting of native nodes running TCP/IP applications and an SNA subnetwork running sock-

Figure 12 An MPTN connection comprised of concatenated subnetwork connection segments



ets over SNA access nodes. A minor variation of this configuration consists of adding more than one gateway between the subnetworks (*parallel gateways*) for load balancing.

Combining two such gateways back-to-back gives rise to a configuration where two subnetworks running the same native protocol can communicate with each other through a third (backbone) subnetwork running a different protocol. If the subnetworks at the edges have access nodes, native protocols are used in the backbone. Conversely, if the subnetworks at the edges have native nodes, MPTN protocols are used in the backbone.

Interactions with native nodes. When an MPTN gateway is attached to a subnetwork with native nodes, it must appear to be a full-function native node itself, with the ability to participate in native connection-oriented and connectionless protocols. In addition, the gateway must also be able to participate in native routing protocols of the subnetwork in order to advertise its ability to reach native end systems to access nodes and other gateways, and vice versa.

MPTN gateways must be able to participate in two fundamental types of native routing protocols:

- Routing-table-based protocols—These protocols continuously distribute *reachability information*, which includes the path and characteristics of that path, to a destination or set of destination end systems. When a route to a particular destination is required, the next hop to reach the destination must already be known at each routing node along the path. Otherwise, the destination is assumed to be unreachable. Examples of such protocols are route table update protocols used in IP routers,[3] such as Routing Information Protocol (RIP), Exterior Gateway Protocol (EGP), and Open Shortest Path First (OSPF).
- Search-based protocols—These protocols only perform routing functions when a path to a particular destination is required. At that time, a broadcast search for the required destination is generated in order to determine how to route to that address. Advanced Peer-to-Peer Networking*[1] (APPN*) and NetBIOS[4] use such a routing protocol.

To support routing-table-based protocols, the MPTN gateway must present the appearance of a router for that protocol to the native subnetwork. That is, it must advertise within the native subnetwork its ability to reach end systems outside of that subnetwork, and it must also be able to learn what

systems can be reached in the native subnetwork. With this capability, the native protocol will be able to correctly route connection requests and datagrams destined for nodes outside its subnetwork to the gateway. Conversely, requests that originate outside the native subnetwork, but are destined for a node in the subnetwork, can be routed to the gateway.

For search-based routing protocols, the MPTN gateway must be able to receive all native searches and respond when the particular resource exists on a different subnetwork in the MPTN network. It sends a positive response to the search if it can reach the address, which will cause subsequent connections or datagrams to that address to be routed to the gateway by the native protocols. A gateway can also initiate a search in a native subnetwork when a connection request or datagram is sent to it from another subnetwork.

Interaction with access nodes. MPTN gateways interact with access nodes using the same MPTN protocols that access nodes use to communicate with each other, for both connection-oriented and connectionless communication. For example, when an access node initiates a connection request with a partner that is in a different subnetwork, the address mapping mechanism returns the transport provider address of the first gateway in the path to that partner. The access node receives no indication as to whether the transport provider address returned belongs to a gateway or an access node, and no such indication is required. The access node where the connection request originates sets up an underlying transport connection to the gateway, and flows an MPTN-specific format as the first unit of data on that connection, as described earlier.

Interaction between gateways. When more than one MPTN gateway is in the path between the transport users that want to communicate, each pair of adjacent gateways consecutively perform the same operations for connection establishment and datagram forwarding as previously described. The only difference arises for connection setup in that the gateways withhold the positive MPTN connection response until all intermediate connections have been established.

MPTN protocol for connection establishment through gateways. Consider the configuration consisting of a single MPTN gateway between two subnetworks, with native nodes on one subnetwork and access nodes on the other. When the connection request originates in a native node and is destined for an access node in the other subnetwork, the native node learns that the connection request must be forwarded to the gateway because of its participation in the native routing protocols of that subnetwork. When the gateway receives the native connection request, it uses some address mapping technique to discover the transport provider address of the access node where the partner resides. The MPTN connection setup protocol described earlier is then used to set up the nonnative hop of the MPTN connection between the gateway and the access node.

When the connection request originates in an access node and is destined for a native node, the access node first uses some address mapping technique to discover the transport provider address of the gateway to which the connection request must be forwarded. For example, when using a protocol-specific directory such as the DNS in the SNA over TCP/IP product, the LU name of a partner that resides in a native SNA node is mapped to the IP address of the gateway concatenating the SNA and TCP/IP subnetworks. When using the MPTN address mapper, the gateway registers a wild card entry that advertises its ability to reach all LUs with a given SNA netid, on the subnetwork with native SNA nodes. Once the gateway address is found, the access node uses the MPTN connection setup protocol described previously to set up the first hop of the MPTN connection to the gateway. When the gateway receives the MPTN connection setup request, it uses native protocols to set up the native hop of the MPTN connection with the partner node.

In a back-to-back gateway configuration consisting of two subnetworks at the edges of a backbone, the connection setup protocol is very similar to the above except that the gateway involved in the first hop of the MPTN connection sets up the next hop of the connection to another gateway on the backbone, instead of to a native node or an access node. When native protocols are used in the backbone, the routing protocol used by the first gateway points it to the second gateway as the next hop of the connection. When MPTN protocols are used in the backbone, the address mapping technique being used will point the first gateway to the second. The second gateway sets up the last hop of the MPTN connection.

Network management. Although MPTN makes it possible to reduce the number of transport provider protocols, since all transport users can communicate using any chosen protocol, at least one existing protocol must always be present to support communications between any two nodes. The network management protocols of the selected transport provider are available for native network management support. The presence of MPTN has no impact on transport user protocols and applications, and therefore, their existing network management functions continue to operate there as well. MPTN entities (access nodes, address mappers, and gateways) can utilize either the transport user's or the transport provider's network management protocols to report errors. Although the existing scheme is sufficient for basic MPTN problem reporting and determination, it does not provide the network administrator with a view of the associations between connections from the transport user's point of view (based on transport user addresses) and from the transport provider's point of view (based on the transport provider addresses). Provision for tools to allow network administrators to view and manage the association between user and provider resources, as well as gateway connections and gateway topology, is an important future extension for the MPTN architecture. It will be especially useful for problem determination in the presence of gateways since, in that case, an end-to-end connection is supported by more than one transport provider connection.

MPTN provides the ability to eliminate parallel communications protocol code from workstations.

Applying MPTN solutions. With use of MPTN, various end-user networking problems can be solved in new and efficient ways. The following subsection describes how MPTN can be employed in three network configurations to solve six customer problem scenarios.

Single network configuration. MPTN in a single network can allow a customer to implement the following three solutions:

Introduction of a new application that uses a different protocol. A customer with a network of one type may wish to use a commonly available application that relies on a different communications protocol. For instance, there may be an SNA network with OS/2* (Operating System/2*), DOS/Windows**, or AIX* (Advanced Interactive Executive*) workstation clients, and MVS/ESA* (Multiple Virtual Storage/Enterprise Systems Architecture) or AS/400* (Application System/400*) servers. The customer may want to use a file server application that is written to the sockets programming interface that normally uses TCP/IP as its communications protocol. The server application can run on a machine that has sockets over SNA access node support installed (see the section on AnyNet products). It will run the existing server code, written to the sockets programming interface, without modification. If the various workstations running OS/2, Windows, or AIX install sockets over SNA support, they can access the file server as sockets application clients. The SNA network does not need to be modified in order to support this new application, and existing NetView* services can be used to manage the network. It is also not necessary to install new hardware to support this configuration.

Reduction of parallel networks. MPTN provides the ability to eliminate parallel communications protocol code from workstations. It eliminates the need to maintain and administer multiple networks. For instance, a single SNA network can provide transport services for SNA, NetBIOS, and sockets applications. OS/2 workstations can install the SNA feature of Communications Manager* providing SNA host access and emulator support. NetBIOS over SNA access node support (see the section on AnyNet products) will allow applications written to the NetBEUI interface, e.g., LAN Requester* and LAN Server*, to interoperate. Sockets over SNA will allow sockets applications to run in the same workstation. The three different types of applications can each efficiently make use of the rich networking capabilities of the underlying SNA network. Problems that can be caused by running different transport protocols on the same LAN, e.g., inequitable or inefficient use of the available bandwidth caused by different flow control mechanisms, or the use of broadcast, are avoided in this solution.

Preserving existing applications while changing the network. It may be desirable to gradually replace an existing network with another while maintaining support for existing applications. As one example, this situation may occur because a merger between two companies results in a consolidation of different networks. For instance, in order to convert an underlying SNA network to TCP/IP, it would be sufficient to install SNA over TCP/IP access node software (see the section on AnyNet products) on each node on the SNA network. Once the conversion is complete, only the TCP/IP network is required.

It is possible to incrementally upgrade each node without losing connectivity to other SNA nodes during the conversion phase by using an SNA over TCP/IP gateway. The gateway can support native SNA nodes on one side and SNA over TCP/IP access nodes on the other. When the first SNA node installs SNA over TCP/IP access node software, its connectivity with native SNA nodes is maintained through the SNA over TCP/IP gateway.

Interconnecting networks of dissimilar type. Below are presented several examples in which different types of networks are connected.

Extending the reach from one network type to another. A subnetwork of access nodes that is attached to a native network through an MPTN gateway can allow a customer to interoperate between two different networks. This situation can be seen as an extension of the problem described above—of introducing an application that uses a different transport type. In this case the customer needs to communicate with applications on another subnetwork running a different communications protocol. For example, an MPTN gateway will allow a customer's applications, written to the sockets programming interface running on an SNA network, to communicate with similar applications on a TCP/IP network. This configuration can give sockets applications running on SNA nodes access to the services of the Internet, such as databases and e-mail.

Merging two networks. When two companies with different network types merge, they may decide not to eliminate either network. An MPTN-based solution can help. For instance, a company with an SNA network may merge with a company that uses TCP/IP. It is likely that users of the TCP/IP network will want to use some of the applications that the company with the SNA network uses, e.g., to access SNA application databases. To enable this usage, it is only necessary to install SNA over TCP/IP access node software on nodes on the TCP/IP network that want to access SNA-based database servers, and to install an SNA over TCP/IP gateway to interconnect the two networks. Similarly, if employees of the company with the SNA network want to use TCP/IP applications, they can install sockets over SNA access node software in their SNA machines and interconnect the SNA and TCP/IP networks using a sockets over SNA gateway.

LANs interconnected over a backbone network. When a LAN protocol must be routed over a different backbone protocol, MPTN can provide a solution that requires no new hardware. For instance, in order to route TCP/IP traffic over an SNA wide area backbone network, back-to-back sockets over SNA gateways can be utilized. The source and destination nodes flow native TCP/IP protocols and require no modifications. In addition to providing connectivity between TCP/IP LANs, MPTN will transparently provide connectivity to TCP/IP applications on the SNA network where the workstations have sockets over SNA installed.

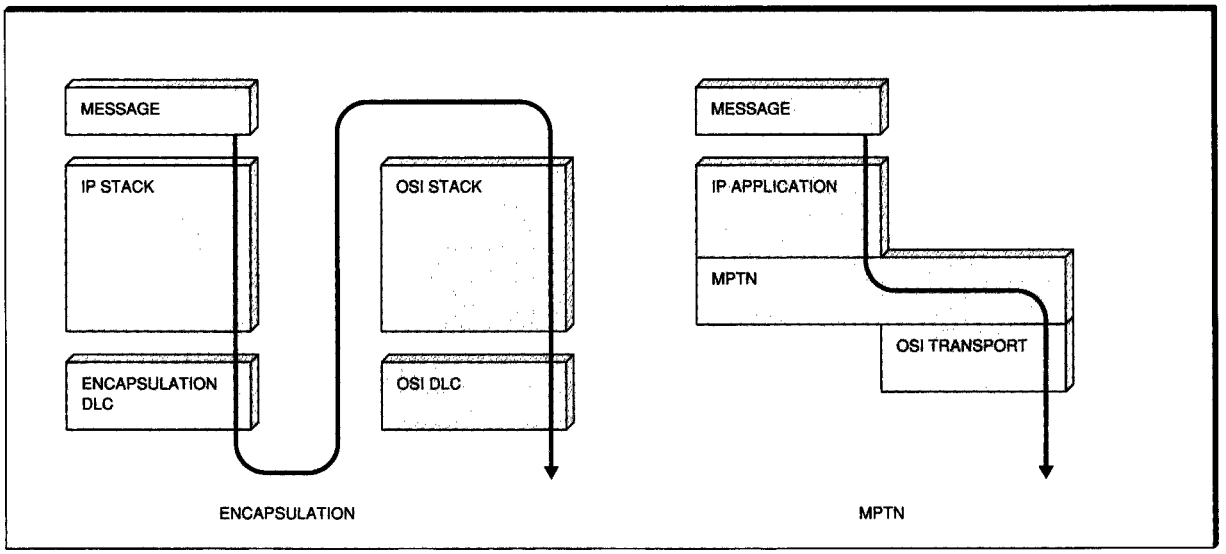
Comparison of MPTN with other solutions

In this section, other multiprotocol techniques that can be used as partial alternatives to the MPTN solutions are examined. Some are optimized to particular environments, some are simpler to roll out but more complex to manage, and others are more expensive. Network connectivity problems that MPTN is not designed to address, such as communication between unlike applications, may be solved by some of these other solutions. The following four techniques are described, and their advantages and disadvantages are compared: (1) Encapsulation, (2) multiprotocol routers, (3) application gateways, and (4) middleware.

MPTN can coexist with these and other network connectivity techniques as a customer's communication system evolves. For example, as an organization moves toward a single network backbone protocol, a mix of MPTN access nodes and application gateways might be used on the periphery of the network, whereas a mix of MPTN gateway nodes and multiprotocol routers may be used within the backbone.

Encapsulation. Encapsulation is a general, widely used technique that involves executing the user protocol stack, then taking the resulting packets

Figure 13 Encapsulation architecture comparison with MPTN



and treating them as user data for the provider stack, thus wrapping the user data and its protocol headers in the provider protocol[17] (see Figure 13). Encapsulation is sometimes known as tunneling if it takes place at the lower layers.

The primary advantage of encapsulation is that it is conceptually simple. The application protocol can be wrapped in the transport protocol, and the transport protocol is unaware of the contents of the encapsulated message. At the destination the wrap can be taken off, and the application protocol that is left can be interpreted by the application, which is also unaware that a different protocol was used to transport the packet from one node to another.

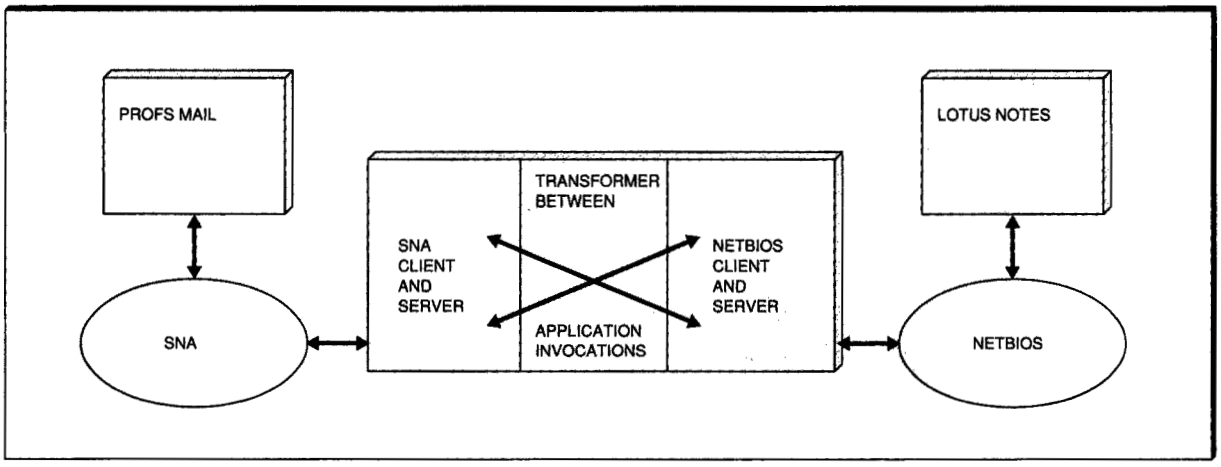
There are some disadvantages to the encapsulation technique. It may spawn unnecessary control traffic unless filtering is implemented. If filtering is built into the encapsulation, it must be done uniquely for each transport, which may involve some complex administration. Costs are increased because intelligent filtering requires a thorough understanding of the encapsulated traffic. Encapsulation requires complete execution of both protocol stacks. This means that both networks must be managed separately and both networks must be configured completely. Two sets of control flows have to be interpreted and executed at each routing point.

Many of the well-known problems of encapsulation are solved through the techniques used in MPTN. MPTN understands each packet instead of blindly propagating data link level packets without any understanding of their contents. Therefore, network storms are avoided. Configuration and management are simpler in the MPTN environment. Only the transport provider network must be configured completely. Although the transport user needs to be installed, only the installation parameters pertinent to the layers above the transport layer are required. In most cases, integrated management can be applied across the multiple protocols.

Multiprotocol routers. Multiprotocol routers are hardware boxes which, in addition to providing routing services for various protocols, also enable transmission of data for one or more communications protocols over a backbone network running a different protocol. This transmission is commonly done through an encapsulation technique[18] or through protocol translation for each pair of protocols in the router.

Routers are a convenient and efficient multiprotocol solution whose system design and hardware is optimized toward the routing function. They support existing applications without change and can be incorporated in production environments without disruption of ongoing operations. There is lim-

Figure 14 Transformations in the application gateway



ited interoperability between different vendors. There is also very limited interoperability, or crosstalk, between the various communications protocols routed by a multiprotocol router. Limited interoperability makes it difficult to guarantee any service level since there is no way to apply resource allocation between the various protocols. Also, a router satisfies the requirement to go from a network using one protocol through another network using a different protocol, to a third network using the same protocol as the first (A-B-A). But the router will not carry network traffic from network A through backbone network B to a destination network C (A-B-C).

MPTN solves the same problems that routers do, by providing a less expensive software approach to multiprotocol connectivity. The compensations prevent the clashes between dissimilar protocols that are multiplexed over a link, since they simulate record delimiting, flow control, and other mechanisms required by the application. With use of the MPTN gateway, traffic can be routed from an originating network through a backbone to a different destination network (A-B-C), where the MPTN access node will perform the necessary compensations for an A-type application to exist on a C-type transport network.

Application gateways. Another solution in multiprotocol environments is the use of application gateways. As shown in Figure 14, an application gateway is an application-specific solution that connects two or more networks and translates data and messages between applications performing the same general function. One example of an application gateway is a mail gateway that allows a mailer running PROFS* (Professional Office System), originally designed to run on SNA transport networks, to communicate with a mailer running Lotus Notes**, which runs over NetBIOS. There are many mature application gateways, and interoperability is possible between some pairs.

There are, however, some disadvantages to the use of application gateways. Each application gateway must be tailored to the pair of applications it interconnects, giving rise to a scalability problem. Writing an application gateway from scratch requires detailed knowledge of the applications on both ends of the exchange. Typically the application gateway code is more complex than either of the "gatewayed" applications, because all the nuances and error conditions of one application have to be taken care of in the gateway itself. In addition, application gateways exist for a very limited number of applications today, giving limited usability of application gateways to solve customers' multiprotocol problems.

The MPTN solution does not allow dissimilar applications to communicate with one another and, therefore, does not provide the kind of function that an application gateway can provide. However, since MPTN allows the choice of a single application optimized to the user's environment irrespective of the communications protocol that is being run, the use of different applications intended for

specific protocols can be avoided. For example, whereas an application gateway will allow a PROFS mailer to communicate with a Lotus Notes mailer, an MPTN solution will allow users to run either PROFS or Lotus Notes on all networks, irrespective of the underlying communications protocol.

Middleware. Middleware solutions include a product-specific API that can use various transport-layer protocols. This product-specific API provides a consistent interface independent of communications support or operating system dependencies. This independence allows middleware to run on different networks. To use middleware, applications must be written to use the product-specific API as defined by the middleware vendor. Examples of middleware solutions include message queuing interface (MQI),[19] remote procedure call (RPC),[20] and Transport Abstraction Conversion Toolkit (TACT).[21]

The primary consideration in the use of a middleware solution in a multiprotocol environment is the fact that all existing applications have to be rewritten to use the product-specific middleware API. This task can be rather daunting, especially given the enormous number of existing applications, many of which are critical applications that have been used without change for quite a long time.

An MPTN solution has the advantage of applications running unchanged over multiple transport protocols since the applications do not have to be rewritten. MPTN uses functions at the transport layer to insulate the source and destination application programs from knowledge of the transport provider protocols, and thus applications continue running as if they are on their native transport protocol.

AnyNet products

AnyNet is the name of a family of access node and gateway products that are based on the MPTN architecture. It is a family of connectivity software products designed to allow network managers to choose the applications their businesses need, regardless of what communications protocol is used by the businesses in their central or remote sites. AnyNet is available on OS/2, MVS, AIX, OS/400* (Operating System/400*), and Windows environments.

From the beginning, the AnyNet family has been outside the traditional IBM mold. The initial delivery consisted of four parts, sockets over SNA and APPC over TCP/IP (LU 6.2 traffic only), on both MVS and OS/2, with each piece being built by small teams responsible for design, code, and unit and functional test. The complete cycle, from start through general availability, was approximately 15 months. Development cycles since the first release have continued to shrink, with new combinations and new platforms continuously being added. The shrinking cycle time is a result of both the dedication and capabilities of the development team and the architecture design. The MPTN architecture facilitates reuse through its general design and the common compensations.

Two different models have been used for development of AnyNet code. Most of the products have been built from a general structure that mirrors the architectural concepts of MPTN with transport users and providers that can be added to the structure. In other cases, however, it was decided to start with existing API code and to grow that library with the MPTN function beneath it.

The first product started in the AnyNet family was sockets over SNA. With a common sockets library available on a number of platforms, that library became a natural focal point for code development. The availability of the library enabled production of a stand-alone product independent of the TCP/IP product and portable to diverse platforms (in this case, MVS and OS/2). Based on this structure, the product has been further extended to support additional transport providers for protocols such as NetBIOS and IPX. Although the architectural structure has not been employed in this product, the advantages of common compensations have still accrued, making new transports considerably easier to build.

The SNA over TCP/IP product started with a much different set of problems and constraints. It did not begin with a common SNA code base that had applicability on multiple platforms or that could be included as part of the AnyNet product. For this combination, the transport user was instead a very large, complex product in itself. One of the goals of MPTN was to be able to take advantage of all of its upper-layer functions and APIs to support its complex set of existing applications. In this environment, an independent MPTN structure, which could be used by the SNA upper layers with minimal changes to that code, was needed.

Table 1 SNA over TCP/IP products

Platform	Product Name	Support	Availability
MVS	AnyNet feature of VTAM Version 4 Release 2 for MVS/ESA	Access node and gateway support for all SNA applications	Available since June 24, 1994
OS/2	AnyNet/2 Version 2.1	Access node support for all SNA applications	Available since July 31, 1995
OS/2	AnyNet SNA over TCP/IP gateway for OS/2	Gateway support for all SNA applications	Available since December 30, 1994
AIX	AnyNet/6000 APPC over TCP/IP feature of AIX SNA Server/6000 Version 2.1.1	Access node support for APPC (independent LU 6.2) applications	Available since July 29, 1994
DOS/Windows	AnyNet APPC over TCP/IP for Windows	Access node support for APPC (independent LU 6.2) applications	Available in second quarter of 1995
OS/400	AnyNet/400 built-in feature of OS/400 Version 3 Release 1.0	Access node support for APPC (independent LU 6.2) applications	Available since November 25, 1994

With this information as a starting point, the decision was to build a general structure that would allow the addition of new transport users and providers independently and with as much code reuse as possible. This code base and model are the basis for a large number of the AnyNet products. The first product built on this basis was the APPC over TCP/IP product on OS/2. The next product built was NetBIOS over SNA on OS/2. Once both products were running, they were both installed on the same machine, and within three days a third protocol combination, NetBIOS over IP, was demonstrated. (The last has not become a product because of business decisions.) The code was then "ported" to other platforms with very different SNA implementations. Sockets over SNA has been built on this structure, and IPX has been added as a transport provider. The structure has proven itself to provide flexibility and growth so that additions become easier and bring forward more value as the library grows.

Both code libraries (the extended sockets library and the general MPTN structure) have been extended to include gateway function, and again the generality of the architecture is providing additional options and large amounts of reuse.

The proof of the modularity of the architecture and design and the effectiveness of these methods can be seen in the product matrices shown in Tables 1, 2, and 3.

Customer experience. Customers with diverse networks are using MPTN-based AnyNet products to solve application networking problems in various ways. The types of accounts using AnyNet reflect all key industries such as banking, finance, manufacturing, retail, and utilities.

AnyNet products are attractive to customers who:

- Have SNA application solutions that they want to extend to TCP/IP network end users, for example, Customer Information Control System* (CICS*), DATABASE2* (DB2*), Information Management System* (IMS*), NetView Distribution Manager, Distributed Console Access Facility (DCAF), and Time Sharing Option (TSO).
- Are interested in adding support for sockets applications on SNA networks, for example, Distributed Computing Environment (DCE**), Distributed System Object Model (DSOM), File Transfer Protocol (FTP), AIX PS/2 Network File System** (NFS**), packet internet groper (PING), Simple Network Management Protocol (SNMP), Telnet (the Internet standard protocol for remote terminal connection service), and X Windows System**
- Are interested in adding support for NetBIOS ap-

Table 2 Sockets over SNA products

Platform	Product Name	Support	Availability
MVS	AnyNet feature of VTAM Version 4 Release 2 for MVS/ESA	Access node support for sockets applications	Available since June 24, 1994
OS/2	AnyNet/2 Version 2.1	Access node support for sockets applications	Available since July 31, 1995
OS/2	AnyNet Sockets over SNA gateway for OS/2 Version 1.1	Gateway support for sockets applications	Available since June 24, 1994
AIX	AnyNet/6000 Sockets over SNA feature of AIX SNA Server/6000 Version 2.1.1	Access node support for sockets applications	Available since January 27, 1995
DOS/Windows	AnyNet Sockets over SNA for Windows	Access node support for sockets applications	In development
OS/400	AnyNet/400 built-in feature of OS/400 Version 3 Release 1.0	Access node support for sockets applications	Available since November 25, 1994

Table 3 Products supporting other protocol combinations

Platform	Product Name	Support	Availability
OS/2	AnyNet/2 NetBEUI over SNA	Access node support for NetBIOS applications over SNA	Available since July 29, 1994
OS/2	AnyNet IPX over SNA gateway for OS/2	Gateway support for IPX applications over SNA	Available since December 30, 1994
OS/2	AnyNet for OS/2 Version 2.1	Access node support for sockets applications over NetBIOS	July 31, 1995
OS/2	AnyNet for OS/2 Version 2.1	Access node support for sockets applications over IPX	July 31, 1995

plications on SNA networks (e.g., Lotus Notes, cc:Mail**, IBM LAN Requester and LAN Server, IBM Time and Place/2*)
- Want to allow remote branch locations to be managed over an existing SNA network at the customer central site
- Want to consolidate or change network backbones

The following experiences of some customers reflect how AnyNet products have helped solve real-world networking problems.

Pacific Bell. At Pacific Bell, AnyNet/2* APPC over TCP/IP satisfies the need to use APPC applications across the TCP/IP backbone network, without any change to the application.

Pacific Bell, a business with branch offices spread over thousands of miles, depends on its information network to consolidate vital operations and financial data. To control hardware, line, and network management costs, the network service providers have elected to standardize on a single communications protocol, TCP/IP. The local area networks use TCP/IP, and a TCP/IP backbone is in place for host communications.

AnyNet APPC over TCP/IP allowed Pacific Bell to select the best billing application for their business

offices without worrying about network compatibility. Now they have 14 servers and over 100 users accessing APPC applications over the TCP/IP backbone.

Canada Trust. Canada Trust, a retail bank in London, Ontario, uses the AnyNet/2 Sockets over SNA Gateway to allow SNMP-based hubs to be managed with use of NetView/6000* despite the 400 branches of the bank being connected by using SNA subarea and APPN.

The AnyNet/2 Sockets over SNA Gateway seamlessly integrates with the existing Communication Manager/2 (CM/2) gateway located in each branch. The SNMP traffic from each branch flows through an AnyNet/2 Sockets over SNA Gateway running in the branch to an AnyNet/6000 Sockets over SNA access node running on the same RISC System/6000* (RS/6000*) as NetView/6000 at the central SNA site (thereby connecting their multiple TCP/IP LANs across SNA). The AnyNet/2 Sockets over SNA Gateway seamlessly integrates with the existing CM/2 gateway located in each branch.

AnyNet/2 Sockets over SNA allows Canada Trust to be able to put leading-edge SNMP-based hubs in the branches of the bank and employ leading-edge SNMP-based management products like NetView/6000 without making major architectural changes to the mission-critical SNA network of the bank.

The tiered pricing of this gateway also makes pricing attractive for small, medium, and large customer configurations. This pricing allows AnyNet to be cost effective for those customers that have many remote branch office locations, such as Canada Trust.

Nykredit. Nykredit, a mortgage bank in Denmark, uses AnyNet/2 NetBEUI over SNA. The bank plans to put the AnyNet access node product on all of their OS/2 workstations to take advantage of end-to-end networking without limiting their choice of applications.

NetBEUI is the industry-standard programming interface for NetBIOS. Traditionally it is tied to the NetBIOS protocol in the same way that sockets is associated with the TCP/IP protocol and CPI-C is associated with the SNA protocol.

With the IBM AnyNet/2 NetBEUI over SNA product, customers can choose a NetBIOS-based workgroup application such as Lotus Notes to let users share data across remote locations using their existing SNA network. No change is required to the application or the network. This product supports any NetBEUI application including Lotus Notes, cc:Mail, IBM's LAN Server, Time and Place/2, and Person-to-Person/2*.

Benefits

By defining the TLPB and a common set of compensations to make up for differences in the various transport providers, MPTN breaks the binding between an application and the communications protocol. This enables the choice of applications and the choice of a communications protocol to be made independently of each other, resulting in several benefits

- For the end user:
  1. Existing applications can be run over additional network types, expanding the scope of existing applications and giving the end user a wider choice of possible applications.
  2. An application can be chosen based on its merits, since the choice is no longer restricted to the set of applications that can run over the installed communications protocol(s).

- For the application provider:
  1. Application writers developing applications that use communication services can select the API they use based on the functions the API provides, not based on the API the installed communications protocol(s) supports.
  2. Existing applications can be run over additional network types, expanding the market for those applications. Therefore, application providers can concentrate on improving their product and providing additional function for their end users rather than in developing different versions of their product to run on different communications protocols.

- For the network administrator:
  1. Selection of a communications protocol can be based on the merits of that protocol, not on the applications available for that protocol.
  2. Networks can be consolidated, and the number of communications protocols to be managed can be reduced while still supporting all existing user applications.
  3. A network can be changed without affecting

existing applications. If a newer, better communications protocol comes along, the old protocol can be replaced, and all existing applications will continue to work.

Another benefit is that MPTN is an open architecture, although originally developed by IBM. The X/Open Company has published the specifications for the XMPTN access node,[13] XMPTN address mapper,[16] and XMPTN formats.[22] X/Open documents are publicly available and are of sufficient detail to implement an XMPTN-compliant product.

Concluding remarks

The Multiprotocol Transport Networking architecture presented in this paper is a general and open solution that breaks the binding between distributed applications and communications protocols. As has been shown, the MPTN architecture enables existing applications to run unmodified over any communications protocol. It does so by providing a general schema for specifying the semantics of a transport protocol and for supporting those semantics using other transport protocols. As a result, the MPTN architecture decouples higher-layer protocols, application programming interfaces, and applications from protocols at the transport layer and below. In addition, MPTN transport-layer gateways provide an end-to-end communication facility across a number of networks running different protocols, thus permitting a collection of networks running different protocols to appear as a single logical network.

Allowing existing applications to run unmodified over any communications protocol lets customers simplify their networks by reducing the number of communications protocols supported in their networks.

The AnyNet products, which implement MPTN, are currently available on the OS/2, MVS, AIX, OS/400, and DOS/Windows platforms.

Acknowledgment

The authors would like to thank Kathryn Britton for her contribution to the MPTN architecture, and James P. Gray, Matthew Hess, Mark Pozefsky, and Kathleen Riordan for their assistance, guidance, and support.

Cited references and note

1. *Systems Network Architecture (SNA) Networks*, E. R. Coover, Editor, IEEE Computer Society, Los Alamitos, CA (1992).
2. M. T. Rose, *The Open Book: A Practical Perspective on OSI*, Prentice Hall, Englewood Cliffs, NJ (1990).
3. D. E. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, Second Edition, Prentice Hall, Englewood Cliffs, NJ (1991).
4. *IBM Local Area Network Technical Reference*, SC30-3383, IBM Corporation (1988); available through IBM branch offices.
5. C. Malamud, *Analyzing Novell Network*, Van Nostrand Reinhold Co., Inc., New York (1990).
6. C. Malamud, *DEC Networks and Architectures*, McGraw-Hill, Inc., New York (1989).
7. G. S. Sidhu, R. F. Andrews, and A. B. Oppenheimer, *Inside AppleTalk*, Second Edition, Addison-Wesley Publishing Co., Reading, MA (1990).
8. M. L. Hess, J. A. Lorrain, and G. R. McGee, "Multiprotocol Networking—a Blueprint," *IBM Systems Journal* **34,** No. 3, 330–346 (1995, this issue).
9. NetBIOS Working Group, *Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods, Request for Comments 1001*, DDN Network Information Center, SRI International (March 1987).
10. NetBIOS Working Group, *Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications, Request for Comments 1002*, DDN Network Information Center, SRI International (March 1987).
11. M. T. Rose and D. E. Cass, *OSI Transport Services on Top of TCP, Request for Comments 1006*, DDN Network Information Center, SRI International (April 1986).
12. K. Britton, W.-S. E. Chen, T.-Y. D. Chung, A. Edwards, J. Mathew, D. Pozefsky, S. Sarkar, R. Turner, W. Doeringer, and D. Dykeman, "Multiprotocol Transport Networking: A General Internetworking Solution," *Proceedings of the 1993 International Conference on Network Protocols*, IEEE Computer Society Press, Los Alamitos, CA (October 1993), pp. 14–25.
13. *Multiprotocol Transport Networking (XMPTN) Access Node*, X/Open Preliminary Specification, ISBN: 1-85912-040-7, X/Open Document Number P408, available from X/Open Publications, P.O. Box 96, Witney, Oxon OX8 6PG, England.
14. D. Robertson, *Multiprotocol Transport Networking: A Guide to IBM's AnyNet Products*, McGraw-Hill, Inc., New York, to be published.
15. Note that for some protocols (e.g., NetBIOS) addresses are not structured into a node-specific and a local part, and in those cases the complete address must be registered with the address mapper.
16. *Multiprotocol Transport Networking (XMPTN) Address Mapper*, X/Open Preliminary Specification, ISBN: 1-85912-039-3, X/Open Document Number P407, available from

X/Open Publications, P.O. Box 96, Witney, Oxon OX8 6PG, England.
17. R. J. Cypser, *Communications for Cooperating Systems*, Addison-Wesley Publishing Co., Reading, MA, ISBN: 0-201-50775-7 (1991).
18. M. Dolgicer, "MPTN Means Multiprotocol Without Routers," *Data Communications*, 97–99 (November 21, 1993).
19. B. Blakeley, H. Harris, and R. Lewis, *Messaging and Queuing Using the MQI: Concepts and Analysis, Design and Development*, McGraw-Hill, Inc., New York, ISBN: 07-005730-3 (June 1995).
20. W. Rosenberry, D. Kenney, and G. Fisher, *Understanding DCE*, O'Reilly & Associates, Sebastopol, CA, ISBN: 1-565-92005-8 (1992).
21. J. Auerbach, "TACT: A Protocol Conversion Toolkit," *IEEE Journal on Selected Areas in Communications* **SAC-8**, No. 1, 143–159 (January 1990).
22. *Multiprotocol Transport Networking (XMPTN) Formats*, X/Open Preliminary Specification, ISBN: 1-85912-043-1, X/Open Document Number P448, available from X/Open Publications, P.O. Box 96, Witney, Oxon OX8 6PG, England.

General references

D. M. Ogle, K. M. Tracey, R. A. Floyd, and G. Bollella, "Dynamic Protocol Selection for Socket Applications," *IEEE Network* **7**, No. 3, 48–57 (May 1993).

Multiprotocol Transport Networking: Technical Overview, GC31-7073, IBM Corporation (1993); available through IBM branch offices.

Accepted for publication March 30, 1995.

Diane Pozefsky *IBM Networking Software Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: dpoz@ralvm6.vnet.ibm.com).* Ms. Pozefsky is an IBM Fellow in the Networking Software Division. She received an Sc.B. degree in applied mathematics from Brown University in 1972, and a Ph.D. degree in computer science from the University of North Carolina at Chapel Hill in 1979. She joined IBM in 1979 and was one of the lead architects for the Advanced-Peer-to-Peer Networking enhancement to Systems Networking Architecture and the lead architect for Multiprotocol Transport Networking. She is a member of the Technology Council for the IBM Academy of Technology.

Roger Turner *IBM Networking Hardware Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: turner@vnet.ibm.com).* Mr. Turner graduated from Michigan State University in June 1981 with a B.S. degree in computer science. Upon graduation, he joined IBM. In 1987, he joined the Networking Systems Architecture area. In this area, he has worked on the Advanced Peer-to-Peer Networking architecture, Advanced Program-to-Program Communication (APPC, or LU 6.2) architecture, and the Multiprotocol Transport Networking (MPTN) architecture. He was also part of the team that implemented the AnyNet/2 APPC over TCP/IP product. His current responsibilities include working with the X/Open Company Ltd. to advance the XMPTN specifications. Mr. Turner received a First-Level Invention Award and an Outstanding Technical Achievement Award in 1994 for his work on MPTN.

Allan K. Edwards *IBM Networking Software Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: ake@ralvm6.vnet.ibm.com).* Mr. Edwards received a B.Sc. in electrical and electronic engineering from the University of Leeds, England, in 1971. He received an M.S. in electrical engineering from North Carolina State University in 1985. He joined IBM in 1985, and is currently an advisory programmer working in the area of multiprotocol networking. Mr. Edwards received a First-Level Invention Award in 1994.

Soumitra (Ronnie) Sarkar *IBM Networking Software Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: sarkar@vnet.ibm.com).* Dr. Sarkar received a B.Tech. in electrical engineering from the Indian Institute of Technology at Kharagpur in 1978, an M.Tech. in computer science from the Indian Institute of Technology at Madras in 1980, and a Ph.D. in computer science from the Ohio State University in 1989. He joined IBM in 1989. From 1990 to 1991, he worked on the architecture of Multiprotocol Transport Networking. From 1992, he has been working on the implementation of various products based on the architecture. He is currently an advisory programmer in the AnyNet Product Development Organization. Dr. Sarkar received a First-Level Invention Award in 1994.

Johny Mathew *IBM Networking Software Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: jmathew@ralvm6.vnet.ibm.com).* Dr. Mathew received his Ph.D. in electrical engineering from the City University of New York in 1990. He joined IBM at Research Triangle Park in 1990 and since that time has been working on multiprotocol networking. His research interests include distributed systems and heterogeneous communication networks. He is a member of the IEEE Communications Society and IEEE Computer Society.

Gregory Bollella *IBM Networking Software Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: bollella@vnet.ibm.com).* Mr. Bollella received his M.S. degree in computer science from the University of North Carolina at Chapel Hill in 1990 where he is currently completing research for the Ph.D. degree. He joined IBM in 1991 and is currently an advisory programmer in the AnyNet Product Development Organization working as technical lead for the MPTN address mapper and configuration tools.

Karen Tracey *IBM Networking Software Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: kmt@vnet.ibm.com).* Dr. Tracey received her B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Notre Dame in 1987, 1989, and 1991, respectively. She joined IBM at Research Triangle Park in 1991. She worked on a prototype for one of the original MPTN products and is presently a developer in the AnyNet product organization.

Dan Poirier *IBM Networking Software Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: poirier@vnet.ibm.com).* Mr. Poirier is a senior associate programmer in IBM's AnyNet area in Research Triangle Park. He received an M.S. degree in computer science from the University of North Carolina at Chapel Hill in 1991. He joined IBM in 1992, and has worked as a programmer on several workstation networking products for SNA and AnyNet.

John Fetvedt *IBM Networking Hardware Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: fetvedt@vnet.ibm.com).* Mr. Fetvedt is a senior programmer in APPN/APPC architecture. He joined the Systems Network Architecture group at IBM's Research Triangle Park facility in 1984 where he has participated in Advanced Program-to-Program Communication, the Common Programming Interface for Communications, the OSI TP standards activity, the X/Open Distributed Transaction Processing work group, and the Multiprotocol Transport Networking architecture. Prior to that he worked on a wide variety of products, with emphasis on communications and distributed processing, since joining IBM in 1965.

W. Sands Hobgood *IBM Networking Hardware Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: sands@vnet.ibm.com).* Mr. Hobgood has been involved in communications technology ever since the early days of Arpanet, when he developed software for the exchange of laboratory data between IBM's process-control computers and mainframes at NASA and Bell Laboratories. He contributed to the development of layered networking architecture that is fundamental to the design of contemporary communications protocols. He has represented IBM in ISO and ANSI standards committees on office automation and open systems, and is now involved in multimedia architecture and product development.

Willibald A. Doeringer *IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland (electronic mail: wdo@zurich.ibm.com).* Dr. Doeringer received the M.S. and Ph.D. degrees in mathematics from the Karlsruhe University, Germany, in 1979 and 1983, respectively. After two years of teaching at the Department of Operations Research at the University of Ulm, Germany, he joined the Computer Communications Group of mbp Systems and Software, a German software house, to work on the design and implementation of a wide range of protocol software for public and proprietary network architectures. Since 1988 he has been with the IBM Zurich Research Laboratory, focusing on high-speed protocols and communication systems architecture, the latter with special emphasis on heterogeneous internetworking. Dr. Doeringer is a member of several professional societies, the Cusanus Werk, and the International Association of Waldorf Schools.

Douglas Dykeman *IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland (electronic mail: ddy@zurich.ibm.com).* Dr. Dykeman is manager of the ATM Networking group at the IBM Zurich Research Laboratory. He received his Ph.D. from the University of Waterloo in computer science in 1988, and joined IBM that same year. His research interests include internetworking, routing in large networks, and design and implementation of advanced communications subsystems.

Reprint Order No. G321-5578.