Implementing System/36 Advanced Peer-to-Peer Networking

by R. A. Sultan
P. Kermani
G. A. Grover
T. P. Barzilai
A. E. Baratz

System/36 Advanced Peer-to-Peer Networking (APPN) provides highly dynamic, fully distributed peer networking for low-end processors. It is built upon existing SNA Logical Unit 6.2 and Node type 2.1 support. APPN presents System/36 users with a simplified model of communications. The structure of the APPN subsystem is outlined, with particular emphasis on the integration of APPN functions with existing SNA support. The authors describe how particular aspects of the APPN design have been tuned to the System/36 operating environment.

Advanced Peer-to-Peer Networking[1] (APPN) provides enhanced communications capabilities for both the System/36 and the growing number of products that attach to the System/36 via Systems Network Architecture[2] (SNA) protocols. System/36 is a multiuser system intended for business applications. In 1985, System/36 introduced Advanced Program-to-Program Communication (APPC), which included support for SNA Logical Unit (LU) type 6.2 and Node type 2.1 protocols. LU 6.2[3,4] allows transaction processing programs in the System/36 to converse with programs in remote processors via an architected set of verbs. Node type 2.1[5] allows adjacent[6] nodes to communicate with each other as peers.

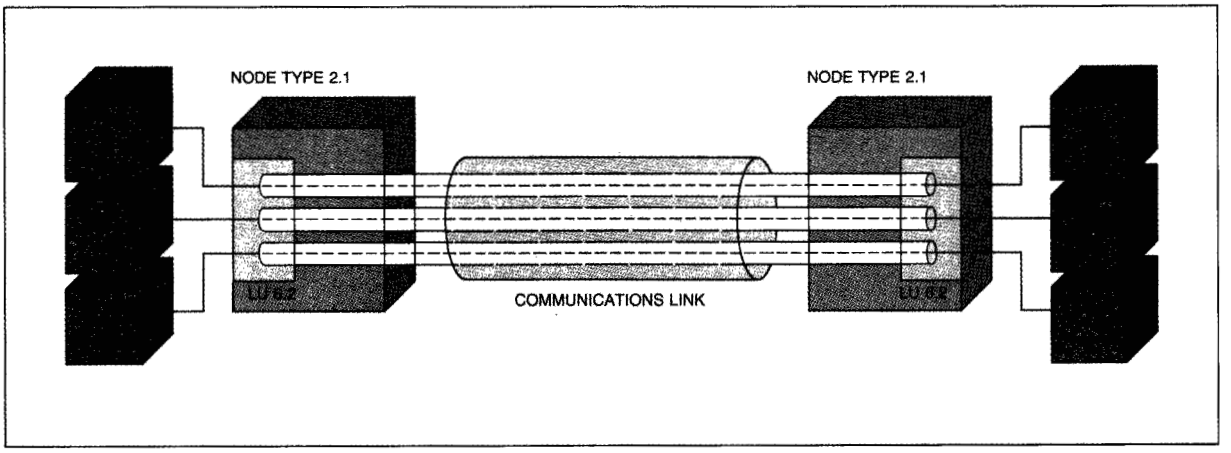
Figure 1 shows two System/36s directly connected by a communication link. LU 6.2 sessions are established between Logical Units[7] in the two nodes. When a program in one node communicates with a program in the other node, it uses one of the available sessions to establish a conversation. From the viewpoint of a transaction program, the Logical Unit is a port through which communications services are obtained.

APPC is the base on which a number of System/36 user services have been built. These services include the following: Distributed Data Management (DDM), which provides remote file access, Display Station Pass-Through (DSPT), which provides remote log-on capability; and SNA Distribution Services[8] (SNADS), which uses APPC transaction programs to forward data asynchronously from node to node. System/36 customers also construct their own distributed applications using APPC.

As the number of products implementing LU 6.2 and Node type 2.1 has grown, so has reliance on this means of providing logical connectivity among minicomputers and workstations. The Node type 2.1 connection protocol, however, requires that peer nodes be adjacent in order to communicate. Providing links between all pairs of communicating processors is costly for customers with large numbers of systems. System/36 APPN introduces full peer networking capability, eliminating the requirement that peer nodes be adjacent in order to communicate.

System/36 nodes configured with the APPN function are called *Network Nodes*; they can serve as inter-

Figure 1 System/36 Advanced Program-to-Program Communication



mediate routing nodes for data passing through the node en route to other destinations. Network Nodes are interconnected to form networks of arbitrary topology, as shown in Figure 2. End Nodes are type 2.1 nodes that do not provide network services[9] and are attached at the periphery of the network. Any two nodes in the network can communicate as peers. LU 6.2 application programs such as DSPT, which were previously limited to a pair of adjacent processors, will now run without change to provide services between nonadjacent peer processors.

Figure 3 shows a four-node APPN network. Session One extends from Logical Unit A to Logical Unit D via Intermediate Routing Functions in Nodes X and Y. The conversation between transaction programs which makes use of Session One has no awareness that the session is extended. The session may be thought of as a set of session segments connected by the Intermediate Routing Functions of nodes X and Y. Each segment is commonly called a hop. APPN is therefore said to provide a multihop data transport. Session Two is a single-hop session and does not require the use of an Intermediate Routing Function.

APPN networks incorporate dynamic control mechanisms. Changes in the status of nodes and links are immediately broadcast to all Network Nodes. Each Network Node uses this information to construct a view of the current network topology. Paths between nodes are constructed dynamically from this topology information. Similarly, Logical Units in the network need only be defined in a single owning Network Node. A dynamic directory function is responsible for finding the Network Node owning any particular Logical Unit.

Dynamic control has several advantages. The operator is relieved of a substantial system definition burden because only information local to a Network Node, such as directly attached links and local resources, need be defined. The network does not cease operation to incorporate such changes. There is less likelihood of failure due to an incompatible definition among nodes. The network is highly adaptive to failure and change. If a link fails, for example, sessions which used the failing link may be re-established using the best existing alternative paths.[10] Dynamic function improves data transport performance because the best route between nodes can be chosen at the time that the communications session is established.

The APPN dynamic network control function is distributed. All Network Nodes have identical algorithms for cooperating in the execution of functions such as the broadcasting network topology information and searching for network resources. This distribution of function provides significant advantages. The failure of a Network Node generally has no effect on the continued operation of the rest of the network because nodes do not perform centralized services on behalf of other nodes. An exception occurs when the only path between two nodes is through the failing node. The failure of such a node

partitions the network into two or more independent subnetworks, each of which continues to function normally. Network Nodes share the processing cost of performing network functions. When a node joins an APPN network it receives services via the network and, in exchange, performs its share of the distributed processing necessary to maintain these services.

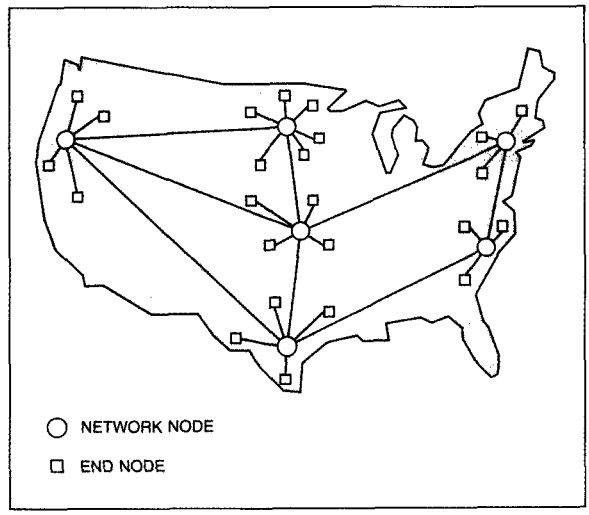
Members of the Network Architecture and Protocols group at the Thomas J. Watson Research Center had for some time been exploring areas related to dynamic peer networking such as distributed broadcast algorithms,[11] dynamic routing,[12] adaptive flow control, and deadlock prevention. Much of this work was incorporated into a proposed design for networking small systems,[13] a joint effort between the Communication Products Division and the Research Division. The design is based on the use of the existing LU 6.2 verbs and protocols for sessions while extending Node type 2.1 services to provide multihop data transport and dynamic network control. Many problems needed to be addressed. Could the design actually be implemented successfully in small processors? How would such a network perform? Could network definition and operation be made simple for the small systems environment where the user is often the operator? How do characteristics of the system environment for a particular product affect the design of algorithms and data structures for the implementation of networking software? Interest in investigating such questions, along with System/36 product interest in meeting customer requirements, motivated the development of System/36 APPN as a joint project of the IBM Thomas J. Watson Research Center and the IBM System Products Division.

APPN design overview

APPN functions may be viewed as a set of services. *Connectivity Services* performs activities related to changes in the physical connectivity of the network. Each Network Node maintains a database describing the current network topology. As a node becomes aware that its state or the state of its attached links has changed, it makes use of a distributed broadcast algorithm to spread the information throughout the network. Network Nodes then update their databases accordingly.

Directory Services identifies the node containing a specified Logical Unit. It may perform a distributed search of the network or simply verify information which has been previously cached. It is necessary to know where an LU is located in order to establish a communications session to that LU.

Figure 2 A mesh of Network Nodes with attached End Nodes



Route Selection Services identifies the best path available to a specified node in the network. A shortest-path algorithm is applied to the database of network topology which is maintained in each Network Node.

Session Activation establishes a communications session between a pair of Logical Units located anywhere in the network using the path provided by Route Selection Services. Information is established at each node in the session path so that data can later be transported on the session.

Data Transport performs the actual flow of data traffic between two session endpoints, including the flow through all intermediate routing nodes on the path. At each hop, data must be properly routed to the next node on the session path. The flow of data is throttled and messages are segmented when necessary.

User view of System/36 communications

The System/36 is designed for exceptional ease of use. All user and operator functions are menu-driven. Context-sensitive help screens are available.

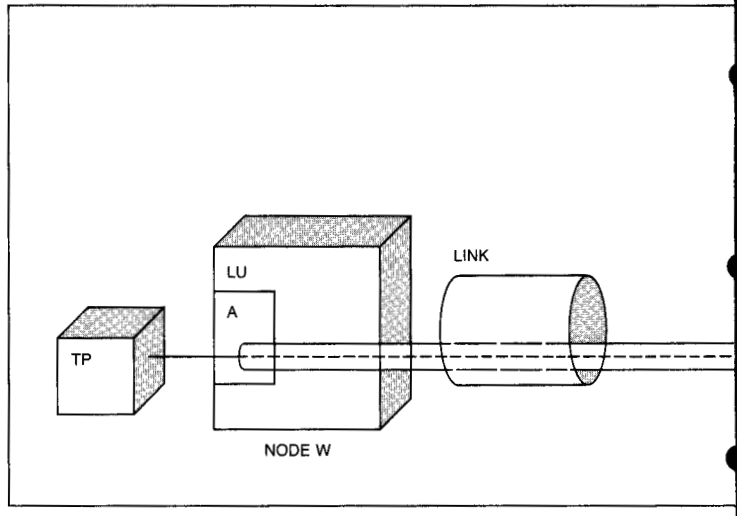
A wide variety of communications functions, called subsystems, are accessed via a single Interactive Communications Facility (ICF).[14] This allows the same basic set of menus to be used for the configuration, activation, control, and deactivation of all communications subsystems. Thus, the System/36 was an excellent environment in which to test the premise that the distributed and dynamic nature of the APPN design results in networks that are very easy to establish and operate.

Configuring the communications environment, activating and deactivating communications resources, conversing with programs in remote nodes, and monitoring communications status are frequently performed communications activities. The sections that follow describe a functional view of each of these activities and the changes which have been made in order to integrate the APPN subsystem into the existing System/36 communications framework. Although the changes have been introduced because APPN requires a different functional model from other subsystems, they have the effect of enhancing usability and substantially reducing system definition.

Line configuration. Configuration is performed at a System/36 display station via menus. The user defines a Line Configuration for each communications line attached to the System/36. The Line Configuration specifies characteristics of the line and the Remote Nodes attached to the line. A simplified Line Configuration is shown for one of the three lines illustrated in Figure 4. APPN required no changes to Line Configuration since the existing menus provided information about adjacent nodes and links, and APPN network control functions were designed to dynamically obtain this information for nonadjacent nodes and links.

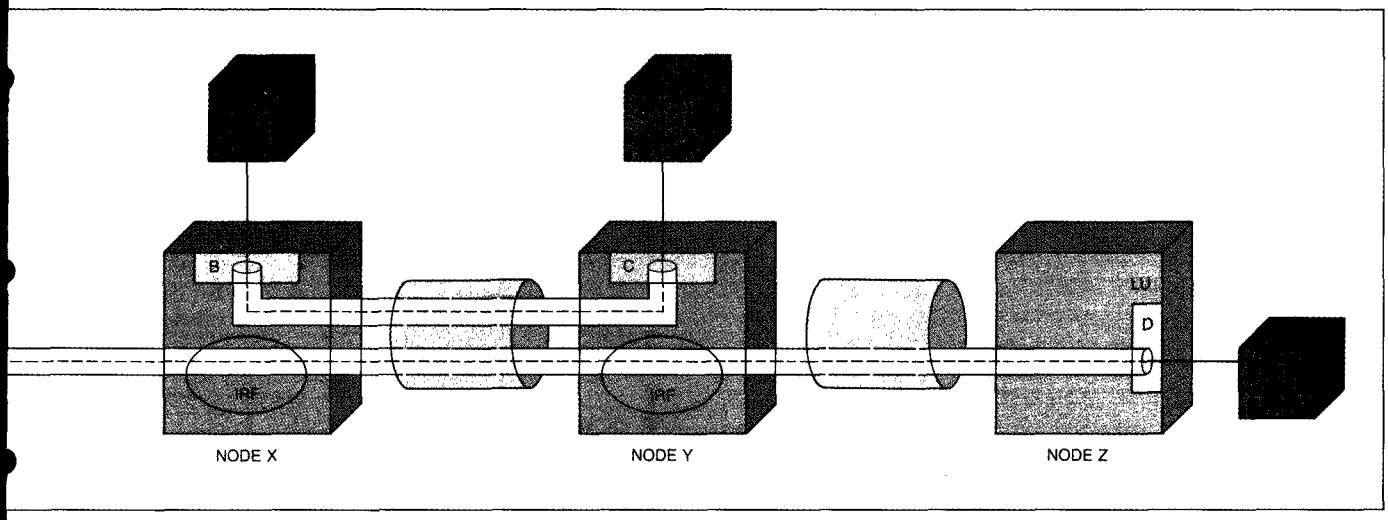
Subsystem configuration prior to APPN. The user defines a Subsystem Configuration describing each communications Subsystem which is to be activated on a line. The Subsystem may be thought of as the Local Logical Unit. The Subsystem Configuration names Remote Logical Units to which the Local Logical Unit can establish communications sessions. The Subsystem Configuration names a specific Line Configuration and can only be used to establish communications on that line. Each Remote Logical Unit is mapped to a specific Remote Node named in that Line Configuration. Associated with each configured Remote Logical Unit the user defines a number of Modes that specify a set of session characteristics, such as maximum allowable message size, for sessions associated with that Mode. The Subsystem Configuration can be viewed as describing logical connectivity. An example is shown in Figure 5. The operator at node YKTN must perform a subsystem configuration for each of the local Logical Units JIM, ALAN and BOB. The subsystem configuration for local Logical Unit ALAN must include descriptions of remote Logical Units MARY, ALEX, and JUNE. A set of Mode definitions, not shown in the figure, must be associated with each of the remote Logical Units.

Figure 3 System/36 Advanced Peer-to-Peer Networking



Readers familiar with the System/36 are aware that the terms Logical Unit, Node, and Mode do not appear on configuration menus. They are replaced by the terms Location, System, and Session Group. This is done so that the same terminology can be applied to the configuration of both SNA and non-SNA Subsystems. It would be confusing if an SNA term such as Logical Unit were applied to non-SNA subsystems. System/36 therefore uses non-SNA terminology to describe all configurations. SNA terminology is used consistently in this paper.

APPN subsystem configuration. APPN simplifies Subsystem Configuration. It is not possible to associate an instance of the APPN subsystem with a single line, as is done in other subsystems. Data received on one line may be forwarded on a different line. The APPN Subsystem Configuration does not, therefore, specify one Line Configuration with which it is



to be associated. A single APPN Subsystem Configuration is applied to all Line Configurations on which APPN is active. The association between the Subsystem and the line is made at the time that the Subsystem is made active on the line rather than at the time of configuration.

In the APPN environment, it is not necessary to configure Remote Logical Units[15] since Directory Services dynamically associates Remote Logical Units with the Nodes in which they reside. Exceptions are those Logical Units which reside in adjacent End Nodes. The identity of these Logical Units cannot be learned dynamically since Directory Services does not extend to the End Node.

If Remote Logical Units are no longer configured, it follows that Mode definition can no longer be performed for each Logical Unit. The user instead defines a master list of Mode names and associated characteristics which serve all Remote Logical Units. This significantly reduces system definition requirements at the cost of some small loss of granularity in the specification of session characteristics. An example of the simplified APPN Subsystem Configuration is shown in Figure 6. One local Logical Unit, BOB, is defined. It serves all lines on which the APPN subsystem is active. The only remote Logical Units requiring definition are in the adjacent End Node, ROCH. Information about other Logical Units is learned by means of APPN's dynamic directory function. Note that the APPN network of Figure 6 is larger than the pre-APPN network shown in Figure 5, but its system definition requirements are substantially smaller.

When defining Modes, System/36 users have the option of specifying a number of pre-established sessions. These sessions become active when the Mode becomes active and they remain active whether or not they are being used. Such sessions are useful because they are available immediately to transaction programs without requiring time for session activation. In the APPN environment, maintaining sessions for long periods of time carries a penalty. These sessions will not be able to make use of path improvements caused by changes in network topology. While pre-established sessions may be available quickly, they may not perform as well as sessions established as needed.

Activation of Subsystems other than APPN. A Remote Logical Unit must be explicitly activated before any communications sessions can be established to that Logical Unit. Activation is performed by issuing an ENABLE command naming the Logical Unit or an entire Subsystem Configuration. In the latter case all Logical Units named in the Configuration are activated. The first ENABLE of a Remote Logical Unit belonging to a particular Subsystem activates the Subsystem itself. Any Modes to be used for sessions to the Remote Logical Unit must also be activated. This is done by issuing a STRTGRP command naming the Mode and associated Logical Unit.

Figure 4 Line configuration

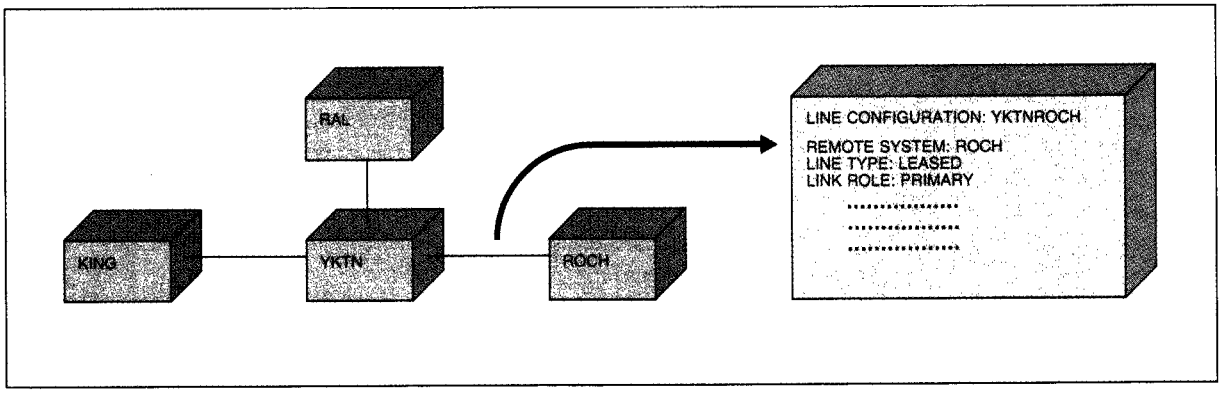
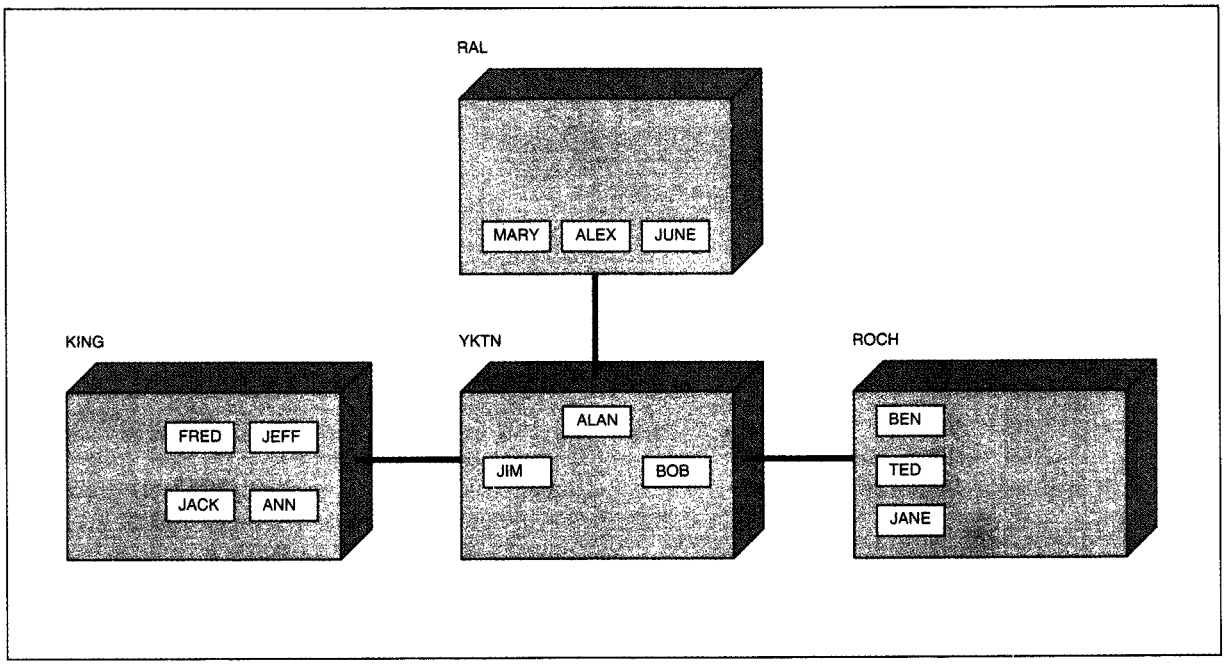


Figure 5 Pre-APPN subsystem configuration



The user does not explicitly request line activation. In the non-APPN environment, there is only one link on which the Logical Unit may be reached so the link can be activated automatically when the Remote Logical Unit is activated.

APPN Subsystem activation. The above activation model changes in the APPN environment. In an APPN network the number of Remote Logical Units and associated Modes could grow quite large. It would be an unreasonable burden on the user to perform explicit activation and deactivation. This requirement has therefore been eliminated for APPN. Logical Units and Modes are dynamically activated as they are needed. The ENABLE command is still used in the APPN environment, but its primary purpose is

the activation on lines in which the APPN subsystem is being used.

Conversation. Transaction programs are application programs which have embedded verbs for program-to-program communications. Examples of such verbs are ACQUIRE, which obtains a session to a specified Logical Unit, EVOKE, which establishes a conversation over the session, and PUT, which sends data from one program to another on the established conversation.[16] Transaction programs which were written to run on the APPC subsystem will run without change on the APPN subsystem. Conversation protocols are performed only at the endpoints of a session. They are not sensitive to whether the underlying session is a single-hop session established by the APPC subsystem or a multihop session established by the APPN subsystem.

Monitoring. System/36 provides a number of Status functions that allow the user to view communications activities. The status of links, Logical Units, Modes, and sessions can be displayed. Although these displays have been modified to reflect changes in the APPN functional view, they do not provide the global information which is necessary to manage and troubleshoot a network. APPN uses the fact that network topology is maintained in every Network Node in order to provide useful network management information. A utility called APPNINFO accesses the topology database in order to display the current status of nodes and links throughout the network.

System/36 communications structure

The System/36 provides a multiuser, multitasking environment via its operating system, the System

Figure 6 APPN subsystem configuration

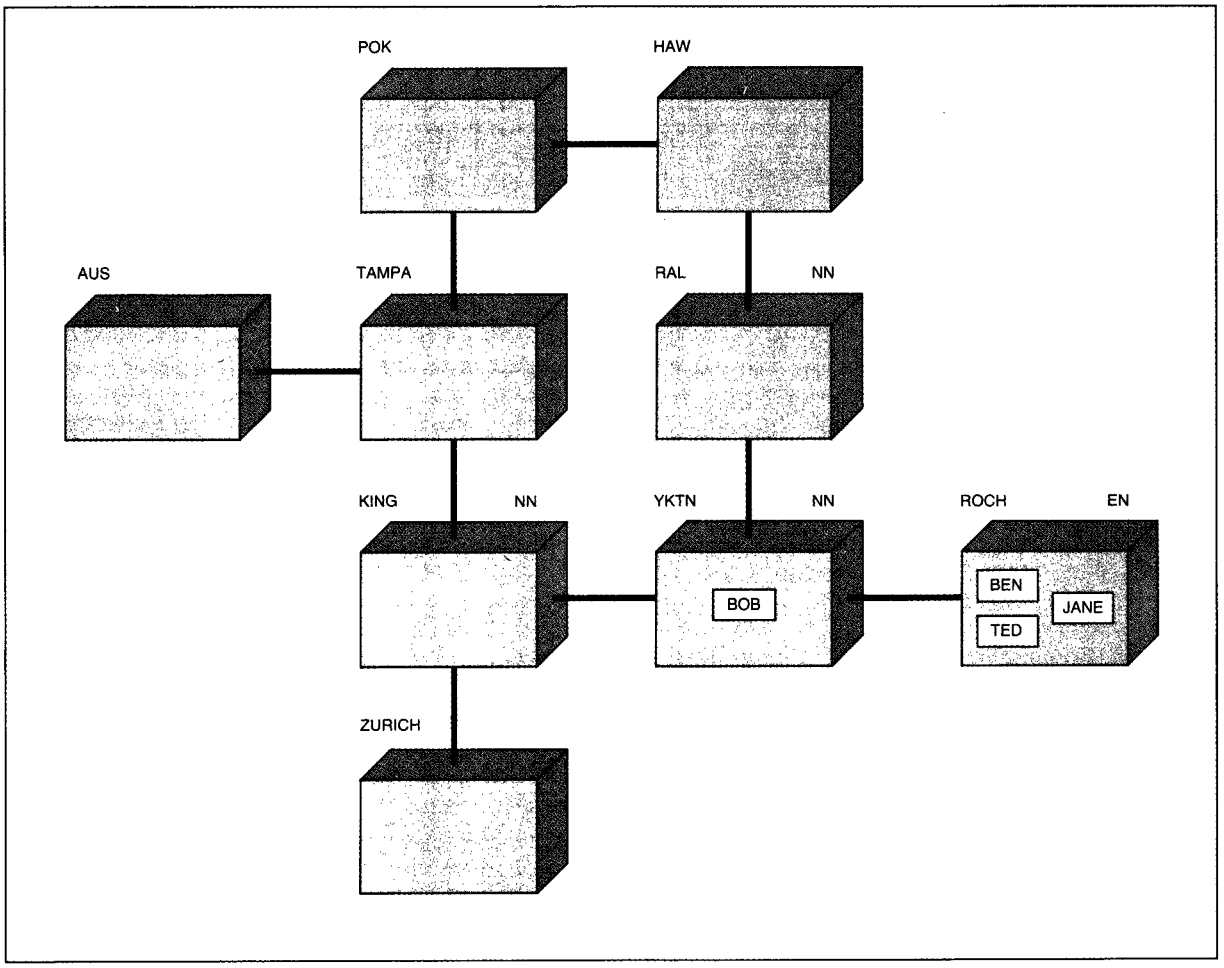
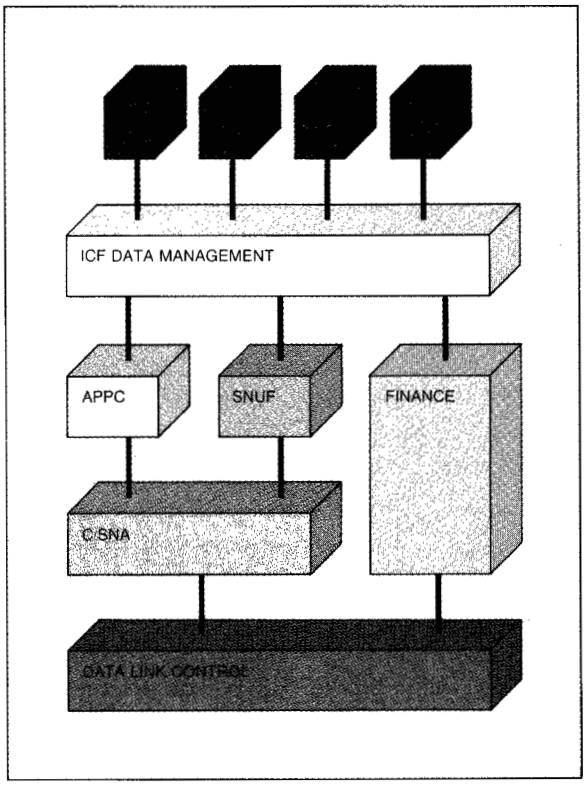


Figure 7 System/36 communications structure



Support Program (SSP). Systems can have up to seven megabytes of main storage and 1.6 gigabytes of disk storage. Ten communications lines are available with line speeds of up to 56K bits per second. IBM Token Ring, X.25 packet network, Binary Synchronous Communications (BSC), and Synchronous Data Link Control (SDLC) are supported line protocols.

Task size is limited to 64K bytes. Tasks requiring additional space make use of transients which overlay portions of the task when the functions they perform are needed. Much use is made of transients in both the operating system and communications subsystems.

Real storage allocated to tasks by the operating system is called System Queue Space (SQS). Virtual storage, called Task Work Space (TWS), is also available. An area of disk is designated as a Work Space. The operating system allocates TWS to tasks out of a Work Space in much the same manner as it allocates SQS out of its pool of available real storage. Virtual storage must be mapped to real storage before it can be referenced in a program. Mapping requires a disk access unless the storage to be assessed has remained in real storage since an earlier mapping operation.

A portion of the System/36 communications structure is shown in Figure 7. The figure shows three SNA subsystems which have transaction programs as their end-users.[17] Each box represents a task. Messages are passed between tasks via operating system queues.[18] The top row of boxes represents transaction programs. ICF Data Management routes verbs issued by the transaction programs to the subsystems for which they are intended. In the case of the Finance Subsystem,[19] all of the SNA support except the Data Link Control[20] (DLC) is contained in a single task. The other two subsystems shown split this support between two tasks. The SNA Upline Facility (SNUF) allows communication with host subsystems.[21] It provides transaction programs with a different set of functions than the APPC subsystem. The interpretation of such functions is performed by the Presentation Services layer of SNA. Presentation Services are therefore implemented in separate SNUF and APPC tasks. Lower-level functions, such as the pacing and segmentation of data messages, are common to the two subsystems. These functions are placed in a single task, which is called Combined SNA (C/SNA).[22] C/SNA implements the Data Flow Control, Transmission Control, and Path Control layers of the SNA architecture. There is only a single instance of the C/SNA task, and it provides single-threaded execution of communications functions. When C/SNA is posted with an external event, such as the receipt of a message from ICF Data Management or a message segment from the DLC, that external event is completely processed before the next external event is accepted for processing.

APPN implementation overview. APPN Session Establishment, Data Transport, and portions of Connectivity Services are implemented as modifications to existing communications support. Most of the change occurs in the C/SNA task where the SNA functions to be extended are located. C/SNA is divided into subcomponents. Three new subcomponents were created: Session Connection performs the Intermediate Routing Function and other Data Transport functions, Session Connection Manager performs intermediate session establishment, and Node Buffer Manager dynamically allocates buffer resources to sessions. Session Connection is written as part of C/SNA mainline processing in order to perform the Intermediate Routing Function with a minimum of delay. Session Connection Manager and

Node Buffer Manager are written as transients and are obtained from disk as needed.

In addition to the new components described in the previous paragraph, modifications were also made to existing C/SNA subcomponents. The changes were implemented so as to be transparent to the other subsystems which share C/SNA. A software switch setting in the C/SNA task indicates whether the APPN modifications should be executed, in which case the node functions as a Network Node, or should not be executed, in which case the node does not perform network services. In the latter case the node may be attached to an APPN network as an End Node. The node may not perform both roles simultaneously. The choice is made at the time of system configuration.

There is no new APPN task equivalent to the APPC or SNUF tasks. The APPC and SNUF tasks perform Presentation Services functions. APPN uses APPC Presentation Services; this is what allows APPC transaction programs to run without change in the APPN environment.

APPN Directory Services, Route Selection Services, and the portion of Connectivity Services which maintains the Topology Database are implemented as two new application tasks. They are called Directory Services (DSS) and Route Selection and Topology Database Update (RSS). Two additional transaction tasks, Send and Target, allow control information to be exchanged with adjacent Network Nodes. The Send task establishes an LU 6.2 conversation with a Target task in an adjacent Network Node when it has network control information, such as a topology database update message or a directory search message, to send. In order for the Directory task in one Network Node to send a message to the Directory task in an adjacent Network Node, it posts the message to the local Send task. The Send task establishes an LU 6.2 conversation to a Target task in the adjacent node and sends the message. The Target task then posts the message to the destination Directory task. A fifth task, the Control Point Manager (MGR), is responsible for synchronization and coordination among the network control tasks. The five tasks are known collectively as the Control Point.

If only a single session were used to carry control messages between adjacent Network Nodes, it is possible that the Send tasks in the two nodes would contend for the use of the session. This would happen, for example, if the two nodes attempted to send directory search messages to each other simultaneously. In order to avoid such contention, two sessions are maintained for the exchange of information between adjacent Control Points, one for each direction of data flow. These two sessions may be viewed as simulating a single full duplex session. Conversations between adjacent Control Points are started as needed, but the sessions they use are maintained as long as the link between the two Control Points remains intact. Figure 8 shows Control Points in two adjacent nodes and the pair of sessions on which they broadcast network control messages.

All of the Control Point tasks except the Target task become active when the APPN subsystem becomes active. An instance of the Target task becomes active each time a Send task in an adjacent node establishes an LU 6.2 conversation for the purpose of broadcasting control information.

Implementation issues

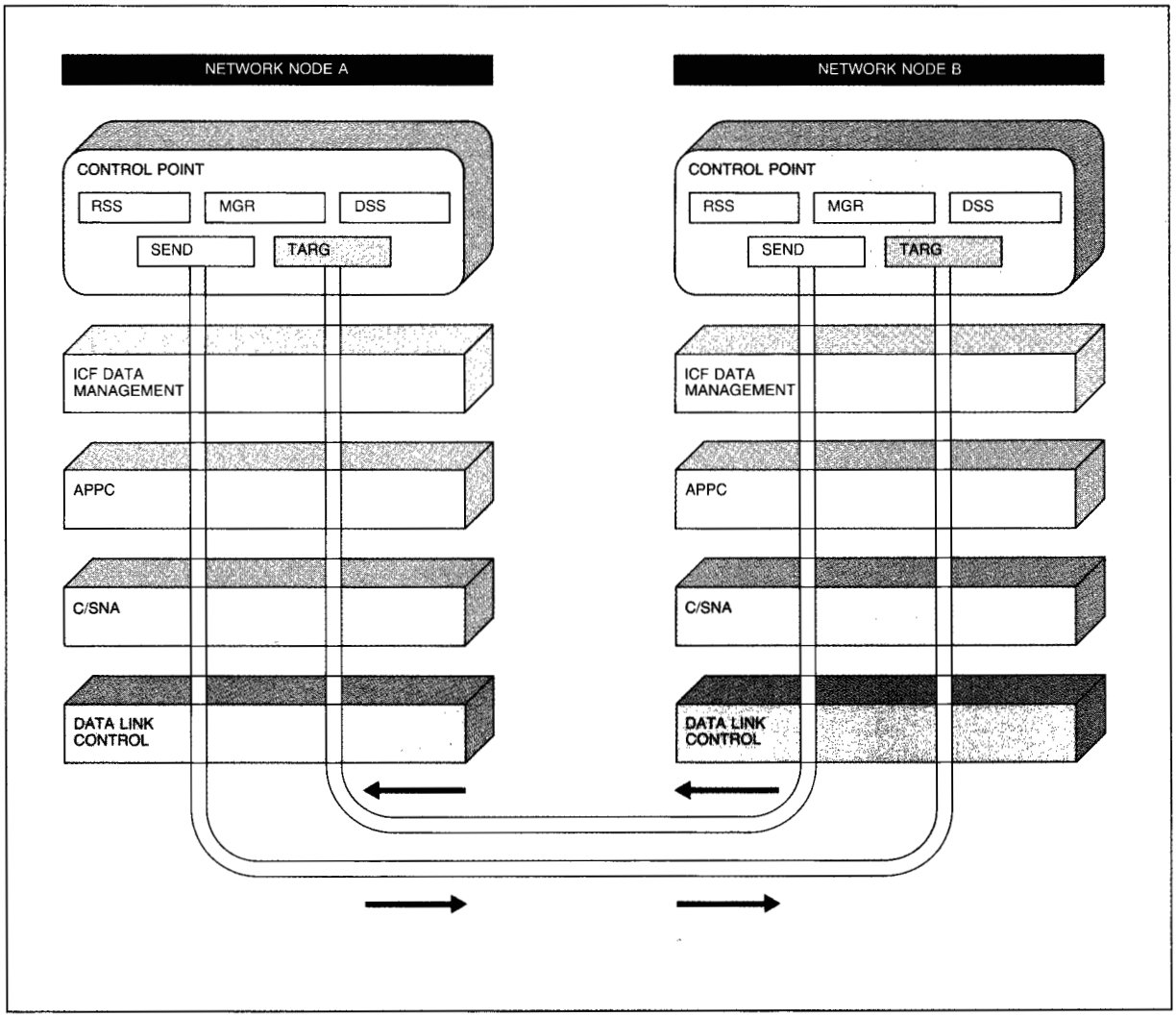
Characteristics of the System/36 environment had considerable influence on the APPN implementation. This section describes portions of the implementation with particular emphasis on those issues which are specific to the System/36 environment.

Link activation. As described earlier, link activation for subsystems other than APPN is transparent to the user. C/SNA activates a link when a Remote Logical Unit associated with that link is activated. APPN differs from other subsystems in that it does not associate a Remote Logical Unit with any particular link. The Node in which the Logical Unit resides might be reached by a variety of paths. In order to use C/SNA's existing link activation function, APPN creates one Remote Logical Unit associated with each link on which the APPN Subsystem has been ENABLEd. This Logical Unit has the same name as the Remote Node associated with the link. The association between the Logical Unit and the link is used only for purposes of link activation. Route Selection Services ignores the association when computing paths.[23]

The creation of a Remote Logical Unit associated with the link allows C/SNA's existing link activation function to be used by APPN. The operator continues to use the familiar ENABLE command, but its primary purpose in the APPN environment is link activation rather than Logical Unit activation.

Endpoint session activation. Communications sessions are established by sending a session activation message (BIND) from one endpoint of the session to

Figure 8 Sessions between APPN control points

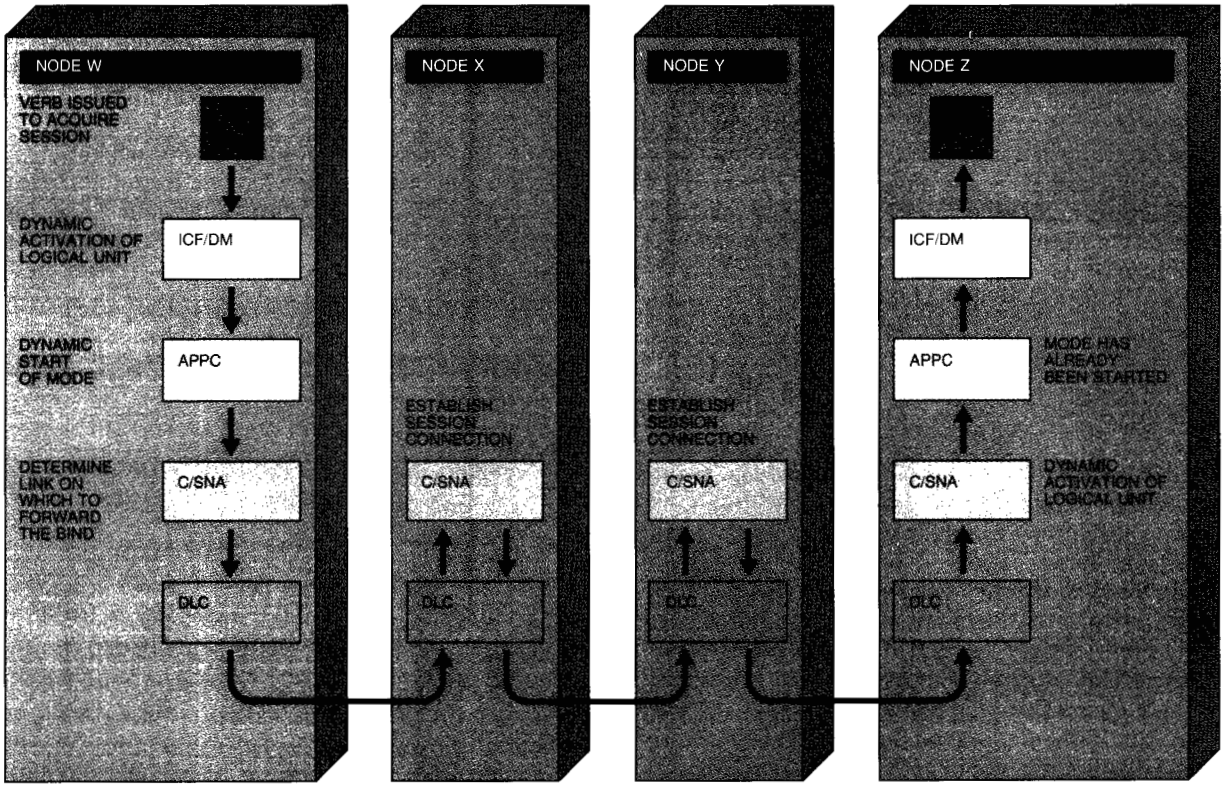


the other and receiving a response message in reply. The BIND response allows session parameters to be negotiated and indicates that activation has been completed. Data structures required for the transport of data on the session are established at each node in the session path.

Prior to the implementation of APPN, C/SNA provided support to the APPC subsystem for the establishment of single-hop LU 6.2 sessions. APPN uses this session activation function, with some modifications, at the endpoint nodes of multihop sessions. Session activation in intermediate nodes is a new function that is described later.

The session activation function, which existed in C/SNA prior to APPN, expected a specific environment to be present at the time of its invocation. It required (1) an active Remote Logical Unit, (2) an active Mode, and (3) knowledge of the link on which the BIND was to be sent. The first two requirements were met because the operator performed explicit Remote Logical Unit activation by issuing an ENABLE command and explicit Mode activation by issuing the STRTGRP command. The third requirement was met because there was only one link on which any particular Remote Logical Unit could be reached. In the APPN environment the operator is not required to explicitly activate Remote Logical Units or Modes

Figure 9 Multihop session activation



and the link on which the BIND is to be sent depends on the route which is computed for the requested session.

The following paragraphs describe the sequence of session activation events which occur at session endpoints, including the new functions which have been inserted to allow activation to proceed in the APPN environment. The sequence is shown graphically in Figure 9. The description begins with a transaction program requesting the use of an LU 6.2 session in order to establish a conversation with a Remote Logical Unit. The figure depicts the events which occur as a BIND flows from session origin to destination.

Dynamic activation of Remote Logical Units. A transaction program issues an ACQUIRE verb to ICF Data Management indicating that a session has been requested. The Remote Logical Unit named in the verb is matched against a list of active Logical Units in order to route the request to the proper Subsystem. If no Logical Unit is found, it is assumed that the Remote Logical Unit is associated with the APPN subsystem and should be activated dynamically.[24] The Logical Unit to be activated might be anywhere in the APPN network, and there is no adjacent link with which it is associated. Dynamic activation of the Implicit Logical Unit involves little more than the creation of a data element naming the Logical Unit and serving as a place to anchor sessions associated with that Logical Unit. The System/36 user can observe whether a Logical Unit has been activated dynamically by issuing the Subsystem Status (DI) command at a display station. Dynamically activated Logical Units show a dashed line where the associated line number would ordinarily be displayed.

Dynamic activation of modes. The session activation request is passed from ICF Data Management to the APPC task. The APPC task keeps a count of the number of sessions activated per Mode to ensure

that session limits are not exceeded. This processing depends upon the existence of an active Mode. The APPC subsystem requires that Modes be explicitly activated by operator command prior to session activation. APPN allows the dynamic activation of Modes during session activation. Session activation is suspended while a function called Change Number of Sessions (CNOS) is performed. A CNOS transaction program establishes a conversation with a CNOS transaction program at the Remote Logical Unit. Session limits are negotiated and the named Mode becomes active at both the origin and destination Logical Units.[25] When Mode activation is completed, the APPC task can increment its count of active sessions and verify that session limits have not been exceeded.

Where to forward the BIND. The session activation request is next passed from the APPC task to the C/SNA task. A Remote Logical Unit and Mode are active, but the link on which the BIND should be sent is still unknown. C/SNA requests that the Control Point provide a route through the network to the node in which the destination Logical Unit resides. Directory Services resolves the named destination Logical Unit to the name of the node in which it resides. Route Selection Services computes the optimal path to the Remote Node based on a characteristic, called the Class of Service (COS), associated with the Mode. C/SNA examines the route to find the first link in the path. This is the link on which the BIND is to be sent. With an active Remote Logical Unit and Mode, and knowledge of the link on which the BIND is to be forwarded, C/SNA is able to use the LU 6.2 session activation function which served the APPC subsystem prior to the introduction of APPN.[26] This is one of many instances where System/36 APPN was able to build upon existing function provided by C/SNA. The fact that APPN is designed as an extension to Node type 2.1 significantly reduces the effort required to implement APPN when the type 2.1 node already exists.

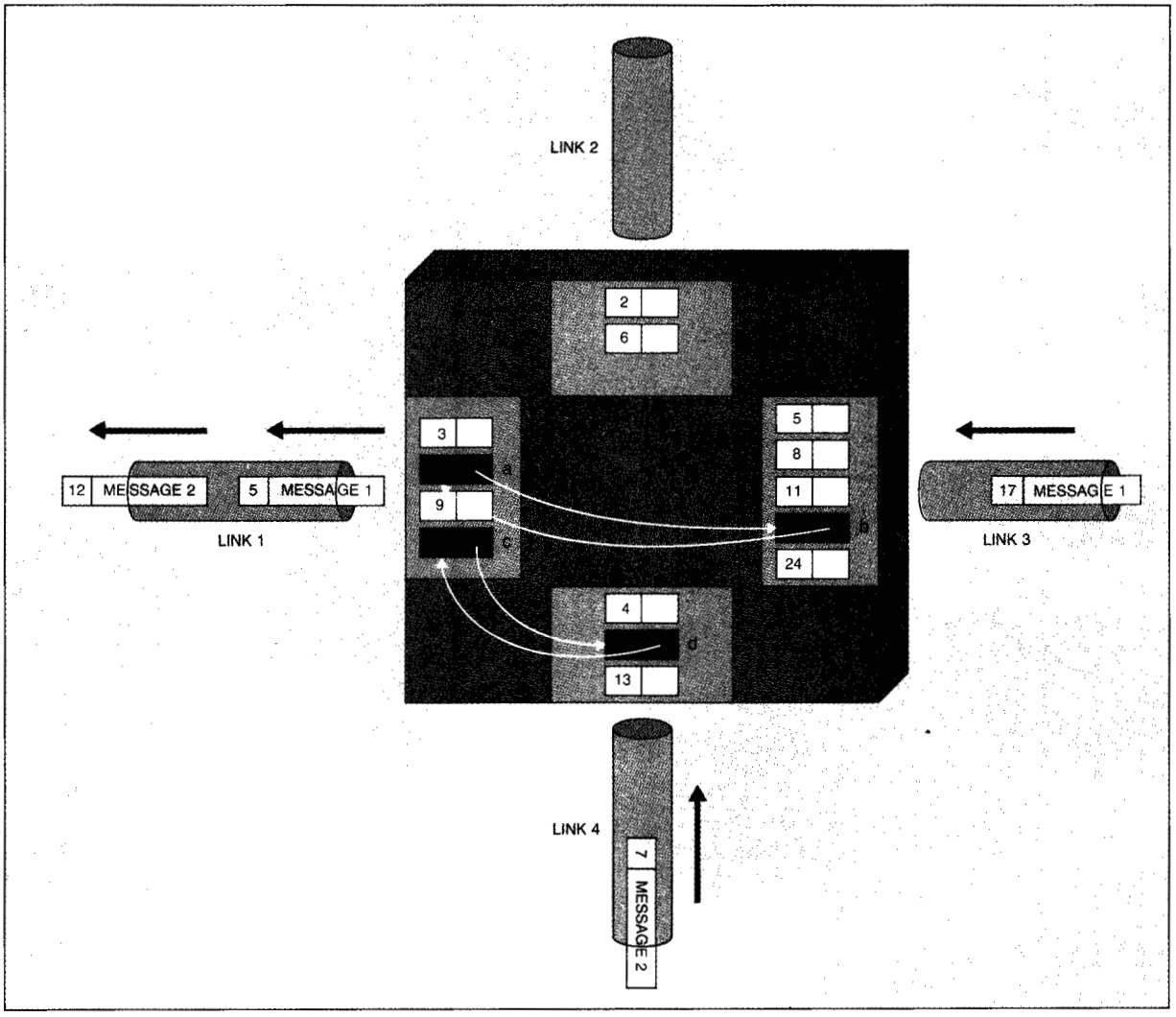
Session activation at the destination. Making use of the activation code at the session destination is somewhat simpler. When Mode activation is performed dynamically at the session origin, the Mode is also activated at the destination. It is therefore unnecessary to dynamically activate the Mode at the destination. The link on which the BIND response flows is the same link on which the BIND was received. It is therefore unnecessary to identify the link on which the BIND response is to be sent. The only remaining requirement is the dynamic activation of the Remote Logical Unit. The node receiving a BIND may know nothing about the Logical Unit which originated the BIND. When the BIND reaches C/SNA via the DLC, C/SNA dynamically activates the Remote Logical Unit in a manner similar to the way in which the ICF task does this in the session origin node.

Intermediate session activation. Session activation at Intermediate Routing Nodes performs different functions than the activation at Session Endpoint Nodes. Activation in the Intermediate Routing Node must establish the environment in which data messages can be passed through the node. A session passing through an Intermediate Routing Node is represented by a pair of data elements. One element represents the session segment or hop over the link on which the BIND has been received, and the other represents the segment on which the BIND is forwarded. This pair of data elements is called a Session Connector. The receiving element saves the identity of the inbound link and a 17-bit address assigned by the previous node and carried with the BIND. The sending element saves a new 17-bit address which identifies the next segment of the session and points to the link on which the BIND is forwarded. The identity of this link is determined by examining the next element in the path which is carried in the BIND. The receiving and the sending data elements point to each other. Intermediate node session activation provides the environment which allows data messages to travel through intermediate nodes of the session.

Pacing allows the receiver of data to tell the sender that a specific quantity of data may be sent.

After the session has been established and data messages begin flowing, the Session Connector is used to route messages through the Intermediate Routing Node. The 17-bit address carried in the header of the message and the link on which the message is received uniquely identify a data element. This data element is one of a pair of data elements forming the Session Connector. The partner data element

Figure 10 Session connection



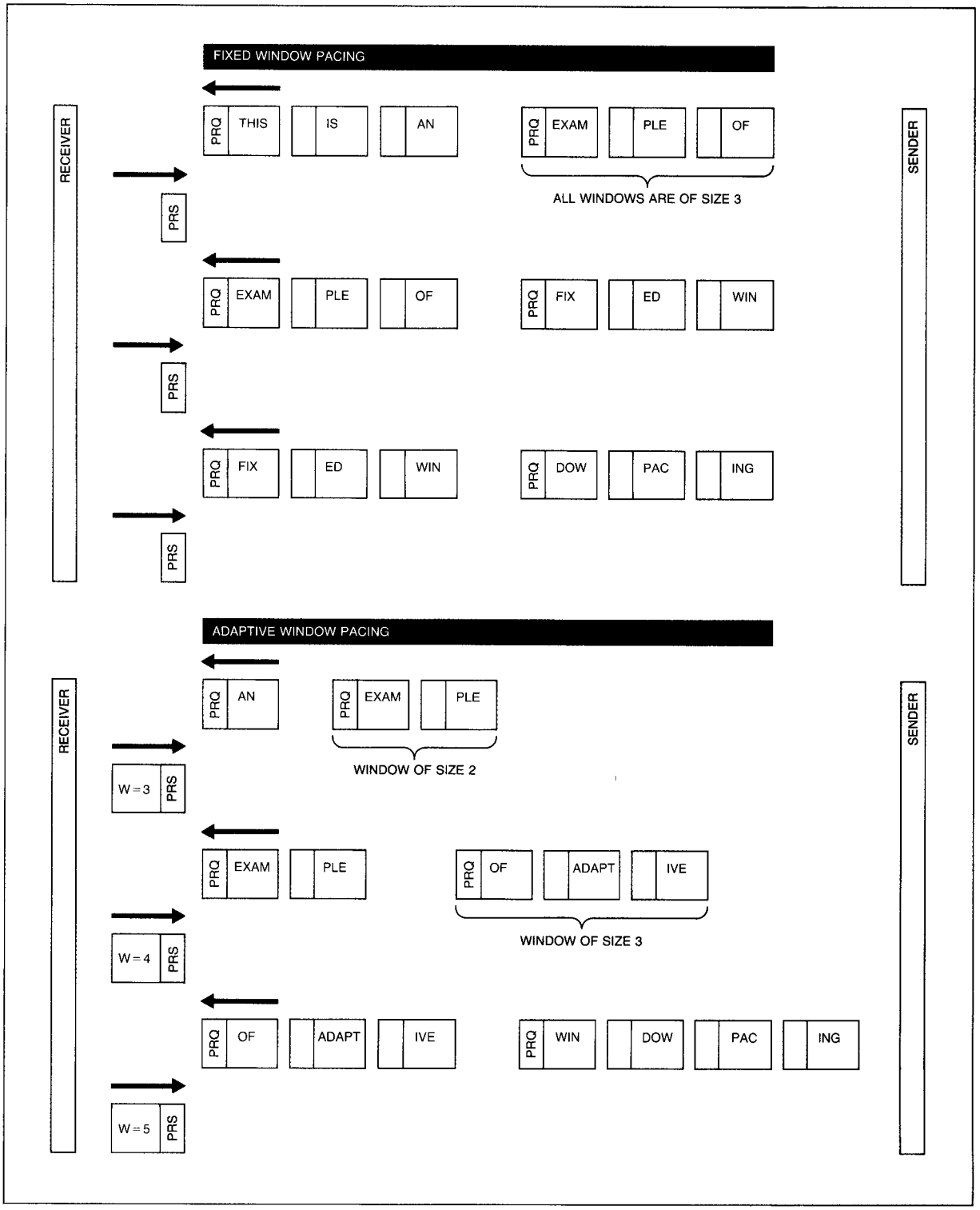
provides the address carried by the message on its next hop and the identity of the link on which the message is to be forwarded.

An example is shown in Figure 10. A message arriving on link 4 searches through the pool of session data elements associated with link 4 in order to find one with an address of 7, matching the address carried in its header. This data element, labeled **d** in the figure, points to a partner data element, labeled **c**, associated with link 1. The pair of data elements, **c** and **d**, form a Session Connector. The Session Connection function changes the address in the message from its inbound address of 7 to its outbound address of 12 and forwards it on link 1, as indicated by the Session Connector.

Flow control and buffer management. Data messages flowing on a session are *paced* on each hop or segment of the session path. Pacing allows the receiver of data to tell the sender that a specific quantity of data may be sent. The receiver guarantees that sufficient storage is available to buffer the data on arrival.

Pacing is performed independently for each of the two directions of data flow on the session segment.

Figure 11 Fixed vs. adaptive pacing



Each instance of pacing has a *sender* and a *receiver.* The receiver grants the sender permission to send a specified number of messages. The number of messages is known as the *pacing window.* The receiver must ensure that sufficient storage will be available to buffer the messages when they are received. The exact size of the messages is not known by the receiver, so it reserves storage for the largest possible quantity of messages which can be sent on the session. The maximum size of messages is fixed at the time of session establishment. By granting permission to send data only when there is sufficient storage to buffer the data, the receiver maintains complete control over the flow of data into the node.

Adaptive pacing facilitates more efficient buffer management.

The pre-existing System/36 APPC subsystem, which allowed only single-hop data transport, implemented fixed window pacing. When the sender asked the receiver for a new window, the receiver always granted a window of a particular size. The amount of storage required to buffer received data was therefore fixed and could be allocated at the time of session establishment.

Fixed pacing is not an attractive alternative in the APPN environment. An Intermediate Routing Node may have many sessions routed through it. Each of these sessions would require sufficient storage to receive data transmitted in both directions. Buffer requirements to support fixed windows could exhaust available storage in Intermediate Routing Nodes, thus limiting the number of sessions supported through the node. If the problem is solved by reducing the size of the fixed windows, this may result in poor utilization of available bandwidth on the communications links. Sessions may wait for permission to send data while links are idle.

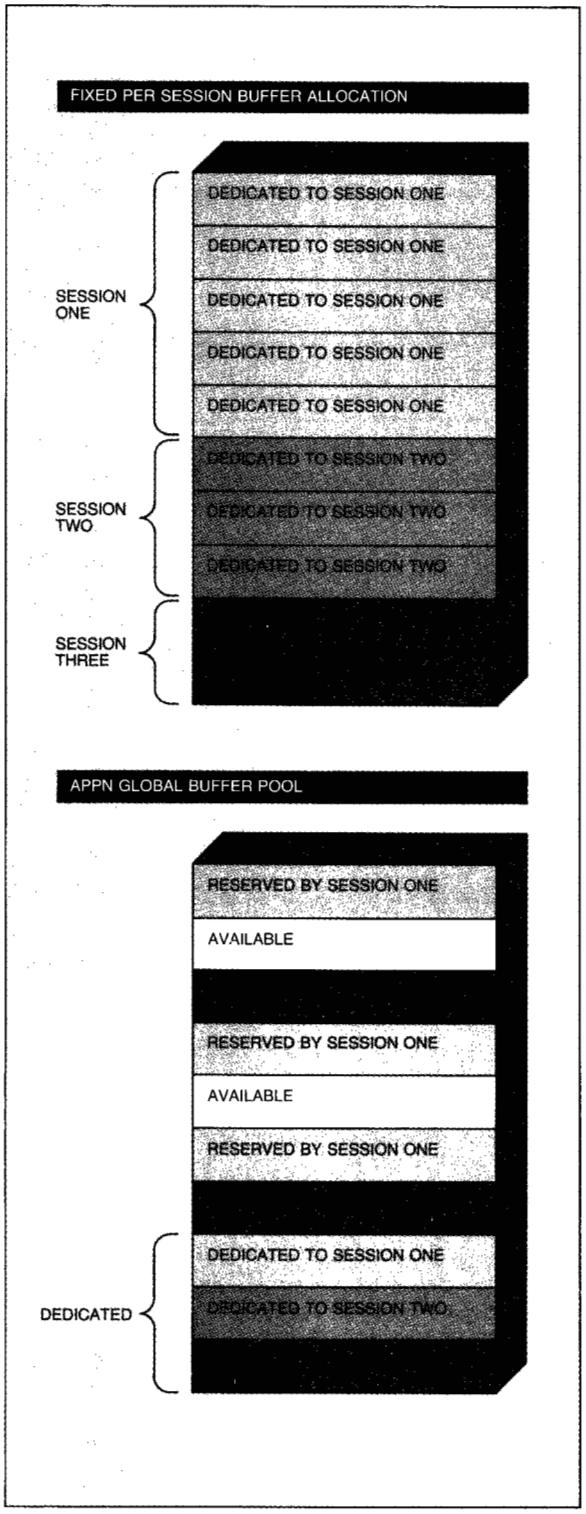
The solution used in APPN is adaptive window pacing. The pacing window is allowed to vary. Window size is computed by the receiver on the basis of the sender's utilization of previous windows, the availability of buffers in the receiver, and the level of congestion in the receiver as measured by the number of received messages which have not yet been processed. Storage available in an Intermediate Routing Node is divided fairly among the sessions according to these criteria.

Figure 11 shows a comparison of fixed and adaptive pacing. In both cases the sender asks for a new window with a pacing request (PRQ) attached to the first message of each window. The receiver grants the request with a pacing response (PRS). In the case of fixed pacing, the size of the window is assigned at session initiation. The pacing response always grants a window of the assigned size. In the case of adaptive pacing, the pacing response carries the size of the window being granted.

Adaptive pacing facilitates more efficient buffer management, as shown in Figure 12. The figure shows two ways of managing a pool of ten buffers. Fixed pacing, shown at the left, requires that each session be allocated a fixed number of buffers dedicated to that session. Because the window size is fixed, buffers cannot be shifted from one session to another according to current need. A request for a new session cannot be honored since buffers are not available. Adaptive pacing, shown on the right, requires that only one buffer be dedicated to each session for the purpose of deadlock prevention. Other buffers are distributed among the sessions in such a way as to achieve high buffer utilization. The association between buffers and sessions is dynamic. A change in the buffer allocation of a session is reflected in the size of the window it grants. In the example shown, two buffers are available to establish new sessions or to increase the windows of current sessions. If these buffers are allocated, it is still possible to start a new session by decreasing the buffer allocation of one of the existing sessions.

The sender's participation in the window size determination is an important feature of APPN flow control. The sender requests a new window when it sends the first message of its current window. When the receiver gets the request, it sends back a reply which includes the window size. Between the time that the sender requests the new window and the time it receives the reply, the sender may send messages using its current pacing window. When the sender receives the response to the window request, it takes the opportunity to gather some useful infor-

Figure 12 Dedicated vs. pooled buffer allocation



mation. The sender looks at its queue of outbound messages. If there are messages to be sent and if the current pacing window has been exhausted, this indicates that the sender could make use of a larger window. The sender remembers this, and on the next window request it asks the receiver to provide a larger window. The receiver may not grant this request. It is just one piece of information used by the receiver in determining window size.

The adaptive pacing function was added to the Connection Point Manager subcomponent of C/SNA, which previously performed only fixed window pacing for session endpoints. It was also added to the new Session Connection subcomponent of C/SNA, which performs data transport through intermediate nodes on a session path. Another new component, the Node Buffer Manager, is responsible for determining the window size that the receiver grants to the sender. The rules for determining window size for a session are quite simple. The queue of messages received, but not yet processed, is examined. If this queue is long, the window is one less than the previous window. If the queue is short and the sender has requested a larger window, the new window is one larger than the previous window. In all other cases, the window size is the same as that of the previous window. These rules may be pre-empted by an additional set of rules related to buffer availability. The structure of APPN buffer management and the manner in which buffer availability may determine window sizes are described in the sections which follow.

Managing real storage. The Node Buffer Manager maintains a 10K-byte pool of real storage and a 1M-byte pool of virtual storage. It does not physically allocate storage to sessions from its pools. The operating system performs the physical allocation from either real or virtual storage when data messages actually arrive in the node and must be buffered. The Node Buffer Manager maintains an accounting of the real storage which has been allocated to contain arriving data and of the virtual storage which has been reserved at the time that pacing windows are granted.

The tracking of real storage usage is done for performance purposes. It is much more efficient to buffer data in real storage than in virtual storage. Real storage is, however, a scarce resource in the System/36. If too much real storage is used by APPN, the performance of operating system functions may be seriously degraded. The Node Buffer Manager's 10K pool of real storage places a limit on the amount

of real storage which can be used by APPN data transport. A threshold value is associated with the pool. When usage crosses this threshold, the node is said to be in a state of real storage depletion. When the Node Buffer Manager is in this state and a request is made to supply a session with a new window value, the request is honored in a manner similar to any other window request. No special attempt is made to reduce the size of the window that is associated with the session requesting the new window. There is no reason to believe that the session making the request is a significant cause of the node being in depletion state. Instead, a search is performed to find the session which currently holds the largest window. The next time a window request is made for this greedy session, the window granted will be one less than its current window. In this way, window sizes are gradually reduced until real storage usage drops below the threshold. This scheme does not prevent the 10K real storage limit from being reached, but it does reduce the likelihood of this occurrence. If the limit is reached, messages received above the threshold are buffered in virtual storage and some performance degradation is observed. A goal of buffer management is to maintain high throughput by adjusting window sizes so that real storage can be used for buffering messages most of the time.

Virtual storage is tracked to prevent deadlocks.

Managing virtual storage. Virtual storage is managed in such a way as to ensure that there is always storage available to receive data. This guarantees freedom from deadlock. The Node Buffer Manager reserves virtual storage for a session each time a new window is granted. Data messages may actually be buffered in real storage when they arrive. The reservation simply guarantees that if real storage is not available, there will be sufficient virtual storage to buffer the data. When the data messages are forwarded to the next node or to a user within the node, and the buffer is emptied, the buffer is reclaimed by the Node Buffer Manager. The reservation associated with the buffer is canceled and the count of available virtual storage is incremented. The Node Buffer Manager is not permitted to reclaim storage until the buffer has been physically deallocated by the operating system and is ready for reuse. Deadlock may result if the storage has been logically reclaimed but is not physically available.

Virtual storage is not as scarce a resource as real storage; however, it is not unlimited. Virtual storage reserved for APPN data transport is limited by the amount of disk storage which is dedicated for this purpose. An excessively large pool would waste disk storage resources, while a pool that is too small would limit the number of sessions supported by the node. System/36 APPN uses a one-megabyte pool. The System/36 operating environment permits the reservation of this pool of virtual storage exclusively for the use of APPN data transport, while still allowing physical allocation of blocks of storage from the pool to be performed by the operating system. Without this feature, the Node Buffer Manager would be forced to physically allocate the entire pool of storage in order to reserve it. It would then have to implement its own storage manager in order to divide the pool into smaller segments when storage is actually needed to buffer received data.

The dedicated buffer. When a session is established, it is given an initial window of one, and a corresponding single buffer is reserved from the virtual storage pool. This buffer is dedicated to the session for the life of the session. In this way every session always has at least one buffer, guaranteeing that windows will not be reduced to zero due to lack of buffers. Not only does data transport cease on a session when the window is reduced to zero, but there is also a potential for deadlock.

The first message received in a window always carries a request for a new window. The message may be viewed as occupying the single dedicated buffer associated with the session. In an Intermediate Routing Node the message is processed by forwarding it to the next node on the session path. When the message leaves the node, the buffer which contained the message is physically deallocated and reclaimed by the Node Buffer Manager. The request for a new window carried by the message is then processed. The dedicated buffer which was just reclaimed by the buffer pool is reserved once again as part of the new window being granted.

It is possible that between the time the buffer is reclaimed and the time that it is again reserved, a

window request from a different session could steal the dedicated buffer from the session to which it belongs. Since the dedicated buffer is returned to the buffer pool like any other buffer, there does not appear to be anything to prevent this. The Node Buffer Manager relies on the single-threaded nature of the C/SNA environment. This environment ensures that processing associated with one event will be complete before the processing of the next event. In this case, the reclaiming of the dedicated buffer and the subsequent reservation of the dedicated buffer cannot be interrupted by activities related to a different session. Knowledge of the single-threaded nature of the C/SNA environment has been used to reduce the complexity of the APPN implementation.

Why slam the window to zero?

Window slamming. A threshold value is maintained for the virtual storage pool, as was the case for the real storage pool. When this threshold is exceeded, the node is said to be in a virtual storage depletion state. If a window request is made while the node is in this state, the request is honored in a manner similar to any other window request. As in the case of real storage depletion, no special attempt is made to reduce the size of the window associated with the session that requested the new window. There is no reason to believe that the session making the request is a significant cause of the node being in the depletion state. Instead, a search is performed to find the session which currently holds the largest window. The window associated with this greedy session is slammed to zero by sending a special unsolicited pacing message to the sender.

Why slam the window to zero? This would seem to have the undesirable effect of stopping all new data flow from the sender. Consider what would happen if the window were reduced to some nonzero value: one, for example. In this case the number of buffers reserved by this session would actually have to be increased by one in order to provide the new window. Buffers in the current window cannot be reclaimed because the receiver has no knowledge of whether the sender has actually used part or all of the current window. Clearly, if the sender has used all of the current window but the messages have not yet arrived at the receiver, the receiver cannot allow any part of the current window to be reclaimed due to the risk of buffer unavailability and deadlock.

When the sender receives the pacing message slamming the window to zero, it sets its pacing count to zero and sends no more messages except for an acknowledgment to the receiver that the window has been shut. When the receiver gets the acknowledgment, it waits for the queue of received messages to be processed. It can then reclaim the storage associated with that part of the window which the sender did not use, since it knows that no more messages associated with this window will arrive. The receiver then sends to the sender a pacing message with a new window of one, allowing data transport to proceed on the session but with a much reduced window. The window can gradually grow in size again with subsequent window requests for the session.

The Node Buffer Manager may be viewed as a buffer reservation accounting service. Its knowledge of buffer distribution and availability allows it to compute and adjust the sizes of windows granted to sessions in order to obtain high throughput and avoid deadlock. Of all APPN components, the Node Buffer Manager is perhaps the most sensitive to the operating environment. It must be tailored specifically to the performance characteristics of each type of available storage and to the primitives available for the manipulation of these storage types.

The topology database and route selection

APPN maintains a database of network topology information in every Network Node. Synchronization of the replicated database is performed by a distributed broadcast algorithm. The database consists of a Node Table, which contains the names and characteristics of all Network Nodes, and a Link Table, which contains the identity and characteristics of all links between pairs of Network Nodes. A link is uniquely identified by the pair {Origin Node, Destination Node}. The link from node A to node B is considered to be distinct from the link from node B to node A.

The Node Table is a simple list. Each entry points to a list of links which have their origin in that node. Link Table entries specify the link origin and destination as indices into the Node Table.

Computing routes. In order to compute an optimal path from a source node to a destination node, weights must be assigned to nodes and links. When a transaction program requests a session, it specifies a Mode with which the session is associated. One attribute of the Mode is its Class of Service (COS). A Class of Service might specify that the path be secure (e.g., no public or satellite links) or that links on the path have sufficient bandwidth to support batch data traffic such as large file transfers.

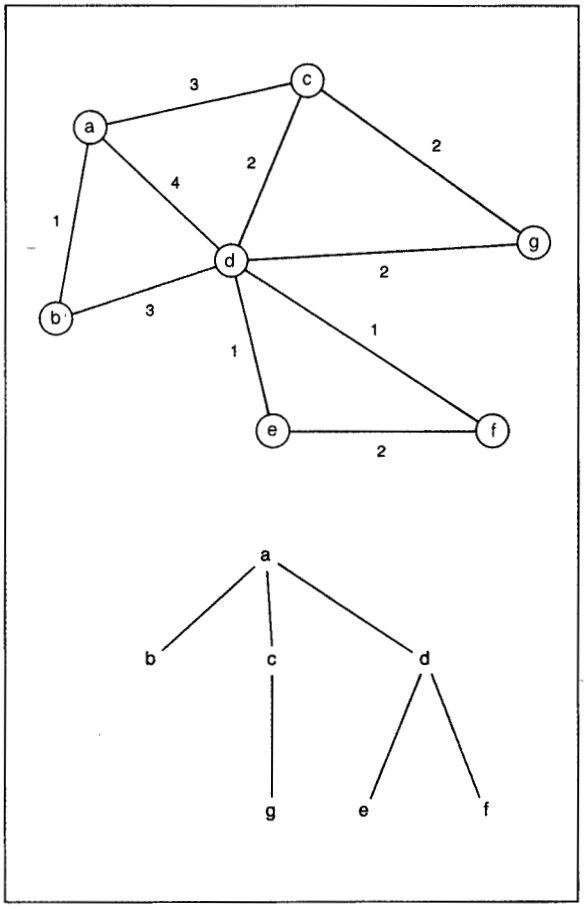
Route computation consists of the construction of a spanning tree, rooted at the node which is the path origin, indicating the best path to all Network Nodes for a specified Class of Service. Such a Rooted Tree is shown in Figure 13, and is constructed by using a variation of Dijkstra's basic single-source shortest-path algorithm.[27,28] The number shown next to each link in the network is the weight assigned to that link. It is computed by applying the COS mapping to the current characteristics of the link and the node which is the link destination. The path of least weight from any node **n** to the source node can be found by traversing the Rooted Tree from node **n** to the source.

Topology database performance issues. Many topology database entries are referenced during a single path computation. Disk access of entries would result in poor performance. Dedicating real storage (SQS) to the database is difficult to justify, since path computation is performed relatively infrequently and real storage is a scarce resource. The solution chosen is to place the database and all components which access it in a single task. It is a characteristic of the System/36 that all pages of a task must be in real storage before the task can enter execution state. The solution has the characteristic that the database resides in real storage when the functions that use it are executing; it is swapped to disk at other times. Since all components which access the database reside within the same task, it is unnecessary to provide a locking mechanism to serialize database access. The disadvantage of this solution is that there is a 64K-byte maximum task size which implies a limit on the size of the database. The current implementation allows 150 Network Nodes and 600 links between Network Nodes.

Directory Services

Directory Services provides the identity of the node in which a specified Logical Unit resides. This information is needed so that a path to the node can be computed and a BIND sent in order to establish a session to the Logical Unit. In the future one may wish to locate other types of resources such as transaction programs, files, and printers.

Figure 13 A rooted tree



Since Directory Services initiates searches only for Logical Units, it is assumed that a request for Directory Service will always be followed by a request for a route to the node in which the Logical Unit resides. Anticipating this request, Directory Services obtains a path to the Node from Route Selection Services.

The Distributed Directory. Each Network Node maintains a Local Directory Table containing those resources which reside in the node or in adjacent End Nodes. The Network Node is said to own these resources. The union of the individual Local Directory Tables may be viewed as a distributed Network Directory.

Finding Logical Units. A session may be requested by a Logical Unit in an End Node or in a Network Node. When an End Node requests a session to a Remote Logical Unit in the APPN network, it perceives the Logical Unit as residing in the adjacent Network Node. The End Node therefore sends a BIND to the adjacent Network Node. The C/SNA task in the Network Node recognizes that the Logical Unit for which the bind is actually destined does not reside in the Network Node. The BIND is sent to the Session Connection Manager subcomponent so that intermediate session activation may be performed. The Session Connection Manager invokes Directory Services in order to resolve the Destination Logical Unit to the name of the node in which it resides. If Directory Services locates the Logical Unit, it obtains a path to the node from Route Selection Services and returns both the location of the Logical Unit and the optimal path on which it can be reached. The Session Connection Manager appends the path to the BIND.

When the Logical Unit requesting the session is in a Network Node, it is the LU Network Services subcomponent of C/SNA, responsible for session endpoint activation, which invokes Directory Services in order to resolve the Destination Logical Unit to the name of the node in which it resides. Whether the session originates in an End Node or in a Network Node, the Network Node which invokes Directory Services is the *search origin.*

When Directory Services receives a request from C/SNA, it first checks the Local Directory Table to determine whether the Logical Unit is in the Network Node or an attached End Node. If the Logical Unit is found in the table, the name of the node in which it resides can be learned directly from the table.

If it is not found in the table, a cache of previously located Logical Units is searched. The cache is simply a fixed-size table of Logical Unit names and associated owning nodes into which newly located Logical Units are placed, displacing existing table entries on a Least Recently Referenced basis. If the Logical Unit is found in the cache, its location must be verified, because its residence may have changed since the last time it was referenced. Verification is performed by sending a message directly to the node which the cache indicates is the owner of the Logical Unit and obtaining a confirmation from that node. The verification message follows a path through the network supplied by Route Selection Services. If verification fails, the incorrect cache entry is removed.

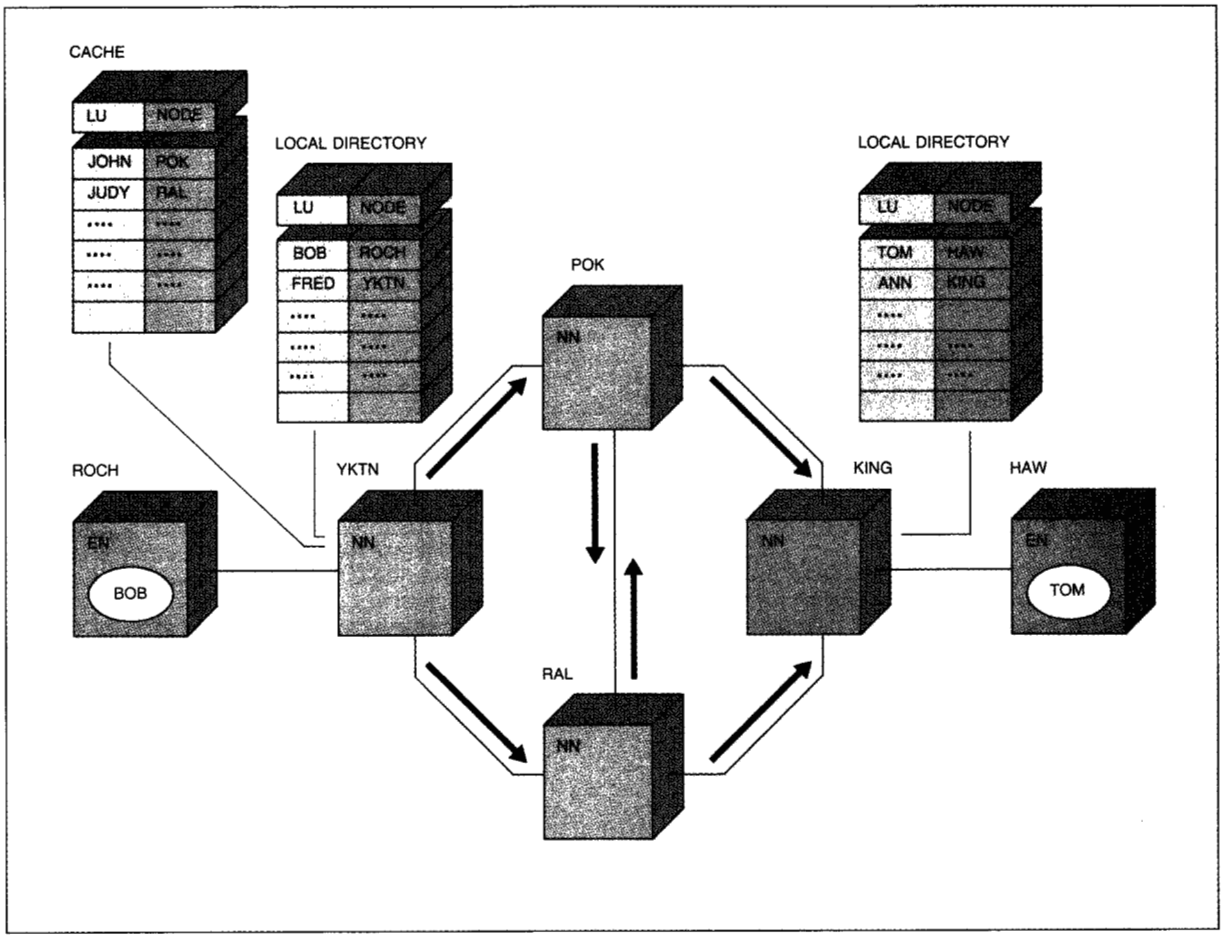
If the Logical Unit is not found in the cache or if verification fails, a *broadcast search*[11] is initiated. The object of the broadcast search is to examine the Local Directory at every Network Node in order to determine whether the Logical Unit is present and pass this information back to the search origin. A broadcast search uses considerable bandwidth, since search messages are forwarded to every Network Node in the network. The caching of Logical Units and optimizations described in later sections reduce the likelihood that broadcast searches will be required. The broadcast algorithm itself ensures that the search origin learns as quickly as possible when the Logical Unit has been found, although the search itself may not yet have reached all nodes. This reduces the response time seen by the user waiting for session establishment.

Each Network Node searches its local directory.

Figure 14 shows how a Logical Unit is found in the APPN network. Network Node YKTN has received a BIND from Logical Unit BOB in End Node ROCH. The destination Logical Unit is TOM. YKTN searches its local directory of Logical Units which reside in YKTN or in attached End Nodes. When this fails, the cache in YKTN is examined, but the Logical Unit is again not found. YKTN initiates a broadcast search among the Network Nodes of the network, and each Network Node searches its local directory. TOM is found in the directory of node KING. The information that TOM resides in node HAW is propagated back to YKTN. The information is placed in the cache at YKTN so that a broadcast search may not be necessary in the future.

Data structures and design considerations. Each search being processed by a node is represented by a data element containing a unique identifier associated with that search. Chained to the search element

Figure 14 APPN directory services



is an uptree element representing the node from which a search message has been received and a linked list of downtree elements representing the nodes to which search messages have been forwarded. As in the case of the Topology and Route Selection task, Directory Services implements its data structures from storage obtained within its own 64K task address space.

Directory Services optimizations. Examination of the System/36 environment suggested certain implementation choices which would be likely to improve performance.

Batching duplicate search requests. It is possible that a node may attempt to establish a large number of sessions to a Logical Unit within a short period of time. This is particularly likely if the Mode has been configured for large numbers of pre-established sessions. In this case the C/SNA task can flood Directory Services with requests to find a Logical Unit. If the resource is not in the Local Directory Table, a separate directory search may be initiated for each session to be established.[29] All of these searches are identical except for the search identifier. This situation wastes bandwidth and slows session activation response time. Directory Services recognizes new service requests which are identical to service requests that are being processed. It chains these duplicate requests to the original request and satisfies all such requests with the results of a single directory search.

Avoiding directory search. A System/36 APPN node has only one Local Logical Unit. This Logical Unit must have the same name as the Local Node. Before

performing any search activity outside the node, Directory Services invokes Route Selection Services to determine whether there is a Node listed in the Topology Database with the same name as the Logical Unit for which directory service has been requested. If the Node name is found in the database, there must be a Logical Unit bearing the same name contained in that Node. Since Logical Unit names are required to be unique in the network, it follows that the Logical Unit being sought can be resolved to the Node with the same name. Route Selection Services provides Directory Services with a route to the Node. If Route Selection does not find the name in its database, Directory Services returns to its normal processing to resolve the Logical Unit to the name of the node in which it resides. The number of additional cycles required to perform this check on the Logical Unit name is small compared to the cost of a directory search.

Concluding remarks

System/36 APPN provides highly dynamic, fully distributed peer networking in a product which is easy to use. Preliminary performance studies indicate that the time required for data to pass through an intermediate node is sufficiently small to support multihop interactive activities such as remote log-on capability.

This paper has described some of the ways in which APPN was tailored to the System/36 environment. The authors believe that the Advanced Peer-to-Peer Networking design may be implemented successfully in a wide variety of system environments.

Acknowledgments

The System/36 APPN implementation was a joint effort of the IBM Thomas J. Watson Research Center and the IBM System Products Division. The authors would like to express their high technical regard and warm personal regard for those people from SPD Rochester who participated in the System/36 APPN implementation. Those who made substantial technical contributions are: Dan Amundson, Dan Brossoit, Dave Christensen, Beth Gabriel, Carole Mataya, John McGinn, Mike Monday, Karen Orbeck, Chris Jones, Narendra Singh, Laurie Strand, and Saeid Sakhitab. Marv Kulas, Scott McCreadie, and Rosie Rocke spent many hours testing APPN networks in a wide variety of configurations to verify the correctness of the implementation. Don Morrison provided early technical direction. We would also like to thank the following members of the CPD Architecture Group at Research Triangle Park, North Carolina, for their advice and review of the design of System/36 APPN: Jeff Carley, Kathy Clarke, John Drake, and Melinda Pollard. Those most responsible for the concept of APPN are Alan Baratz, Jim Gray, Paul Green, Jeff Jaffe, Diane Pozefsky, and Lee Rafalow. Others who contributed to the realization of APPN are Paul Frosch, Bharath Kadaba, Jean Lorrain, Frank Moss, Larry Plank, Mark Pozefsky, and Chuck Wood.

Cited references and notes

1. System/36 Advanced Peer-to-Peer Networking is System Support Program (SSP) Feature No. 6096 and Program No. 5727-SS1/17. It is available through IBM branch offices or by calling 800-IBM-2468.
2. SNA is IBM's architecture for networking. It specifies exactly those functions and protocols which a product must implement in order to cooperate with other products in the SNA network.
3. J. P. Gray, P. J. Hansen, P. Homan, M. A. Lerner, and M. Pozefsky, "Advanced program-to-program communication in SNA," *IBM Systems Journal* **22,** No. 4, 298–318 (1983).
4. *Transaction Programmer's Reference Manual for LU Type 6.2,* GC30-3084, IBM Corporation (1985); available through IBM branch offices.
5. *Format and Protocol Reference Manual: Architecture Logic for Type 2.1 Nodes,* SC30-3422, IBM Corporation (1986); available through IBM branch offices.
6. The term *adjacent* means that there are no intervening SNA nodes. Examples of adjacent nodes are the primary and a secondary on a multidrop leased line, two nodes connected by a dial line, or two nodes connected across an X.25 packet network.
7. An SNA Logical Unit (LU) is a port through which an end user accesses the SNA network. An end user can be an application program or a workstation operator.
8. B. C. Housel and C. J. Scopinich, "SNA Distribution Services," *IBM Systems Journal* **22,** No. 4, 319–343 (1983).
9. A customer may wish to attach a System/36 as an End Node node to the APPN network when communication into the network is required without the burden of providing network services. The customer activates the APPC subsystem to obtain End Node function, or the APPN subsystem in order to obtain Network Node function.
10. The user transaction program must recognize a return code indicating that a session failure has occurred. The program can then reissue the ACQUIRE verb. A new session will be obtained on an alternate path if such a path exists. Participation of the transaction program is required for recovery from session failure due to link outage.
11. A. Segall, "Distributed network protocols," *IEEE Transactions on Information Theory* **IT-29,** No. 1, 23–35 (1983).
12. J. M. Jaffe, A. E. Baratz, and A. Segall, "Subtle Design Issues in the Implementation of Distributed, Dynamic Routing Algorithms," *Computer Networks and ISDN Systems* **12,** No. 3, 147–158 (1986).
13. A. E. Baratz, J. P. Gray, P. E. Green, J. M. Jaffe, and D. P. Pozefsky, "SNA networks of small systems," *IEEE Journal on Selected Areas in Communications* **SAC-3,** No. 3, 416–426 (1985).

14. P. E. Green, R. J. Chappuis, J. D. Fisher, P. S. Frosch, and C. E. Wood, "A perspective on Advanced Peer-to-Peer Networking," *IBM Systems Journal* **26,** No. 4, 414–428 (1987, this issue). This paper provides additional information on System/36 communication functions.
15. Although configuration of Remote Logical Units is not required, the user is given the opportunity to supply an optional list of Logical Units and associated Nodes. This allows the user to selectively activate a link to a specific adjacent Remote Node by enabling the associated Logical Unit. Most users do not require this level of granularity in link activation and should leave the menu blank.
16. The ICF verbs are not the same as the LU 6.2 Architecture verbs, but in the case of the APPC subsystem there is a mapping between the two sets of verbs. For example, the ACQUIRE and EVOKE verbs together map to the single LU 6.2 verb ALLOCATE.
17. There are many such SNA subsystems. These three have been chosen for illustrative purposes.
18. Implemented by POST and WAIT supervisor calls.
19. The Finance subsystem allows System/36 users to communicate with programs running in the 3601 and 4701 Finance Controllers and the 3694 Document Processor.
20. DLC is the protocol by which data packets are actually sent on a line. It performs flow control functions so that link buffers are not overrun and ensures the integrity of data sent over the link. System/36 supports SDLC and X.25 protocols.
21. Communication with CICS and IMS via LU 0.
22. C/SNA supports LU 6.2, Secondary LUs 0, 1, 2, and 3, and Node types 2.0 and 2.1. Two additional subsystems, SNA 3270 Emulation and Multiple Session Remote Job Entry, make use of the C/SNA function in a manner similar to APPC and SNUF.
23. The Control Point sessions which carry network control messages always use the single-hop path over the link directly connecting the two nodes. In this case the route is obtained from the association between the Remote Logical Unit and the Remote Node, since no alternative path is possible.
24. If dynamic activation of Remote Logical Units is to be performed, the user must specify the new parameter APPN-YES on the SESSION OCL statement.
25. For simplicity, the establishment of the session on which the CNOS conversation takes place is not described. It is established like any other LU 6.2 session, except that it uses a Session Group whose limits are not negotiated, so that the possibility of recursion is avoided.
26. Some additional APPN specific activities are required in the preserved session endpoint activation function. The session activation message must be extended to contain the session path and other information required along the route. Initial values are established for functions such as adaptive hop-by-hop flow control which are not performed by other subsystems.
27. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Co., Inc., Reading, MA (1975).
28. J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET," *IEEE Transactions on Communications* **COM-28,** 711–719 (May 1980).
29. There would be a broadcast search corresponding to each requested session only if no searches completed before the last of the multiple service requests received from C/SNA. If one of the broadcast searches did complete, the resource would be cached and subsequent searches would be directed searches.

Robert A. Sultan *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598.* After joining IBM in 1979, Mr. Sultan worked on the design and development of manufacturing systems. He joined the Computing Systems Department of the Research Center in 1981, working in the areas of systems monitoring and telecommunications planning. In 1983 Mr. Sultan joined the Network Architecture and Protocols Group, where he participated in System/36 APPN design and implementation. He is currently a member of the Data Communications Architecture Group, working on the design and analysis of network control algorithms. Mr. Sultan received a B.S. in humanities and science from MIT in 1968 and an M.S. in computer science from Pennsylvania State University in 1979.

Parviz Kermani *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598.* Dr. Kermani has been a Research Staff Member in the Computer Science Department at the Research Center since 1978. He received a B.S. in engineering from the University of Tehran, Iran, in 1969, an M.A. in mathematics from the University of Waterloo, Canada, in 1973, and a Ph.D. in computer science from UCLA in 1977. He is currently manager of the Data Network Analysis Group at the Research Center. In 1984 he was a member of the Technical Planning Staff of the director of the Research Division, and in 1986 spent a sabbatical year at the IBM Research Laboratory in Zurich, Switzerland. His work has been in the area of design, development, and performance analysis of communication networks. Dr. Kermani is a member of the IEEE.

George A. Grover *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598.* Mr. Grover was a member of the Network Architecture and Protocols Group at the Research Center which implemented System/36 APPN jointly with SPD Rochester. He was involved in the design of its deadlock-free flow control and of critical synchronization processes. From 1979 to mid-1982 he was a member of the Communication Programming System Design group in CPD Kingston. In both Research and CPD Mr. Grover has worked extensively in the design of SNA networking functions. Previously he participated in assembler, compiler, and operating system design and development activities in conjunction with System/360, the Stretch computer, and the 7950 (a special-purpose extension of the Stretch computer), and in technical planning activities relating to advanced technology, security, and privacy. Mr. Grover received a B.A. from Amherst College in 1951 and joined IBM in 1954.

Tsipora P. Barzilai *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598.* Ms. Barzilai is a member of the Data Communication Architecture group at the Research Center. She has been with IBM since 1982, and was involved in the development of a tool for performance evaluation of computer networks. In 1984 she joined the team that was responsible for design and implementation of APPN on the System/36. Recently she has been working on cost performance and design of computer communication protocols. Ms. Barzilai received a B.Sc. and an M.Sc. from the Technion, Israel Institute of Technology, in 1975 and 1979, respectively.

Alan E. Baratz *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598.*

Dr. Baratz is manager of the Telecommunications Network Architecture Project at the Research Center. He did his undergraduate work at UCLA, where he received a B.S. in math/computer science in 1976. His graduate work was done at MIT, where he received M.S. and Ph.D. degrees in computer science in 1979 and 1981. His thesis research with Professor R. L. Rivest was on algorithms for integrated circuit signal routing. In 1981 Dr. Baratz joined the IBM Thomas J. Watson Research Center as a Research Staff Member, and in 1984 assumed his present position. His research interests have included computer communications, distributed algorithms, VLSI layout algorithms, and combinatorial graph algorithms.

Reprint Order No. G321-5306.