Multiprotocol Transport Networking: A General Internetworking Solution

Kathryn Britton, Wen-Shyen E. Chen, Tein-Yaw D. Chung[1]
Allan Edwards, Johny Mathew, Diane Pozefsky,
Soumitra Sarkar, Roger Turner,
Willibald Doeringer*, Douglas Dykeman*

IBM Networking Systems
P.O Box 12195, Research Triangle Park, NC, 27709, U.S.A.

*IBM Zurich Research Laboratory
Rueschlikon, Switzerland

ABSTRACT

A user of computer networks has many choices for communication protocols. Current application programming interfaces (APIs), such as sockets and CPI-C, run over some of these protocols. The trend toward global interorganizational networks is handicapped because programs written to one interface will often run on only one type of transport network, and two programs must run on the same full protocol stack to communicate. *Ad hoc* solutions exist, but they are expensive and limited.

The Multiprotocol Transport Networking (MPTN) architecture proposed in this paper is a general solution to providing interconnectivity for applications. The MPTN architecture provides a protocol-independent system interface that includes most functions provided by existing 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. Using the MPTN architecture, existing and new applications can function unmodified over any transport supported under the MPTN interface. In addition, MPTN transport-layer gateways provide an end-to-end communication facility across a number of networks running different protocols. Therefore, a collection of networks running different protocols can serve as a single logical network.

The paper describes four major aspects of the MPTN solution: a *transport-layer protocol boundary*, providing generic semantics at the transport layer so that the applications can be transport-independent; *protocol compensation*, adjusting for discrepancies between services required by generic transport layers and those provided by individual transport protocols; *address mapping*, resolving the difference when the application and the protocol used to transport data use different address types; and general *transport-level gateways*, connecting networks running different protocols.

1 Introduction

Communication networks have been developed over time by different organizations to provide connectivity in specific environments using technologies that were current at that time. As a result, today's networking environment consists of a proliferation of networks whose architectures and protocols vary widely (such as SNA [1], OSI [2], TCP IP [3], NetBIOS [4], IPX [5], DECnet [6], and AppleTalk [7]).[2] Each communication protocol has its own advantages and disadvantages in terms of network performance, network overhead, ease of network management, availability of important applications, presence or absence of multi-vendor implementations, and difficulty of configuration.

Traditionally, companies satisfied all of their communication requirements with a single protocol architecture. This is no longer true. Most large organizations have

[1] This author is currently with Computer Engineering and Science Department, Yuan-Ze Institute of Technology, Taiwan, R.O.C.

[2] Some trademarks/registered trademarks of IBM Corporation in the United States and/or other countries appear in this paper. They are: **AnyNet, OS/2,** and **MVS**.

Some trademarks/registered trademarks of companies other than IBM Corporation also appear in this paper. They are:

Trademark/Registered Trademark	**Owned by**
AppleTalk	Apple Computer Inc.
DECnet	Digital Equipment Corporation
IPX, SPX	Novell Inc.
X/Open	X/Open Company Limited

0-8186-3670-X/93 $03.00 © 1993 IEEE

several communication protocols running in different parts of their organizations and the number of multiprotocol nodes continue to grow. Given the reality of a multiprotocol world, problems exist for both network owners and application vendors:

- The diversity of communication protocols provides many choices, but no single protocol can meet all of the networking requirements of an organization.
- Supporting multiple active protocols will incur higher cost in each end system (installation, demand for extra memory, storage and processing power). In addition, the task of building and managing each network adds complexity.
- The immense investment involved in migrating from current multiple protocols to a small set of targeted communication protocols prevents the network user from making radical network changes.
- The application vendor who wants to broaden the markets for his/her products faces extensive costs of rewriting applications for different communication protocols.

1.1 Survey of Current Solutions

In response to the aforementioned problems, many different solutions have been developed in industry, academia, and various standard organizations. These proposed solutions can be classified into the following categories.

Converging to One Standard Communication Protocol: An apparent solution to the problem of internetworking in heterogeneous environments is to converge to one worldwide standard communication protocol, such as the OSI standard. Although the standard plays a key role in reducing the number of protocols, convergence as the solution is not feasible [8, 9] because there are already thousands of SNA, TCP/IP, IPX, DECnet, AppleTalk and other networks installed at various sites in the world with tremendous investments in existing hardware and applications.

One-to-One Protocol Conversion: This is an *ad hoc* approach in which conversion between two specific protocols is performed. Depending on where the protocol difference occurs, the conversion can take place in an end system [10, 11, 12, 13, 14], or in a gateway between two subnetworks obeying different protocols [15]. For example, the use of RFC 1006 for running OSI upper layers over TCP/IP illustrates protocol conversion that takes place in an end system. Note that to provide full interoperability for N different protocols, N(N-1)/2 protocol converters have to be constructed, which is economically infeasible. The use of an application gateway to allow TCP/IP's SMTP and OSI's X.400 to operate with each other is an example of protocol conversion that takes place in a gateway. In this case, the number of protocol converters is even greater because it is unique for each application.

Protocol Server/Remote Application Programming Interface (API): With this approach, one or more protocol servers are provided for a group of LAN workstations[3]. Instead of processing the API requests locally, a client workstation forwards the requests to the protocol server using some LAN protocol. The protocol server will act as an agent for the client workstation to communicate with the destination. As a result, only the protocol server needs to maintain the needed state tables and perform protocol processing. Drawbacks of this approach are that the protocol server can potentially become the performance bottleneck and APIs with a lot of local functions can increase LAN traffic significantly. In addition, when two LAN workstations want to communicate with each other using that API, they have to go through the protocol server even if they are directly connected, resulting in performance degradation.

Encapsulation/Tunneling: Using the encapsulation/tunneling approach [16, 17, 18], a complete network is used as a subnetwork to provide connectivity to another protocol architecture. This technique is only applicable when both the source and destination end systems use the same full protocol stacks. While the network using the subnetwork does not need to know the topology of the subnetwork, the network owner still configures and manages both types of networks.

Most multiprotocol networks use some form of tunneling to carry protocols that cannot be routed natively. Because undifferentiated forwarding of traffic across a wide-area network can cause severe performance problems, all modern tunneling techniques perform filtering, especially to prevent broadcast storm from entering the backbone network. Filtering significantly increases the cost of this technique because it requires understanding of the traffic being carried.

General Solutions: In order to improve the re-use of concept and code, two classes of general solutions have been previously proposed: *formal methodologies* to synthesize protocol converters, and a *protocol conversion toolkit* to supply a reusable set of conversions.

Formal Methods: There have been a large number of proposals to establish formal methods for synthesizing protocol converters from their specifications [19, 20, 22, 23, 24, 25, 26, 27, 28]. The idea is that given the formal

[3] This technique is not unique to LAN workstations (for IBM's licensed program OSI/Communications Subsystem uses this technique between mainframes), but this is its most common use.

specifications (e.g., Communicating Finite State Machines) of two different protocols, the formal method finds a converter, if one exists, for the given protocols. Since formal specifications can be machine-readable, the process of finding a protocol converter can be automated. Although formal synthesis is proposed to provide formally verifiable solutions, it is computationally difficult and can only be applied to a limited subset of the problems.

Protocol Conversion Toolkit: The transport abstraction conversion toolkit [29], or *TACT*, was intended as a solution to a broad subclass of protocol conversion problems. It consists of a set of small, modular, mutually compatible tools which can be composed in various combinations to provide protocol conversion and substitution. The number of functions addressed by TACT is fairly limited and does not include such items as differences of expedited data support and close semantics. In addition, the emphasis of TACT is on providing conversion at the communication end systems, and does not address the design of a gateway.

Each of these individual solutions addresses part of the problems of internetworking and has advantages as well as limitations. In general, the aforementioned approaches do not provide adequate flexibility when compared to the degree of interoperability required, in which any application program can run over any network protocol.

1.2 The MPTN Solution

We present in this paper a general solution to multiprotocol problems. The solution, the *Multiprotocol Transport Networking* (MPTN) architecture, is

- scalable: adding new networks and interfaces implies linear growth in implementation effort.
- cost effective: providing protection of the large investment in current networks, and
- flexible: allowing any application to run over any network.

Before the MPTN architecture is described, some terms used in the rest of the paper are defined. A *single protocol transport network* (SPTN) is a network in which all nodes use the same protocol architecture. If a protocol stack is split at the point where transport function is provided, i.e., the transport layer, the portion above the split can be thought of as a *transport user*, and the portion below as the *transport provider*.

As illustrated in Figure 1, the term *matching* describes a horizontal relationship between peers. To *match* means

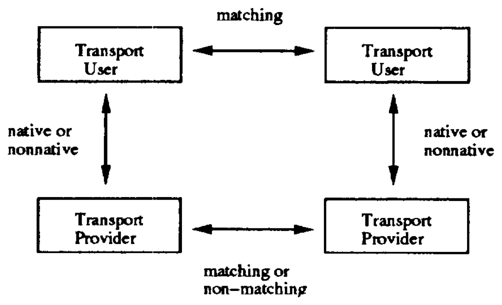

Figure 1. The use of "matching" and "native".

to have the same application support and to belong to the same application programming family. For example, two sockets programs or two CPI-C programs match, but a sockets program and an CPI-C program do not. The terms *native* and *nonnative* describe a vertical relationship between a user and its corresponding provider. If the transport user and transport provider are from the same protocol architecture, they are native with respect to each other. Otherwise, they are nonnative with respect to each other. A *native node* is a node with no MPTN function.

The *transport-layer protocol boundary* (TLPB) is the semanatic defintion of a generic transport interface[4], to enable the operation of application programs across nonnative transport providers (e.g., sockets applications over SNA). An end node with the TLPB and supporting MPTN protocols is referred to as an *MPTN access node*. The term *transport network* is used to refer to a network running the protocols of a particular architecture up to and including the transport layer. MPTN facilitates interoperation of end nodes by using *transport gateways* to concatenate different transport networks (e.g., between NetBIOS and SNA transports). An MPTN network is a homogeneous network that consists of a single SPTN or a heterogeneous network that consists of a number of SPTNs interconnected by MPTN transport gateways. In an MPTN network, *communicating transport users must always match*. Transport providers need not match if one or more MPTN transport gateways (defined later in this section) are available to concatenate SPTNs running different communication protocols.

A principal conceptual component of the MPTN architecture is the TLPB, which provides the generic semantics of different existing transport-layer protocols. Each transport user in an MPTN access node has to be augmented to use TLPB functions (as opposed to its native transport-provider functions) to operate nonnatively. Each trans-

[4] It is a protocol boundary, not an interface per se. The syntax is left open for implementations of different hardward/software platforms.

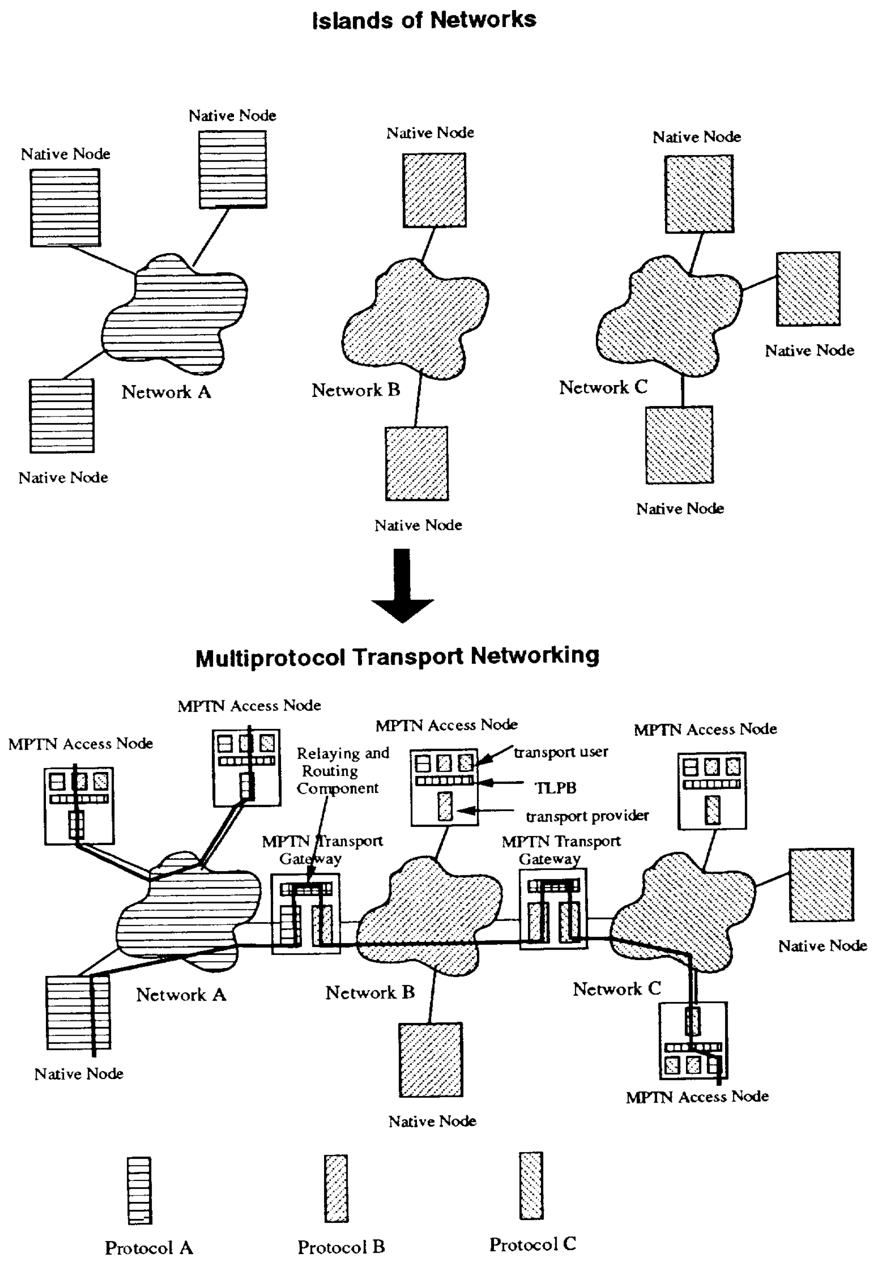

Figure 2. Overview of Multiprotocol Transport Networking.

port provider has to be augmented to support the generic semantics of the TLPB. For transport-layer APIs (e.g., sockets and NetBEUI), the transport user is simply the API processing layer. For higher-layer APIs (e.g., CPI-C), the transport layer is the higher layers (above transport) of the native protocol stack. The TLPB removes the dependency of APIs and higher layer protocols on specific transport protocols. An MPTN transport gateway has transport providers for each SPTN it concatenates. The MPTN transport gateway can communicate with native nodes using native protocols, as well as with MPTN access nodes using MPTN protocols.

Figure 2 provides an overview of the MPTN solution. The figure illustrates how, in an MPTN network, applications originally running on protocol B can now communicate with their partners using transport provider A, as long as the transport users match each other. An application running on a native node of network A needs no change to communicate through an MPTN transport gateway with its partner running nonnatively in an MPTN access node on network C.

MPTN Access Node:

Figure 3 shows a high-level picture of an MPTN access node. The problem of translating the transport semantics requested by a transport user into the semantics supported by a nonnative protocol stack can be addressed by modifying each transport user to generate TLPB primitives, and augmenting each transport provider to support these TLPB primitives.

The figure also shows MPTN transport users that request services from the TLPB, which are in turn serviced by MPTN transport providers that provide these services. The Common MPTN functions component (CM) provides services such as connection establishment, datagram routing, general address mapping, and selection of compensations. A protocol compensator (discussed in Section 3) is required for each transport provider. The figure also shows an optional address mapper component, which can communicate with the local address mapping (directory) services component provided by CM, or with a remote general address mapping services component through a transport connection. The TLPB provides a common set of transport services to satisfy the requirements of existing transport users. The discrepancies between the functions offered by the TLPB and the actual services available in the native transport networks is addressed by a universal set of *MPTN protocol compensations*. The MPTN architecture also provides *address mapping* functions so that an application can identify itself and its partner(s) using addresses belonging to its native address type, even though the underlying transport provider uses addresses of a different type.

MPTN Transport Gateway

MPTN transport gateways build on the TLPB functions of MPTN access nodes to provide connectivity between unlike SPTNs. SPTN connections are terminated at the MPTN transport gateway. The gateway uses normal TLPB functions in order to relay data between related SPTN connections to achieve end-to-end connectivity. In order to accomplish this function, the CM is extended in an MPTN transport gateway as illustrated in Figure 4.

The MPTN transport gateway includes a routing protocol to determine which MPTN transport gateways to use to provide connectivity between a given pair of end systems in different SPTNs, and a relaying function to relay data between those SPTN connections. Similar routing and relaying support is also provided for connectionless traffic.

The rest of the paper is organized as follows. Section 2 summarizes the set of generic transport services that the MPTN architecture supports. Section 3 presents the protocol compensation provided to bridge the discrepancies between the transport services requested by a transport user and those provided by the transport provider. The address mapping mechanism of the MPTN architecture is described in Section 4. Section 5 describes MPTN transport gateways. Finally, concluding remarks are made in Section 6.

2 MPTN Transport Services

The set of MPTN transport services, as embodied in the TLPB, contains those services considered to be generally useful and necessary for transport even if not included in all protocols. No one transport protocol supports all of its functions. For example, SNA, NetBIOS, and OSI support a record model for data exchange, while TCP/IP supports a stream model. Both data stream formats are supported by the TLPB. Compensation for missing functions is performed by MPTN protocol compensations. The transport services described in this section fall into two broad classes: connection-oriented and connectionless.

2.1 Connection-Oriented Services

Connection-oriented services are used to establish a logical connection between two partners for the period during which they communicate with each other. Data transfer takes place over a full-duplex connection in a reliable manner. The MPTN architecture takes advantage of all the services provided by the underlying transport provider, such as reliable delivery, and compensates for any mismatches between a particular transport user and transport provider. There are three distinct phases in a connection-oriented data exchange:

- **Connection Establishment**

  Some transport users send connection data (e.g., a BIND in SNA) to specify the characteristics of the connection. The partner examines the connection data before accepting or rejecting the connection. Others (e.g., TCP/IP) do not. The TLPB supports connection establishment with or without connection data.

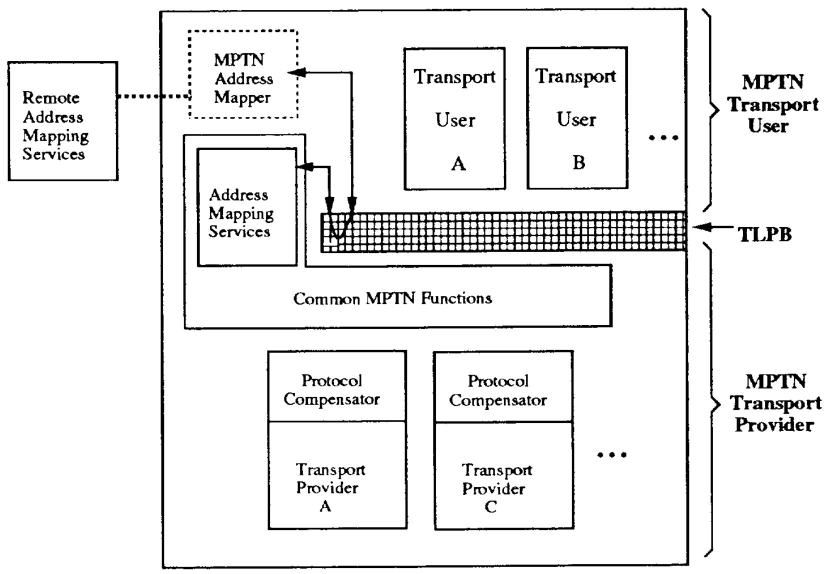

Figure 3. An MPTN Access Node

The TLPB allows a transport user to define a *service mode* that should be used for establishing the connection. For example, a user may be looking for a secure, high-bandwidth connection or one that minimizes delays.

- **Data Transfer**

  Some transport users assume data record boundaries are preserved by the underlying transport provider. Others send data as a byte stream with no record boundary delineation. Both data-delivery models are supported by the TLPB. Whereas record boundaries are preserved, the maximum record sizes vary among the various transport providers. The MPTN architecture compensates if a transport user expects a larger maximum than the transport provider supports.

  Some transport users send expedited data. Within the TLPB definition, expedited data has the following characteristics:

  - It will not be delivered later than any normal data sent after it, though it may be delivered ahead of normal data sent before it.
  - It can be distinguished from normal data when it is sent and received.

- **Connection Termination**

  During connection termination, some transport users deliver user termination data, such as the reason for the termination, to the partner. Other transport users do not. Some transport users expect to terminate one direction of the (full-duplex) connection at a time (*simplex* termination), while others expect to terminate both directions of the connection at once (*duplex* termination). For example, TCP/IP shutdown provides simplex termination, while OSI TP4 and NetBIOS support duplex termination only. Similarly, some transport users expect the delivery of all data preceding connection termination (*orderly* termination), while others do not expect this guarantee (*abortive* termination). For example, TCP/IP shutdown provides orderly termination, while OSI transport provides abortive termination. When connections fail, some transport users require immediate notification; others do not. All of these variations are supported by the TLPB.

2.2 Connectionless Services

Connectionless service is the mode of data transfer in which different units of data are passed independently through the network. Delivery is provided on a best-effort basis, but undetected packet loss may occur, e.g., if the network becomes congested. Compensation is defined for the case where a transport user sends a larger datagram than the transport provider supports.

Various connectionless modes are supported by TLPB:

- **Unicast Datagrams**, i.e., datagrams delivered to a single partner.

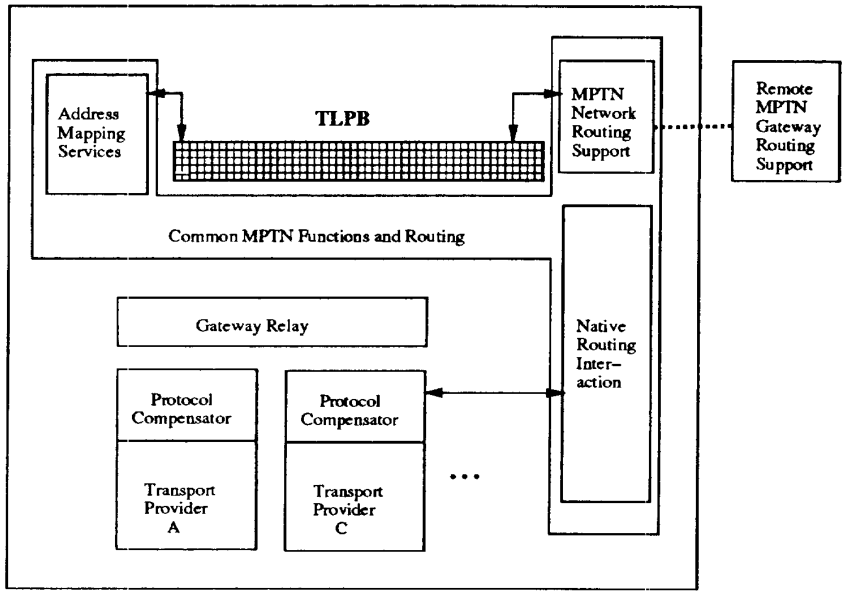
Address Mapping Services
TLPB
MPTN Network Routing Support
Remote MPTN Gateway Routing Support
Common MPTN Functions and Routing
Gateway Relay
Native Routing Inter-action
Protocol Compensator
Protocol Compensator
Transport Provider A
Transport Provider C

Figure 4. An MPTN Transport Gateway

- **Multicast Datagrams**, i.e., datagrams delivered to partners that have joined a multicast group. The sender does not know which programs belong to the group and need not be a member of the group itself.

  Multicast is a significant service provided on various local-area networks. TLPB support for multicast requires that all members of a multicast be located in the network with the same *network identifier* (netid) in order to scope the extent of delivery.

- **Broadcast Datagrams** are treated as a special case of multicast and are therefore limited to a specific netid.

As noted earlier, the transport services supported by the TLPB might not be available in the underlying transport providers and therefore protocol compensation needs to be provided. The following section presents the MPTN protocol compensation solution.

3 Protocol Compensation

Since the TLPB provides semantics useful for any transport user, one standardized compensation technique can be defined for each TLPB feature that a transport provider may not support. Thus, associated with the transport-layer protocol boundary are a small number of compensation techniques. For each transport provider used, the set of compensations needed is identified by the TLPB features it does not support natively. This set constitutes the protocol compensator for that transport provider.

Most protocol compensation is provided by a *protocol compensator* component that adds special headers to the data stream. The protocol compensator at the receiving end recognizes these headers and takes appropriate (compensatory) actions. This section describes the compensation services that could potentially be required and how the compensations are performed. A typical transport provider only needs to implement a subset of these compensation services.

3.1 Compensation Mechanisms

Protocol compensation services are performed cooperatively by the two protocol compensators involved at the sending and receiving ends of the underlying connection of a datagram. When an MPTN connection is being established, the source node will determine if there are any transport service differences between the user and the provider. The required services are exchanged so that an MPTN transport gateway can determine which compensations are needed over a different transport provider in the next SPTN.

Most of the compensation services are performed by adding special headers to the data stream. Exceptions to this are 1) the compensation performed for sending connection data, which uses a separate flow, 2) the compensation performed for sending expedited data, which adds a header to the data stream and also sends the header and data as a datagram, 3) the compensation performed for connection outage notification, which exchanges special datagrams, and nothing flows on the data stream, and 4) the compensation for performing duplex-over-simplex termination, where only state information is required. When a compensation requires special headers, a header is attached to every packet on the data stream even if the data in that packet does not require compensation. This is necessary so that the protocol compensator in the partner node can distinguish headers from data. On record-oriented protocols, the header is one byte; on stream-oriented protocols, the header is five bytes because it includes a length field identifying the location of the next header.

3.2 Universal Set of Compensations

The list below includes the compensation that have been identified for various TLPB services, and describes how each compensation is to be performed for a transport protocol that does not provide the serices.

1. **Connection data**: Connections between user applications are set up as virtual connections over native connections between the transport providers. Once the native connection is set up, special packets containing the (protocol-specific) names of the applications, the connection characteristics they require, and the connection data and response (if any) are exchanged between the two CMs. Any connection data sent by the application initiating the connection is extracted from the special packet by the CM at the passive end of the connection request, and passed to the transport user to which it was sent. Any reply to the connection data is handled in the same way.
2. **Termination data**: A special header is inserted by the compensator at the sending end to identify the termination data. The compensator at the receiving end removes the header, extracts the termination data, and passes it up to the application. For abortive termination, the header, and any termination data, is sent as expedited data, which might involve futher compensation techniques (see below).
3. **Record boundaries**: A special header identifies the beginning of the record and its length, enabling it to be reassembled by the receiving compensator even if it has been split by a stream-oriented transport protocol.
4. **Expedited data**: A special header identifies the expedited data and its length. It is sent in the normal data stream and extracted by the compensator at the other end. The advantage of sending expedited data in the normal data stream is that it can not arrive after the normal data that follows it. The expedited data is also sent in a datagram to ensure delivery to the partner in case of congestion on the connection.
5. **Correlation of expedited and normal data**: When the transport provider does not support expedited data, positional correlation is automatically provided since expedited data is sent on the normal data stream. When expedited data is supported by the underlying transport protocol, positional correlation can be provided by adding a special header in the normal data stream (as a marker) even though the expedited data itself is sent on a separate stream.
6. **Record length difference**: Record length differences are handled for normal data by segmentation. Since ordering is guaranteed, only the end of the record needs to be identified in the segment header.
7. **Expedited length compensation**: If the transport provider supports expedited data, but transport user data length is greater than tranport provider data length, the user data will be segemented and a header will added to each segment. Each of the segments will then be sent as expedited data. If the transport provider does not support expeditèd data, normal expedited compensation will perform length compensation.
8. **Datagram length compensation**: Every unicast or multicast datagram exchanged between transport users on nonnative providers already contains headers with additional information about source and target names so that the datagram can be delivered to the correct application. When there is a length difference, the datagram is segmented, an additional header is added to each segment, and the segment is sent out as an individual datagram. Information in the segment header that assures proper reassembly, includes a unique datagram id, a sequence number, and an indicator for the last segment.
9. **Orderly termination over abortive termination**[5]: With abortive termination, the underlying transport protocol does not make any effort to flush data in transit when it receives a request to terminate a connection. When the transport user requiring orderly termination requests that the connection be terminated, the compensator at its end sends a packet with a special header on the normal data stream. When the compensator at the other end receives this packet, it can be sure that it has received all the normal data

[5] The discussion of termination describes compensations when only a single type of termination is used. In many cases, multiple types of termination can be used and compensations become somewhat more complex.

that preceded it. It then sends a special acknowledgement packet back to its partner; the partner can then issue the termination since all normal data has been flushed.

10. **Simplex termination over duplex termination:** When the first transport user issues a simplex termination, a special packet is sent by the compensator to its partner. The partner notes the receipt of this packet, but does not terminate the connection. When the transport user it is supporting also issues a simplex termination, the compensator issues a (duplex) termination, actually terminating the connection.

11. **Duplex termination over simplex termination:** In this case, since the two compensators expect their transport users to request duplex termination, this compensation is performed without inserting any additional information in the data stream. When one user issues a duplex termination, the compensator serving the user issues a simplex termination to terminate one end of the transport connection. When the partner compensator is informed about the connection termination, it immediately terminates its end of the connection.

4 MPTN Address Mapping

Each transport provider supports its own addressing scheme to allow users to establish communication with specified partners. For a nonnative transport user, the format of addresses used by the two are different.

MPTN address mapping services allow transport users to identify partners using addresses that are not recognized by the underlying transport provider. These mechanisms include means to

- register a program address so that it is accessible over nonnative transport providers,
- map a destination program address to an address understood by the serving transport provider, and
- convey the program addresses end to end so that the receiving node can route to the proper destination application, and the destination application can properly identify the sender.

In order to allow unambiguous identification of communicating partners, transport addresses must be unique. In SNA, OSI and TCP/IP, transport address uniqueness is controlled by administration, while in NetBIOS, a protocol is used to detect duplicate names. However, since transport addresses are simply bit strings, the same transport address may appear in different protocol address spaces. To avoid ambiguity, MPTN protocols use ordered pairs called *MPTN qualified transport addresses*: (address type, protocol-specific address). Since both the address type and the specific transport addresses in a protocol address space are unique, the MPTN transport address is also unique.

Addresses are normally structured as (netid.hostid.local-address). In order to support scalability of the architecture, MPTN components take advantages of this structure and use addresses as follows:

1. Routing within MPTN access nodes is based on full addresses to reach specific applications.
2. Nonnative address mapping within an SPTN is based on (netid.hostid) to locate a node in a network.
3. Gateway routing is based on netid to locate a network within an internet.

4.1 Address Mapping Alternatives

Different address pairs and protocols offer a number of potential solutions for address mapping. In order to minimize additional components and take advantage of as much infrastructure as possible, MPTN address mapping offers the following three alternative mechanisms:

1. Address mapper, an MPTN component that dynamically resolves transport user node addresses to transport provider node addresses.

   An Address Mapper uses a database which holds the transport user node address to transport provider node address mappings for the MPTN nodes it serves. The address mapper is used in connection establishment as follows:

   a. An MPTN access node registers a node-level (transport user, transport provider) mapping so that it is accessible through nonnative transport providers.

   b. An MPTN access node queries the address mapper for its partner's transport provider address. Given a transport user address, an MPTN access node derives the full address of the partner transport provider by concatenating the partner's node address (returned by the address mapper) to a well-known local address reserved for nonnative communication (e.g., port 397 for the TCP/IP transport provider).

   c. The native connection is established through the transport provider using native source and destination addresses.

   d. The MPTN connection is established by sending an MPTN connection packet, which contains the nonnative source and destination addresses, on the native connection.

2. Existing protocol-specific directories, possibly with enhancements.

   Some protocol-specific directories can be used to handle transport addresses of various formats. With this alternative, all transport user addresses are registered with a protocol-specific directory. For example, for addresses not requiring dynamic verification, the Domain Name Server (DNS) can be used by simply creating a new domain. In the MPTN connection establishment procedure described above, the first two steps are replaced by registration with and access to the protocol-specific directory.

3. Address mapping algorithms

   If the transport user address space is smaller than that of the transport provider, an algorithm can generate a transport provider address from the corresponding transport user's address. In this case, no database is required to hold the mappings. When transport user addresses are registered, they are converted to the corresponding transport provider addresses, which are then created and registered to the existing protocol-specific directory. Algorithmic translation is specific to a transport user-provider pair. Sockets over SNA uses algorithmic mapping [33]. In the MPTN connection establishment procedure described above, the first two steps are replaced with the algorithmic translation at both ends.

The first approach is the most general, but also the most costly. The other alternatives, when applicable, can be used effectively. When they are not applicable or when a number of different mechanisms would be needed, the address mapper solution is justified.

5 MPTN Transport Gateway

The purpose of an MPTN transport gateway is to provide connectivity between application programs across SPTNs of different types. MPTN transport gateways provide interconnectivity at the transport layer. This implies that the SPTN transport and lower-layer protocols are terminated at the gateway. User data is relayed over transport-layer connections between adjacent gateways, or between gateways and end systems. Thus an end-to-end MPTN connection consists of a series of SPTN connections concatenated by relays in MPTN transport gateways, as depicted in Figure 5. Datagrams are relayed end-to-end in a similar fashion.

The remainder of this section describes the manner in which MPTN transport gateways provide the described connectivity. The way that access is provided between end nodes and gateways is discussed first, and then the protocols required between MPTN transport gateways are discussed.

5.1 Protocols Between End Nodes and MPTN Transport Gateways

An MPTN transport gateway can be accessed by an end system using the native protocols of the network to which both are attached, or using MPTN protocols where required. For example, applications in both native nodes and MPTN access nodes can interoperate with their peers across the MPTN network. Thus an application written to the sockets interface, supported using the MPTN architecture on an SNA network, can interoperate with a peer socket running natively on a TCP/IP network.

Native Access: Native protocols can be used between an end node and an MPTN transport gateway. In this case the MPTN transport gateway must participate in the native routing protocols of the network in order to receive connection requests and datagrams that are to be forwarded through the MPTN network.

For protocols continuously distribute *reachability information*, the MPTN transport gateway must be able to exchange reachability information with the native protocol. That is, it must be able to advertise within the native protocol reachability to end systems outside of the native domain, and it must be able to learn what systems can be reached in the attached SPTN. With this capability, the native protocol will be able to correctly route connection requests and datagrams destined for nodes outside its SPTN to the gateway, and the MPTN network will be able to route data to the native domain.

For protocols that search for routing information when needed, the MPTN transport gateway must be able to receive all searches on the SPTN it is on and respond when the particular resource can be reached in some other SPTN, through the gateway. It sends a positive reply to the search if it can reach the address; this reply causes subsequent connection requests or datagrams to that address to be routed to the MPTN transport gateway by the native protocols.

The TLPB interface of an MPTN access node is extended in an MPTN transport gateway to allow these native routing functions to occur. This interface does not depend on any particular protocol and thus, the gateway functions above this interface are protocol independent.

Once the native routing protocols have determined that a connection or datagram is to be routed to an MPTN transport gateway, the native protocols are used to establish the connection or forward the datagram. Similarly, native protocols are used in the native access case to forward data from an MPTN transport gateway to a destination end system.

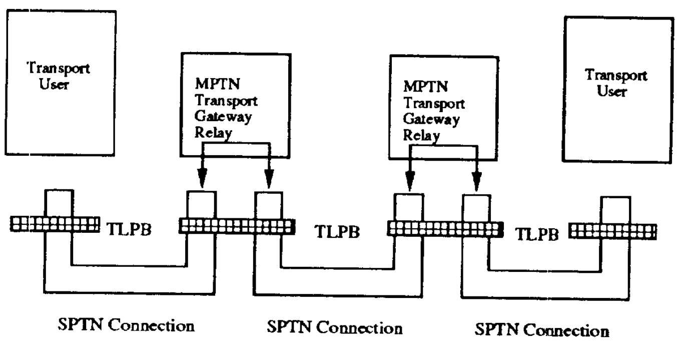

Figure 5. An MPTN Connection Comprised of Concatenated SPTN Connection Segments.

Nonnative Access: When application programs are run nonnatively on an MPTN access node, MPTN address mapping and signalling protocols are used to transfer data between the access node and the transport gateway. In the nonnative case, whenever the MPTN address mapping protocols in an SPTN determine that a particular destination address is not reachable within that network, the address mapper forwards the request to an attached MPTN transport gateway, if one exists. If the MPTN transport gateway can reach the specified address, it sends a positive reply so that the MPTN protocols route the connection or datagram to that transport gateway.

Address mappers directly report nonnative reachability information to local MPTN transport gateways so that connectivity to these nonnative resources is also possible across the MPTN network.

5.2 Protocols Between MPTN Transport Gateways

Since more than one MPTN transport gateway may lie on the path between two communicating end systems, MPTN protocols are required for routing and relaying data between these gateways.

Routing Protocols: Protocols are required between MPTN transport gateways to determine how to route connection requests and datagrams. The information required to make such decisions is provided by the MPTN gateway-to-gateway routing protocol, which is used to distribute reachability information between MPTN transport gateways. As described earlier, in large networks, it is not feasible to distribute reachability information for each individual address, so only information about netids is distributed between gateways.

The MPTN gateway-to-gateway protocols are built on the OSI Inter-Domain Routeing Protocol (IDRP) [32]. IDRP is designed to handle the interconnection of potentially large numbers of heterogeneous networks with different address formats and local routing protocols and provides built-in mechanisms for quality-of-service-based routing. An MPTN transport gateway uses IDRP to advertise to all adjacent gateways reachability information learned from local SPTNs and adjacent MPTN transport gateways. These IDRP messages always contain path information about the SPTNs to be traversed in order to detect potential routing loops.

Connection Management Protocols: As shown in Figure 5, connections between end users attached to different SPTNs consist of a series of concatenated SPTN transport connections, or connection segments, from the source to the first transport gateway, between the adjacent pairs of intermediate transport gateways, and from the final transport gateway to the destination end system. For reasons of protocol simplicity, improved performance, and fairness between different MPTN connections, connection segments are dedicated to a single end-to-end connection. Such MPTN connections are established when an MPTN transport gateway receives a connection request from a local SPTN. The transport gateway first checks the reachability of the destination address as described above[6]. The MPTN transport gateway selects the correct next hop from the routing tables based on the netid part of the destination address, and the required service mode. A new local transport connection is then established to the selected adjacent transport gateway with suitable service characteristics, a data relay instance is created to join the two local segments of the new connection, and an MPTN connection request message is

[6] This procedure becomes somewhat more complex when multiple disconnected SPTNs attached by MPTN gateways have the same netid. While this is a very significant aspect of the MPTN architecture, its description is beyond the scope of this paper.

forwarded including the destination address. This procedure is repeated hop-by-hop until the destination MPTN transport gateway is reached, where a connection is established to the destination end system. A positive reply from the destination is sent back through all intermediate transport gateways to the connection source, thus completing the end-to-end connection establishment and enabling data transmission. Delaying data flow until the completion of this end-to-end handshake prevents data packets queueing in a transport gateway and eliminates many potential performance and congestion problems.

Once an MPTN connection has been established, the main transport gateway function is to perform the relaying of data packets between the corresponding pair of connection segments. Although conceptually simple, this function faces the possible hazard of internal buffer depletion due to differences in throughput between the joined segments. The MPTN architecture solves this problem by providing buffer usage limits for each connection. If data cannot be sent on the outgoing connection, no buffers are freed and the relay stops receiving data, thereby applying back pressure to the incoming connection. Once the back pressure is applied, the native protocol will exercise its flow-control procedures on the connection segment, and hence the back pressure propagates hop-by-hop back to the source end system, which reduces its input to a sustainable rate.

When data packets have to be segmented, lessons learned from other designs [31] have led to placing the reassembly function in access nodes, or destination transport gateways for native access, so that repetitive reassembly is avoided.

Connection termination follows a procedure similar to that for connection establishment, with the connection being terminated hop-by-hop until it is closed end-to-end.

Datagram Protocols: Datagrams are forwarded on a hop-by-hop basis from the source, across MPTN transport gateways, and finally to the destination. The procedure for forwarding datagrams parallels that of connection establishment in that the destination is first located, and then the IDRP routing tables are used to determine the next hop for the datagram to which it is forwarded.

6 Conclusion

In this paper, we have presented a general solution for providing connectivity/communications services to applications independent of the communication protocols being used at the transport layer and below. The proposed Multiprotocol Transport Networking (MPTN) solution provides a scalable, cost-effective, and flexible means to achieve this. With MPTN,

- Network users can add a new application program based solely on how well it meets their business needs without requiring their network administrator to install a specific transport network.
- Network providers can choose the transport network best suited to their business needs without constraining their users' choice of applications programs.
- Application designers can develop new application programs that can run on many network types without complex communication logic.

Four major aspects of the MPTN solutions have been discussed: a transport-layer protocol boundary, protocol compensation, address mapping, and general transport-layer gateways.

AnyNet products, which implement the MPTN architecture, are currently available on both OS/2 and MVS platforms to enable SNA applications to run over TCP/IP transport (including gateway function), and TCP/IP sockets applications to run over SNA [33, 34, 35]. This architectue is currently being considered as a basis for an X/Open specification, and besides IBM, companies such as ki Research and Proginet have indicated their intention to build products conform to the MPTN architecture. For further information, please refer to [36]

7 Acknowledgement

The authors would like to thank James P. Gray, Matthew Hess, John Nicholson, Mark Pozefsky, and Mona Vance for their assistance, guidance, and support.

References

1. *System Network Architecture (SNA) Networks*, E. R. Coover Ed., IEEE Computer Society, Los Alamitos, CA, 1992.
2. M. T. Rose. *The Open Book: A Practical Perspective on OSI*, Prentice Hall, Englewood Cliffs, N.J., 1990.
3. D. E. Comer. *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, Second Edition, Prentice Hall, Englewood Cliffs, N.J., 1991.
4. *IBM Local Area Network Technical Reference*, Order Number SC30-3383, IBM Corp., 1988; available through IBM branch offices.
5. C. Malamud. *Analyzing Novell Network*, Van Nostrand Reinhold, New York, NY, 1990.
6. C. Malamud. *DEC Networks and Architectures*, McGraw Hill, New York, NY, 1989.
7. G. S. Sidhu, R. F. Andrews, and A. B. Oppenheimer. *Inside AppleTalk*, Second Edition, Addison-Wesley, New York, NY, 1990.
8. P. E. Green. "Protocol Conversion," *IEEE Transactions on Communications*, Vol. COM-34, No. 3, pages 257-268, March 1986.

9. P. E. Green. *Network Interconnection and Protocol Conversion*, Selected Reprint Series, IEEE Press, New York, 1988.

10. I. Groenbak. "Conversion between TCP and ISO Transport Protocols as a Method to Achieving Interoperability," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-4, No. 2, pages 288-296, March 1986.

11. M. T. Rose and D. E. Cass. "OSI Transport Services on Top of TCP," *Computer Networks and ISDN Systems*, Vol. 12, pages 159-173, 1986.

12. M. T. Rose and D. E. Cass. "OSI Transport Services on Top of TCP," Request for Comments 1006, DDN Network Inform. Center, SRI Int'l., April 1986.

13. NetBIOS Working Group. "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts a Methods," Request for Comments 1001, DDN Network Inform. Center, SRI Int'l., March 1987.

14. NetBIOS Working Group. "Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications," Request for Comments 1002, DDN Network Inform. Center, SRI Int'l., March 1987.

15. J. Onions and M. T. Rose. "ISO-TP0 Bridge between TCP and X.25," Request for Comments 1086, DDN Network Inform. Center, SRI Int'l., December 1988.

16. D. R. Boggs, *et. al.*. "PUP: An Internetwork Architecture," *IEEE Transactions on Communications*, Vol. COM-28, No. 4, pages 612-624, April 1980.

17. B. M. Leiner, *et. al.*. "The DARPA Internet Protocol Suite," *IEEE Communications Magazine*, Vol. 23, No. 3, pages 29-34, March 1985.

18. *IBM 6611 Network Processor Introduction and Planning Guide*, Order Number GK2T-0334, IBM Corp., 1993; available through IBM branch offices.

19. K. L. Calvert and S. S. Lam. "Formal Methods for Protocol Conversion," *IEEE Journal on Selected Areas in Communications*, SAC-8, No. 1, pages 127-142, January 1990.

20. K. L. Calvert and S. S. Lam. "Adaptors for Protocol Conversion," In *Proc. of IEEE INFOCOM '90*, pages 552-560, June 1990.

21. S. S. Lam. "Protocol Conversion," *IEEE Transactions on Software Engineering*, Vol. SE-14, No. 3, pages 353-362, March 1988.

22. M. T. Liu. "Protocol Engineering," In *Advances in Computers*, M. C. Yorits, editor, Vol. 27, pages 79-195, July 1989.

23. K. Okumura. "A Formal Protocol Conversion Method," In *Proc. of ACM SIGCOMM '86 Symposium*, pages 30-37, August 1986.

24. K. Okumura. "A Formal Protocol Conversion for CFSMs with FIFO Queues Model," TRL Research Report, TR 87-0036, IBM Tokyo Research Labratory, 1987.

25. K. Okumura. "Generations of Proper Adapters and Converters from a Formal Service Specification," In *Proc. of IEEE INFOCOM '90*, pages 564-571, June 1990.

26. J. C. Shu and M. T. Liu. "An Approach to Indirect Protocol Conversion," *Computer Networks and ISDN Systems*, Vol. 21, pages 93-108, 1991.

27. Y.-W. Yao, W.-S. E. Chen, and M. T. Liu. "A Modular Approach to Constructing Protocol Converters," In *Proc. of IEEE INFOCOM '90*, pages 572-579, June 1990.

28. Y.-W. Yao, W.-S. E. Chen, and M. T. Liu. "A Parallel Model for Constructing Protocol Converters," In *Proc. of IEEE GLOBECOM '90*, pages 1902-1907, December 1990.

29. J. Auerbach. "TACT: A Protocol Conversion Toolkit," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-8, No. 1, pages 143-159, January 1990.

30. *System Network Architecture LU6.2 Reference: Peer Protocols*, Order Number SC31-6808-02, IBM Corp., 1993; available through IBM branch offices.

31. D. P. Pozefsky, D. A. Pitt, and J. P. Gray. "SNA's Design for Networking," *IEEE Network*, Vol. 6, No. 6, pages 18-31, November 1992.

32. "Protocol for Exchange of Inter-domain Routeing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs," ISO/IEC DIS10747, August 1992.

33. D. M. Ogle, K. M. Tracey, R. A. Floyd, and G. Bollela. "Dynamic Protocol Selection for Socket Applications," *IEEE Network*, Vol. 7, No. 3, pages 48-57, May 1993.

34. *Multiprotocol Transport Feature Version 3 Release 4.2 for MVS/ESA: Socket over SNA User's Guide*, Order Number SC31-6487, IBM Corp., 1993; available through IBM branch offices.

35. *Multiprotocol Transport Feature Version 3 Release 4.2 for MVS/ESA: APPC over TCP/IP User's Guide*, Order Number SC31-6488, IBM Corp., 1993; available through IBM branch offices.

36. *Multiprotocol Transport Networking: Technical Overview*, Order Number GC31-7073, IBM Corp., 1993; available through IBM branch offices.