Multicast network connection architecture

by N. Budhiraja
M. Gopal
M. Gupta
E. A. Hervatic
S. J. Nadas
P. A. Stirpe

IBM Networking BroadBand Services (NBBS) is a comprehensive networking architecture designed for high-speed networks. To meet the demands of multicast communication applications, e.g., video on demand, software distribution, and video conferencing, NBBS multicast services provide support for efficient best-effort and guaranteed quality-of-service (QOS) multipoint network connections. The NBBS multicast services consist of set management services and multicast network connection services. Set management services provide the ability to create and manage groups of users who are interested in participating in multicast activities. Multicast network connection services use the group membership information provided by set management services to support multiple network connections per group and to provide supporting services such as nondisruptive path switching, path preemption, and bandwidth management. The multicast network connection services support the multicast signaling requirements defined by the asynchronous transfer mode user-to-network interface (ATM UNI) specifications. This paper gives an overview of the NBBS multicast connection services. Set concepts are introduced, an overview of set management protocols is given, and the value of set management services in establishing multicast network connections is discussed. Finally, multicast network connection protocols that establish, maintain, and terminate multicast network connections are described.

The advent of high-speed networks has stimulated the development of many new distributed applications, such as multiparty video conferencing and distributed database applications. Such emerging multicast applications require advanced networking features, such as guaranteed quality of service (QOS) and connection rerouting in case of failures, to provide high quality of service. Several multicast protocols have been proposed in the networking community, but are in the early stages of development.[1–3]

Networking BroadBand Services (NBBS) is a networking architecture that provides for a range of multipoint communication services (multicast services) to coordinate set management and the communication channels utilized by set members. The multicast services consist of set management services, and multicast network connection services. The set management services allow groups of users (or parties) who share common interests to create, join, or leave *sets*. Once a set is created, multicast network connection services can be used to establish and maintain the underlying communication channel utilized by all users in the set. These communication channels can be point-to-multipoint or multipoint-to-multipoint connections. Multiple connections per set are possible, each with different characteristics. Point-to-multipoint connections provide one-to-many connectivity with QOS

0018-8670/95/$3.00 © 1995 IBM

guarantees, e.g., bandwidth reservation. Multipoint-to-multipoint connections provide many-to-many connectivity with best-effort delivery. Once a multicast network connection is established, set management services operate in concert with multicast network connection services to automatically extend or trim a multicast communication channel in response to a party joining or leaving the set. For example, a party may join an existing set for which a video conference is ongoing and be automatically incorporated in the connection.

The paper is organized as follows. NBBS set concepts are discussed in the next section, the NBBS set management protocols are described, followed by a discussion of point-to-multipoint and multipoint-to-multipoint network concepts and the NBBS multicast service models. Finally, we describe the NBBS multicast network connection services.

Set management concepts

The NBBS set management services allow users interested in communicating among themselves to form a *set*. A set consists of a list of users and has a *name*. It comes into existence once the first user joins, and ceases to exist once the last user leaves. Multiple sets can coexist in the network, each with its own unique name. Once created, the NBBS multicast services can be used to create one or more communication channels among the users in the set. For example, consider some users who are interested in participating in a video conference. These users can form a set and subsequently create communication channels among themselves by specifying the name of the set. If necessary, more users can join, or some of the existing users can leave. The communication channels will be automatically extended or trimmed as required by NBBS.

NBBS allows the creation of two kinds of sets, *open* and *closed*. Open sets are well known, and all information about them (membership information, information about the communication channels existing between the members, etc.) can be accessed by any user. The set management services do not provide security; if required, this has to be enforced by higher layer protocols. Furthermore, any user can join or leave an open set. For example, consider the set of users participating in the video conference mentioned above. If they create a set as an open set, then any other user can voluntarily join and begin participating in the conference. Some other user who wishes to only send a message to the set (and not join) can also do so by finding the required information about the communication channels existing within this open set.

On the other hand, closed sets are created in NBBS if the participants wish to restrict the membership. A closed set is created by an *owning* user, who then controls the membership. No information about the set is available to any user who does not belong to the set. Referring again to the video conference example, if the participants do not wish to allow anyone else to join, they create a closed set.

Multipoint communication services support sets of users that communicate among themselves.

The following subsections discuss the two kinds of sets, and the services offered by them, in more detail.

Open sets. Any user, at any time, can request the set management services to join that user to an open set. If the set already exists in the network, then the user is joined to the existing set. If not, then a new open set is created with the requesting user as the member. Similarly, any member can leave at any time. The members can also create multicast communication channels among themselves that can be point-to-multipoint or multipoint-to-multipoint. These channels are trees that span all the members, with bandwidth reservation possible on point-to-multipoint trees. A user, however, does not need to know the current membership to create a multicast tree. The user just needs to provide the set name, and the membership is automatically determined by the network.

There is a special multicast channel that can exist with any open set, named the default distribution tree (DDT). The DDT is a zero-bandwidth, multipoint-to-multipoint tree. A user can request a DDT to be created when joining a set. If the DDT already exists, then the tree is extended to the new mem-

ber. If not, a new tree is created to span the entire membership, and subsequently the tree is extended or trimmed as the membership changes. The information about the DDT of a set (if it exists) is available to any user (not necessarily a set member), and can be used to multicast information to all the members.

As an example of the use of DDTs, an important family of open sets that are extensively used within NBBS are those consisting of the directory agents (DAs) as users. When a DA in NBBS wishes to locate a resource, the agent needs to send out a query in the network. Clearly, it is desirable that the query reaches only those remote DAs who potentially represent the resource. One way to achieve this is to have the DAs join different open sets, depending on the type of resources that they represent. A DA wishing to locate a resource finds out the required information about the set that contains the potential remote DAs, and multicasts the query on the DDT of that set. For example, the DAs representing Internet Protocol (IP) resources with the same subnet address could join a well-known set. Given the subnet address of a resource, one can then multicast the query to only those DAs who can represent the resource.

Closed sets. A closed set is created in NBBS if the membership needs to be controlled. An owning user creates the set and dictates which other users can join. The owner can also remove users. A user can, however, also voluntarily leave. Similar to open sets, users can also create one or more multicast channels among themselves. However, information about these channels is not available to users who are not members.

A closed set only exists as long as the owner wishes, and can be destroyed any time by the owner. When that happens, all communication channels existing among the members are also destroyed. Thus, closed sets are usually used in situations where a user only wishes to create temporary communication channels with a list of users, e.g., for example, a private video conference.

Set management protocols

This section gives an overview of the set management protocols in NBBS. Each NBBS network node has a *set manager* (SM) that handles all set management operations at that node. For each set g in the network, one of the SMs also assumes the role of a *set leader* (SL). Different sets may have different set leaders. The SL of g manages all changes to the membership of g, and keeps up-to-date information about the membership of g. In case of open sets, one of the SMs also assumes the role of a *registrar*. The registrar is a central repository that maintains information about all open sets existing in the network. This includes information about the location of the SL for each set, and information about how to multicast information to the members of that set.

In NBBS, the roles of SLs and registrars are assigned in a dynamic manner. In particular, if an SL or a registrar fails, then some other SM assumes that role in a dynamic manner. In addition, this changing of roles can be achieved without any disruption to the multicast connections existing among the set members. Also, to conserve network resources, if a user at a node fails, then the SM at that node automatically removes that user from all the sets the user belongs to, and all multicast connections extending to that user are also automatically trimmed.

The rest of this section discusses the various protocols between the SMs, SLs, and the registrar, along with various examples. We first discuss open sets and then closed sets.

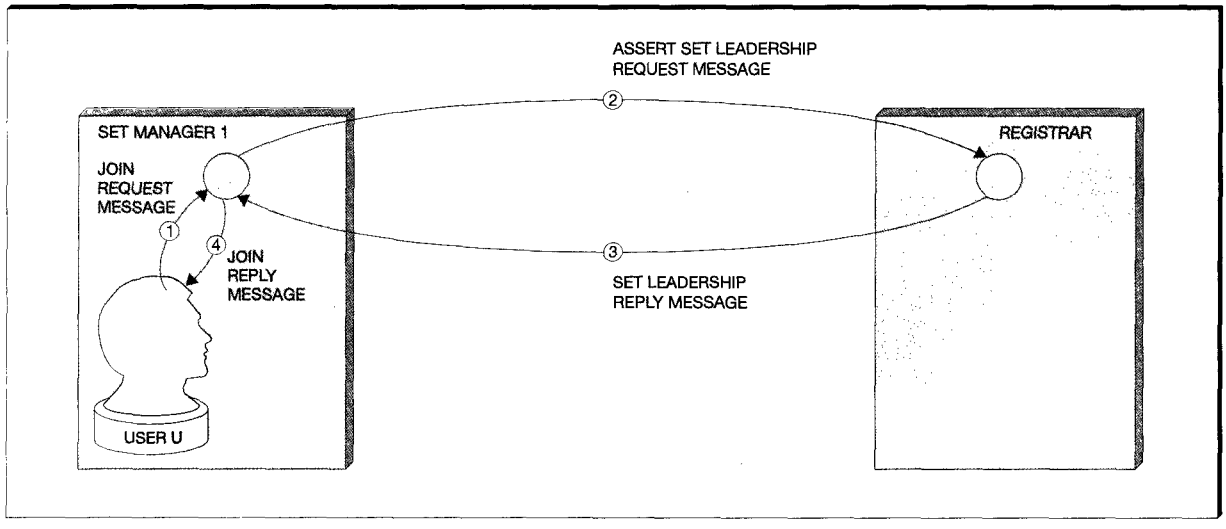
Protocols for open sets. The various operations that are provided to users in the case of open sets are join set, leave set, query set, and create a default distribution tree spanning a set. We describe each of these operations through examples.

Join set. The first example of a user wishing to join an open set is illustrated in Figure 1. In this case, a user u wishes to join an open set g. Consequently, the following messages are exchanged:

- User u sends a *join request* message (labeled 1) to the SM on its node indicating that it wishes to join the set g.
- On receiving the above message, the SM (i.e., SM1) checks to see if it knows the location of the SL for the set mentioned in the join request. In the example, we have assumed that SM1 does not know the location of the SL. Consequently, it needs to find the location of the registrar (as mentioned before, the registrar keeps necessary information about all open sets).

The location of the registrar is discovered through the topology services in NBBS. Any SM that be-

Figure 1 Joining a nonexistent open set



comes the registrar reliably broadcasts this information to the rest of the SMs in the network using the reliable topology exchange mechanism existing in NBBS. Thus, if a registrar exists in the network, its location is available to the SM at any node through the topology services.

SM1 thus locates the registrar and sends an *assert set leadership request* message (labeled 2) to the registrar indicating that it wishes to become the SL for g in case an SL for g does not already exist.

- On receiving the above message, the registrar checks to see if an SL exists for g. In this example, no SL is found. It, therefore, sends back a *set leadership reply* message (labeled 3) indicating to SM1 that it (SM1) is the SL for the new set. The registrar also updates its database to reflect that SM1 is the SL for g.
- On receiving the above reply, SM1 finds that no SL for g exists. Consequently, it becomes the SL, adds u to the set membership list, and then sends back a *join reply* message (labeled 4) to u indicating that it has been added to the set.

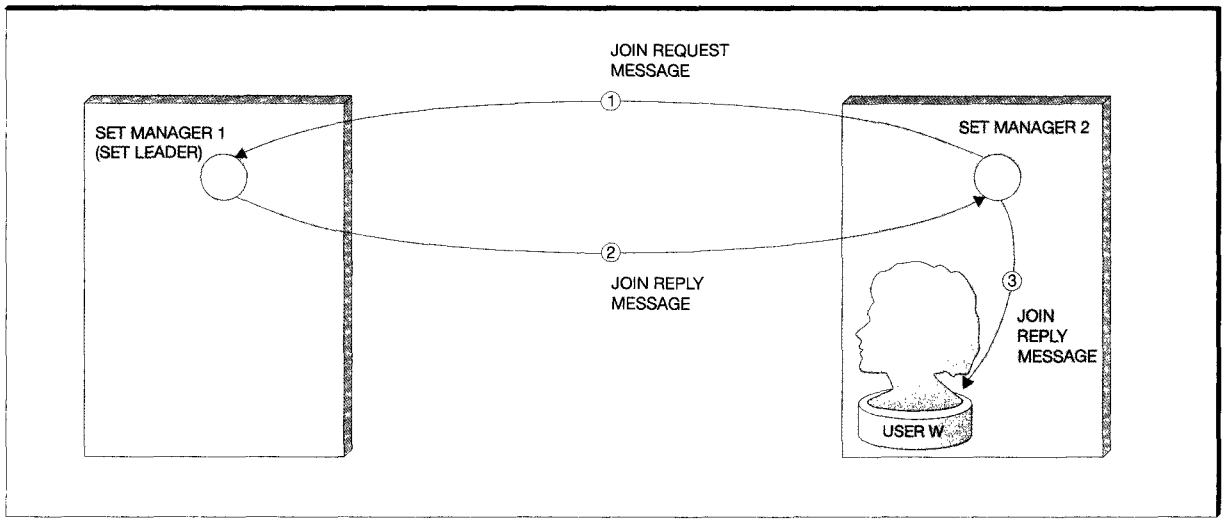
Once the above join is completed, the registrar and the SL (i.e., SM1) monitor each other for failures. This monitoring function is also provided by the topology services. For example, if the SL fails, then the registrar is informed of this by topology, and it removes set g from its database. Similarly, if the registrar fails, then the SL re-registers the information about g at the new registrar. A registrar election mechanism exists in NBBS that results in the election of a new registrar in case the old registrar fails.

In another example of open set join, we go a step further. Let us assume that another user w (at the node of set manager SM2) wants to join the same set g. The same sequence of messages is exchanged as in the previous example, except that the reply from the registrar indicates that SM1 is the SL, and the message also provides the location of SM1. We describe the sequence of events that occur from this point on (refer to Figure 2):

- After receiving the reply from the registrar, SM2 computes the path to SM1 and sends a join request message (labeled 1) to SM1 indicating that w wants to join the set.
- On receiving this message, SM1 adds w to its membership list and sends a join reply message (labeled 2) to SM2.
- On receiving the reply, SM2 indicates to w (labeled 3) that it has been added to the set.

SM1 and SM2 now start monitoring each other. In case SM2 fails, SM1 removes w from the set membership list. Similarly, in case SM1 fails, SM2 will try to become the SL of g by interacting again with the registrar. If the registrar indicates that no SL

Figure 2 Joining an existing open set



exists (as described before, the registrar was monitoring SM1 and would have been informed of the failure of SM1), then SM2 becomes the new SL. However, if another set manager (e.g., SM3) whose users were also set members of g had succeeded in reaching the registrar first, then SM3 would have become the SL. In this case, SM2 would not become the SL, and to ensure that SM3 has the up-to-date membership information, SM2 will send a *merge set membership request* to SM3 that would result in SM3 adding w to the membership list.

Leave set. We now give an example that illustrates how a set member can leave a set.

- Suppose that w wants to leave a set that it had joined earlier. It sends a *leave request* to its set manager SM2.
- On receiving the above message, SM2 sends a leave request to the SL (SM1) indicating that w wishes to leave the set.
- On receiving the above message, SM1 removes w from the membership list, and sends a *leave reply* to SM2.
- SM2 receives the above reply and indicates to w that it has been removed from the set.

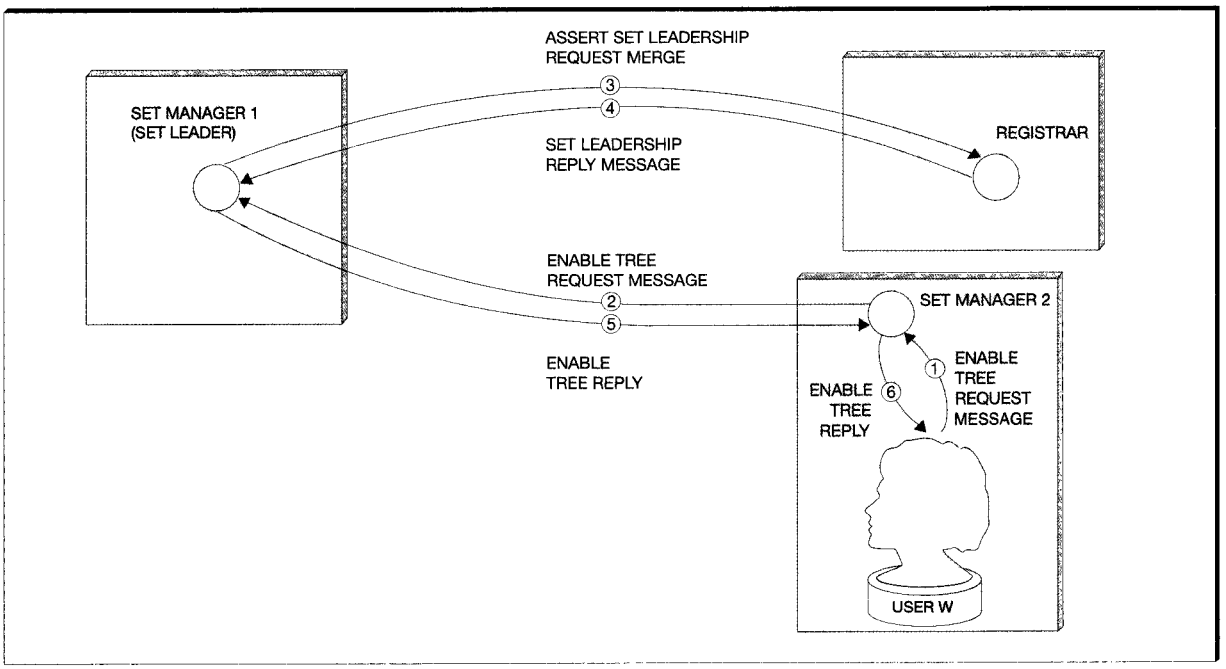
A similar sequence of operations will ensue in case the path to w fails. In this case, SM2 will detect that w has failed, and it will automatically send a leave request to SM1.

Query and tree creation. As can be seen, the user u that first joined the set does not automatically find out that another user w has also joined the set. Similarly, w also does not know that u belongs to the set. However, this information can be obtained by querying the set membership from the SL. In particular, w can send a *query set information request* to SM2, who can then get the required information from SM1.

Similarly, any set member can initiate the creation of the default distribution tree (DDT). As mentioned before, this tree is a multicast channel that spans the set membership. We outline next how this tree is created. Refer to Figure 3 for this discussion.

- Suppose that a set consisting of u and w has been created as described in the earlier examples, and w now wants to create a DDT. It sends an *enable tree request* message (labeled 1) to SM2.
- On receiving the above message, SM2 sends an enable tree request message (labeled 2) to SM1 indicating that a tree needs to be created for set g.
- On receiving the above message, SM1 initiates the tree creation. It first needs to get a tree address that will be used to multicast information on this tree. In NBBS, this information is obtained from the registrar. SM1 sends another assert set leadership request (labeled 3), with an indication that a tree address is required.

Figure 3 Creating a default distribution tree



- The registrar, on receiving this message, generates a tree address, and sends back a reply (labeled 4) to SM1.
- On receiving the above message, SM1 interacts with the multicast connection services to create the tree. SM1 sends a message to the connection services, providing with it the tree address that it had obtained from the registrar. Once the tree has been created, SM1 receives a reply. It then sends an *enable tree reply* message (labeled 5) to SM2 indicating that the tree has been created. The reply also includes the tree address that is to be used to multicast information on this tree.
- On receiving the reply, SM2 indicates to w (labeled 6) that the tree has been created, and w can now start multicasting information on this tree.

In case any other user later joins the set g, it will be automatically added to the DDT. Similarly, if a user leaves the set, it will be removed from the DDT.

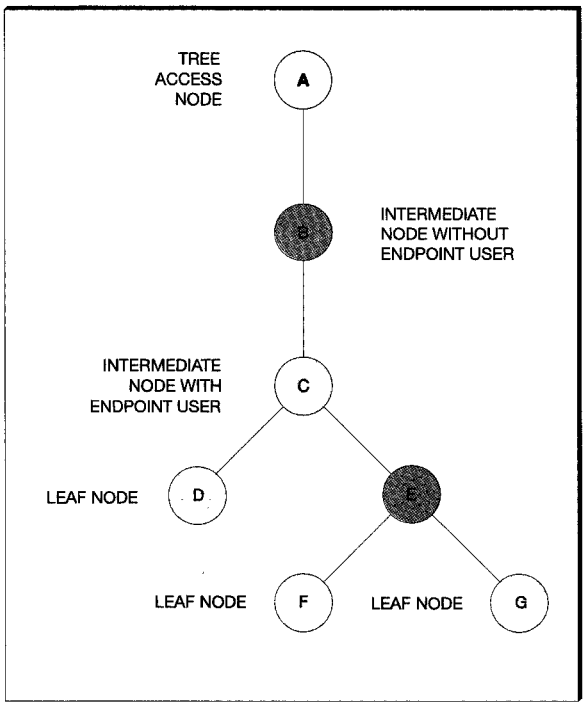
A DDT is a multicast channel on which no bandwidth is reserved. A set member can also create a point-to-multipoint tree with bandwidth reservation. The creation and management of this tree, however, is managed by the protocol agent (PA) representing the user creating the tree (a PA is an NBBS entity that is used to access all NBBS services). For example, if w wants to create a point-to-multipoint tree, its protocol agent will do all the necessary interactions with the multipoint connection services to create the tree. The only interaction with the SL would be to get the set membership information.

Algorithms for closed sets. We now briefly describe the algorithms for closed sets.

Closed set join. A closed set is created by an owning user as follows:

- Suppose user u wishes to create a closed set and wishes to include another user w in that set. User u first sends a directory query to locate w. On receiving a response to the above query, u sends a *create closed set request* to its set manager SM1.
- On receiving the above message, since this is a closed set, SM1 becomes the set leader (no interaction with the registrar is required in case of closed sets, since the same closed set cannot be created by two different users) and adds u to the

Figure 4 Point-to-multipoint and multipoint-to-multipoint network connection tree terminology



set. In order to check if w wants to join the set, SM1 sends a *join remote resource request* to the set manager of w, i.e., SM2.

- On receiving the above message, SM2 forwards it to w.
- If w wishes to join the set, it sends back a positive *join remote resource reply* to SM2.
- SM2 forwards the join remote resource reply to SM1, who then indicates to u that the set has been created, and w has also been added to the set.

Subsequently, if u wants to add another user (e.g., x) to the set created above, it locates x, and sends an *add user to set* request to SM1. SM1 then sends a join remote resource request to x, and in the case that x replies positively, adds x to the set.

Leaving a set. As with an open set, a user can voluntarily leave a closed set. However, in a closed set, the owning user (i.e., the user who initially created the set) can also remove a user from a set. The owning user does this by sending a *remove set member request* to the SL (who is in the same node as the owning user), who then informs the relevant user that it is being removed from the set.

Query and tree creation. Similar to open sets, a user belonging to a closed set can query the set membership of that set, and can also create point-to-multipoint and multipoint-to-multipoint communication channels within the set.

Multicast network connection concepts

Once a set has been created, the NBBS multicast services can be used to set up multipoint communication channels among the set members. This section describes how this is achieved.

Basic terminology. As previously stated, there are two types of multicast network connections, *point-to-multipoint* and *multipoint-to-multipoint*, both having a tree topology. This section defines some of the terms that are used to describe these network connections. Refer to Figure 4 for this discussion.

The *tree access* node of a connection is an entry point to a multicast tree, i.e., it is the node that can be used to multicast information on the tree. A *leaf* node is a node containing one or more target users. An *intermediate node* is a node (containing zero or more target users) that is part of the tree connection and has allocated network resources (e.g., labels or bandwidth) on one or more of its outbound links.

Characteristics of point-to-multipoint connections. A point-to-multipoint network connection allows a requesting party to send information to one or more destination parties. This type of network connection is sometimes called a "unidirectional tree" because each of its paths is unidirectional. With respect to sending data on the tree, a point-to-multipoint network connection has one tree access node (root of the tree). An important characteristic of a point-to-multipoint network connection is that bandwidth may be reserved on this type of connection. Furthermore, other QOS characteristics may be specified, such as those for the paths from the origin to the destination, e.g., maximum hop count or delay constraints. Point-to-multipoint network connections have one or more reserved network connection endpoints (NCEs), each representing one or more users participating in the point-to-multipoint network connection.[4] A point-to-multipoint network connection consists of one

or more paths that are established by the origin connection agent (CA). One path is created per leaf node.

If bandwidth is reserved on the point-to-multipoint network connections, data are sent with real-time, or nonreal-time delay priority. Furthermore, a "leaky bucket" regulates data flow onto the network connection. If bandwidth is not reserved, then data are sent with nonreserved delay priority.

Characteristics of multipoint-to-multipoint connections. A multipoint-to-multipoint network connection allows each party that participates in the connection to send information to all the other parties that are participating in the connection. This type of network connection is sometimes called a bidirectional tree because there is a path to and from each leaf node. Multipoint-to-multipoint network connections, however, do not provide bandwidth reservation, although other QOS characteristics, such as delay constraints from the origin to the destination are provided.[5]

For a multipoint-to-multipoint network connection, all intermediate nodes that represent one or more users and all leaf nodes are tree access nodes. The root of the multipoint-to-multipoint network connection is distinguished from the other nodes because it contains the origin connection agent (OCA) that creates and maintains the multipoint-to-multipoint network connection.

Multicast network connection service models

NBBS multicast services offer several types of network connection models to meet the needs of different types of multicast applications. However, many of the requirements, especially with respect to the ATM Forum[4] are in flux. Given these constraints, and with the understanding that these models are subject to change, this section describes the service models that NBBS currently provides.

In the ATM UNI (asynchronous transfer mode user-to-network interface) 3.0/3.1 specifications, the ATM Forum uses the term point-to-multipoint connection the same way that NBBS uses the term point-to-multipoint network connection.[6] The emerging ATM UNI 4.0 specification makes a distinction between a *call* and a *connection* and allows a call to have multiple connections. NBBS uses the term *set* to refer to the parties connected by a call with one or more connections. NBBS would say that one or more network connections are associated with a set. The ATM concept of multiple connections per call and the NBBS concept of multiple network connections per set is similar. However, NBBS sets have some properties that ATM calls do not, most notably, group IDs to identify sets, a registrar to register sets, and optional default distribution trees.

NBBS offers several types of network connection models to meet the needs of different applications.

Root-initiated creation. Some sender-based applications require that a root (or owner) form the network connection, explicitly issue *add parties* to cause parties to be added to the network connection, and explicitly issue *drop parties* to have parties removed from the connection. In this model, the root protocol agent (PA) obtains a list of parties, and the PA then interacts with the connection agent (CA) to form a tree from the root to all the parties (point-to-multipoint) or between the root and the parties (multipoint-to-multipoint). The root PA issues add party messages and drop party messages whenever desired.

The root chooses whether to permit or prohibit leaf-initiated add parties. In NBBS, when leaf-initiated add parties are allowed, the root always receives notification, either positive or negative, when a leaf-initiated add party is attempted.

Leaf-initiated joins. For receiver-based applications, a leaf can become aware of a set in one of several ways. Either the identification of the set is well known, or the protocol agent of the leaf has learned of the set (and perhaps of connections associated with the set) by interacting with other PAs in the network. Regardless of how the leaf becomes aware of the set, once a leaf is aware of a set, the leaf can initiate a join to the set (a call). The network connections in the leaf node that are asso-

Table 1 Steps in multicast connection establishment

Step	Action
1	The client requests the OPA to create multipoint connection.
2	The OPA requests the OCA to create connection to party members. The OPA may optionally request the SM to resolve the group identifier.
3	The OCA requests a set of paths from path services.
4	The OCA sends connection establishment messages on all paths.
5	The OCA collects replies and notifies the OPA of members participating in the connection.

Note: OPA = origin protocol agent
OCA = origin connection agent
SM = set manager

ciated with the set can then be extended to the leaf. If there are network connections in the node of the leaf that permit leaf-initiated joins, the node attempts to extend these network connections, in priority order, to the leaf. The origin CAs at the root of each of these network connections are notified of add parties. If some network connections in the node of the leaf do not permit leaf-initiated joins, then these are not extended to the leaf.

Leaf-initiated drop parties. Nothing prevents a leaf from leaving a network connection or a set (call) at any time, since there is nothing to prevent it from issuing a drop party across its UNI (user-to-network interface). Accordingly, all the service models include leaf-initiated drop parties from either network connections or sets. When a leaf leaves a set, it leaves all the network connections associated with that set, and if it was the owner of a network connection associated with the set, that network connection is disconnected.

Setless trees. For applications where there is always a one-to-one correspondence between the set membership and network connection participation, in particular, for ATM 3.0/3.1 point-to-multipoint connections, NBBS offers a setless tree, so that the overhead of sets can be avoided for applications where they are never used.

Overview of network connection algorithms

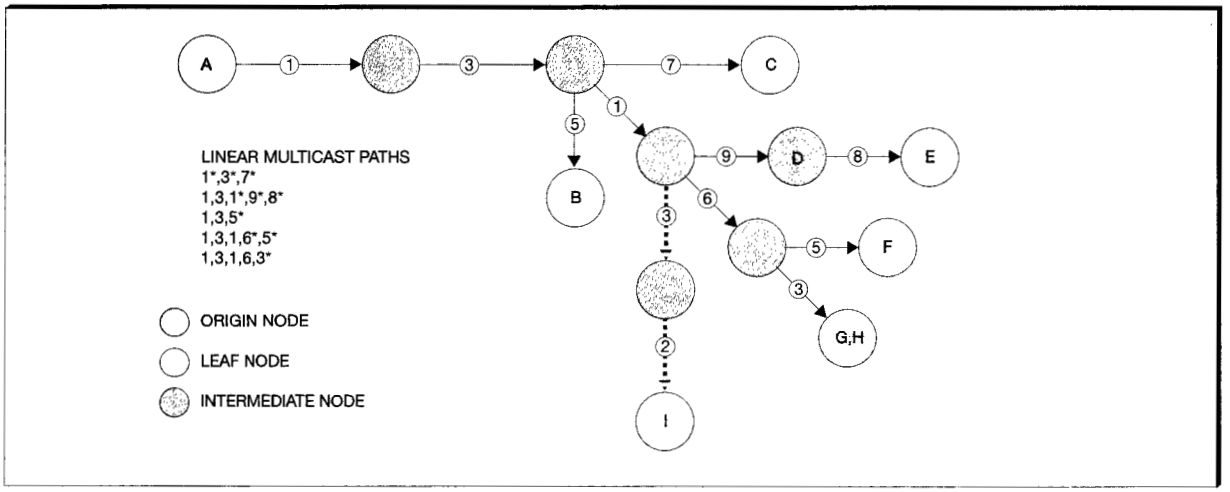
Multicast network connection services consist of several cooperating components that establish, maintain, extend, trim, and take down multicast network connections. The origin connection agent (OCA) is responsible for centrally controlling these modifications. Furthermore, the OCA coordinates refresh processing and nondisruptive path switching.[7] The destination connection agents (DCAs) represent the destination parties that participate in multicast connections. The tree manager (TM) interacts with the SM within a given node to resolve group names to local members or NCEs. Finally, the transit connection manager (TCM) is responsible for managing the link bandwidth and maintaining the connection state. These components operate in a coordinated manner to provide multicast network connection services.

Connection establishment. The sequence of steps to establish a multicast connection is described next:

1. A client party requests its PA (protocol agent) to establish a multicast connection to a provided group identifier or a set of party members.
2. The PA, now referred to as the origin PA (or OPA) of the connection, requests the OCA (origin connection agent) to establish a connection to one or more destination parties, providing the destination party names. Thus, the OPA may need to resolve the group identifier to a set of party members using the SM (set manager).
3. The OCA requests path services to compute a set of paths to the network addresses associated with the party or parties, such that the paths form a tree where the root of the tree corresponds to the requesting party and the intermediate and leaf nodes correspond to the destination parties. The set of paths used to establish the tree are collectively called the *tree object*. It is possible that more than one path traverse a common link and, therefore, share an intermediate TCM. However, it is guaranteed that each such TCM will process the connection establishment message exactly once.
4. The OCA sends a connection establishment message on each path in parallel. The DCAs (destination connection agents) associated with each destination party are requested to participate in the multicast connection.
5. The OCA collects the replies from the DCA(s), and notifies the OPA all of the destination parties that are participating in the multicast connection.

Table 1 summarizes these steps.

Figure 5 An example of a tree object



As stated in step 4 of Table 1 the OCA sends a connection establishment message on each path of the tree object. Figure 5 shows an example of a set of paths (a sequence of links) that comprise the tree object for a multicast connection that originates at node A. The asterisk after each link number in the path indicates that the connection establishment request message is copied to the appropriate TCM (transit connection manager) for that link. Likewise, the absence of the asterisk indicates that the message is not copied. Parties A, B, C, E, F, G, and H are located on leaf nodes, and party D is located on an intermediate node. For now, ignore the dashed arrows and party I.

The intermediate TCMs that receive a copy of the connection establishment request message reserve the requested bandwidth (if any), forward the request to the TM (tree manager) and await the reply from the TM. Once the TM replies back to the TCM, the TCM forwards the reply back to the OCA.

The TM at each node processes the connection establishment request and coordinates the interaction with the local parties as follows:

1. It interacts with the SM to resolve the group identifier in the message to a set of local destination parties.
2. For each party, it forwards the connection establishment request message to the party's CA. The DCA interacts with the party to indicate that a connection establishment request has arrived for that party.
3. It aggregates the connection establishment replies from each DCA, and returns the aggregated reply to the TCM for intermediate node TMs, or directly to the OCA for leaf node TMs.

Once established, multicast connections may be extended to add parties to the connection or may be pruned to remove parties from the connection.

Connection extension. Assume that the client party A signals its OPA (origin protocol agent) to extend the current multicast connection to include party I. The OPA requests the OCA (origin connection agent) to extend the connection. The OCA executes steps 1–4 listed in Table 1 resulting in the tree object shown in Figure 5, including the dashed lines and party I. The set of paths in the tree object is augmented to include the path to party I, mainly 1, 3, 1, 3*, 2*. In order to extend the multicast connection, the OCA sends a connection establishment request message on the new path. The processing by the copied TCMs and the TMs is the same as described above.

Connection pruning. Assume that the client party A requests the OPA to prune party I from the multicast connection. The OPA requests the OCA to prune the connection. This is accomplished by the OCA sending a connection take-down message on the path to party I, specifically 1, 3, 1, 3*, 2*. Again

the TCMs and TM process the message, and reply appropriately. The node upstream to party I's node is also removed from the connection since it is no longer required.

Connection refresh. Networks that provide bandwidth reservation must provide a mechanism for intermediate nodes to recover the reserved bandwidth in the event of a service outage. The reserved bandwidth is periodically refreshed to indicate that the connection is active and still requires the bandwidth. In an NBBS network, a refresh message is used as a mechanism to continue the reservation of the bandwidth that was negotiated on connection establishment. The OCA periodically sends a refresh message at a prenegotiated interval, on the connection, which is copied to the intermediate node TCMs and terminated at all the DCAs. The refresh request message contains a connection identifier so that the receiving components may distinguish which connection is being refreshed.

Nondisruptive path switch. A nondisruptive path switch (NDPS) may be performed whenever a link in a network connection fails. The goals of NDPS are to quickly reestablish the network connection, and to quickly reclaim network resources that are no longer required to maintain the connection.

In the NBBS multicast model, NDPS is centrally controlled by the OCA. When a link in the connection fails, the tree is divided into a main tree, the portion of the tree that includes the OCA, and one or more disconnected subtrees.

If explicit signaling is required to reattach each party in a disconnected subtree, the degree of signaling between the OCA and the DCAs in the disconnected subtree would be proportional to the number of party members that reside in the disconnected subtree. For example, if each party in the disconnected subtree is explicitly reattached to the main tree, the OCA must send O(N) message flows to the N leaf node DCAs in the disconnected subtree in order to reform a multicast connection between all parties. However, if the disconnected subtree is preserved, and the disconnected subtree is reattached by extending the main tree, only O(1) message flows are required. This is accomplished by signaling between a chosen node in the main tree, at which extension to the disconnected subtree is to occur, and the root node of the disconnected subtree. This is the optimal solution in terms of line flows. If the optimal solution is not possible, a heuristic algorithm is executed that attempts to reattach as large a portion of the disconnected subtree as possible.[8]

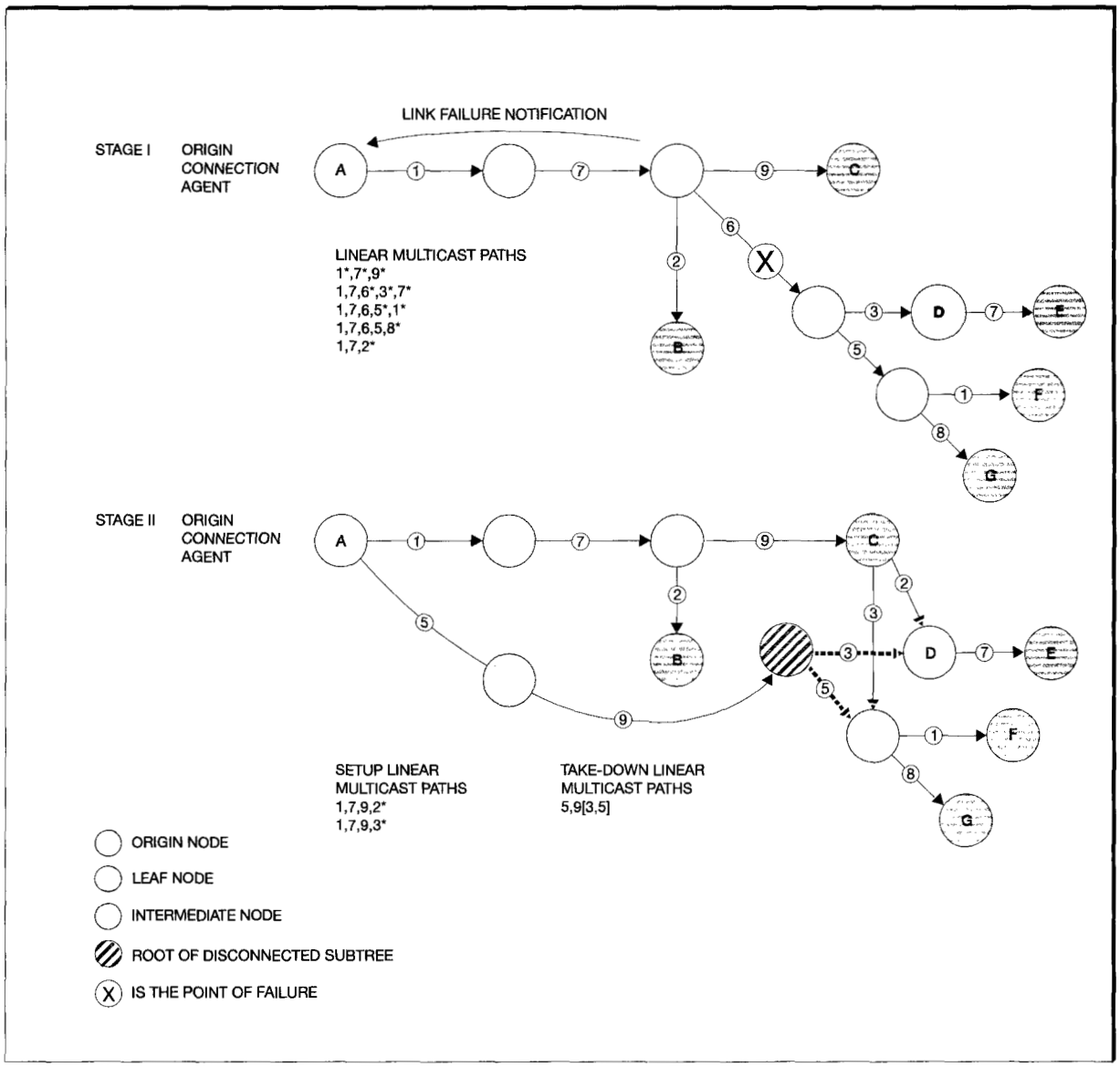
When NDPS is triggered,[7] the OCA performs the following actions:

1. The OCA provides path services with the link failures, the current tree object, and QOS and bandwidth requirements of the multicast connection, and requests path services to compute a new tree object that re-routes the connection "around" the link failure. The OCA receives the new tree object from the path services.
2. The OCA compares the new tree object with the current tree object to calculate two sets of paths, P1 and P2. The first set of paths (P1) consists of the connection establishment portion that will be used to reconnect the main tree with the disconnected subtree. The second set of paths (P2) is used to explicitly reclaim resources on the nodes that no longer participate in the connection.
3. The OCA sends a connection establishment request message on all paths in P1. In parallel, the OCA sends explicit path take-down messages on the paths in P2.

Figure 6 shows an example of the two stages of NDPS. In Stage I, the OCA receives an indication that a link that the connection utilizes has failed. The OCA invokes path services to compute a new tree object and re-route the failed connection. In Stage II, the OCA sends explicit connection establishment requests on the new paths. Network resources are explicitly recovered by sending a take-down message on the path corresponding to resources that are to be reclaimed. The take-down message contains a list of link identifiers whose network resources must be reclaimed. The link identifiers are illustrated in Figure 6 by square brackets in the take-down path.

In step 2 above, the OCA sends connection establishment request messages on all the paths in P1. Unless precautions are taken, a cycle could form in the connection under certain conditions. For example, Figure 7 illustrates a case in which, during NDPS, a cycle in a connection forms. Assume that parties A, B, and C participate in a multicast connection where A is the origin. Assume that the link fails between the nodes where parties A and B reside. The OCA that represents party A commences NDPS processing. Assume that path services com-

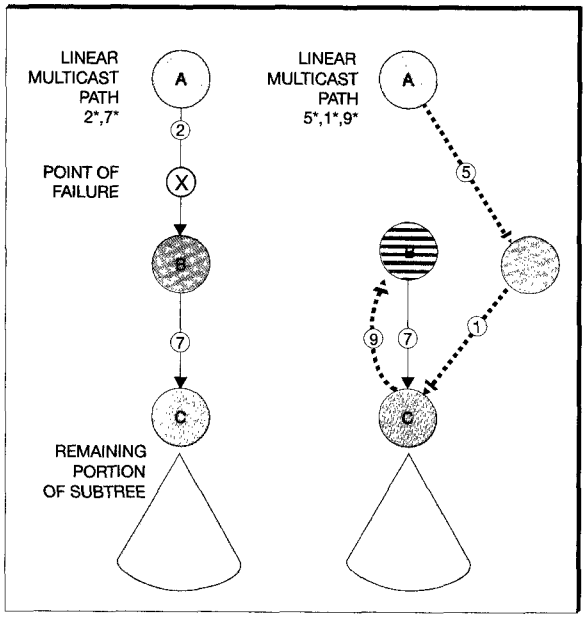
Figure 6 The two stages of nondisruptive path switch (NDPS)



pute a new path to reconnect the disconnected subtree using path 5*, 1*, 9*. The OCA for party A sends a connection establishment message on the path. In Figure 7, we observe that a cycle in the connection has formed between the PAs (protocol agents) representing parties B and C, because the link connecting party B to C has not yet been explicitly taken down.

Cycles during NDPS are prevented by including in the connection establishment message a list of link identifiers that must be "cleaned-up" or reclaimed prior to accepting the new connection establishment request. Since the OCA has the current tree object and the new tree object, it determines which nodes along the new path have links that must be reclaimed during the new connection establishment phase.

Figure 7 Illustration of cycle forming during nondisruptive path switch



In Figure 7, the connection establishment request message sent by party A's OCA would identify link number 7 at party B's node to be reclaimed. Thus, prior to B's DCA accepting the new connection establishment request, the DCA would reclaim the resources on link 7, preventing the formation of a cycle.

If explicit take-down of the network resources is not feasible because the OCA that controls the connection is partitioned from the disconnected subtree, then the OCA must maintain information about network resources it failed to explicitly recover for the duration of the refresh time-out interval. If a party requests to extend the current connection, the OCA, after obtaining a set of paths from path services, compares the paths with the saved state information. If any cycles could be formed if the connection extension were carried out, the OCA denies the extension.

Summary

This paper described the support, in NBBS, of multiuser network applications requiring best-effort or guaranteed QOS. Both the administrative and control aspects of the service (implemented by set management) and the delivery mechanism (implemented by the multicast connection management) were detailed. These mechanisms, coupled with specialized hardware capable of routing multicast messages through the switches with minimal overhead, make it possible to efficiently provide support for existing (e.g., in the local area network environment) and future multiuser applications.

Acknowledgments

The authors would like to acknowledge John Drake, Josh Auerbach, Chee-Seng Chow, Marc Kaplan, Shay Kutten, and Michael Ward for their initial work on the multicast connection architecture. Jong Yoon contributed to the set management protocols.

Cited references

1. D. Katz and P. S. Ford, "TUBA: Replacing IP with CLNP," *IEEE Networks* **7**, No. 3, 38–47 (May 1993).
2. P. Francis, "A Near Term Architecture for Deploying pIP," *IEEE Networks* **7**, No. 3, 30–37 (May 1993).
3. R. Ullman, "Internet Version 7," *Proceedings of the 26th Internet Engineering Task Force* (IETF), Columbus, OH (April 1993), pp. 598–600.
4. The ATM Forum, *ATM User-Network Interface Specification: Version 3.0*, Prentice Hall, Englewood Cliffs, NJ (1993).
5. H. Gün and R. O. Onvural, *On Multicast Tree Formation in Multimedia Networks*, IBM Technical Report TR-29.1432, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 (July 1992).
6. G. A. Marin, C. P. Immanuel, P. F. Chimento, and I. S. Gopal, "Overview of the NBBS Architecture," *IBM Systems Journal* **34**, No. 4, 564–589 (1995, this issue).
7. H. Ahmadi, P. F. Chimento, R. A. Guérin, L. Gün, B. Lin, R. O. Onvural, and T. E. Tedijanto, "NBBS Traffic Management Overview," *IBM Systems Journal* **34**, No. 4, 604–628 (1995, this issue).
8. T. E. Tedijanto, R. O. Onvural, D. C. Verma, L. Gün, and R. A. Guérin, "NBBS Path Selection Framework," *IBM Systems Journal* **34**, No. 4, 629–639 (1995, this issue).

Accepted for publication August 8, 1995.

Navin Budhiraja *IBM Research Division, Thomas J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, New York 10532 (electronic mail: navin@watson.ibm.com).* Dr. Budhiraja is a research staff member in the Advanced Networking Laboratory. Dr. Budhiraja received his B.S. in computer science from the Indian Institute of Technology at Kanpur, India in 1988, and his M.S. and Ph.D. in computer science from Cornell University in 1991 and 1993, respectively. Current research interests include networking, distributed systems, and fault tolerance.

Madan Gopal *IBM Research Division, Thomas J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, New York 10532.* Dr. Gopal joined the IBM Thomas J. Watson Research Center as a research staff member in 1985 and was a member of the Advanced Networking Laboratory until his departure in 1995. He received his master's and doctoral degrees in computer science from the University of Waterloo, Canada, in 1980 and 1985, respectively. Dr. Gopal's interests are in the area of platform-independent ATM switch software. He has published more than 20 papers in various journals and conference proceedings in the area of design, analysis, and performance modeling of network protocols. He is an active member of the Institute of Electrical and Electronics Engineers (IEEE).

Manish Gupta *IBM Research Division, Thomas J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, New York 10532 (electronic mail: guptama@watson.ibm.com).* Mr. Gupta has been working in the Advanced Networking Laboratory since November 1992. He received his B.Tech. in computer science and engineering from the Institute of Technology, Banaras Hindu University, Varanasi, India, and his M.S. in computer science from the University of Kentucky in 1992. His research interests include high-speed network architecture and distributed systems.

Elizabeth A. Hervatic *IBM Networking Hardware Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: hervatic@ralvm6. vnet.ibm.com).* Ms. Hervatic is an advisory programmer in ATM Market Development. She joined IBM in 1989 and has worked in Networking Architecture. Among other positions, Ms. Hervatic has been the technical leader of the NBBS base architecture team. Ms. Hervatic received her B.S. in computer science from North Carolina State University in 1989.

Stephen J. Nadas *IBM Networking Hardware Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: steve_nadas @vnet. ibm.com).* Mr. Nadas is an advisory engineer in Networking Architecture. He joined IBM in 1978 at the East Fishkill facility and has worked in the Myers Corners and Mid-Hudson Valley development laboratories. Mr. Nadas has been a performance analyst and benchmark developer for large systems and a microcode designer in Future Processor Development in Poughkeepsie. Mr. Nadas received his B.S. in mathematics from the State University of New York at Albany in 1976, and his M.S. in computer engineering from Syracuse University in 1994.

Paul A. Stirpe *IBM Research Division, Thomas J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, New York 10532 (electronic mail: stirpe@watson.ibm.com).* Dr. Stirpe has been an IBM employee for the past ten years, with work assignments in Endicott, New York, Santo Palomba, Italy, and most recently as a research staff member in the Advanced Networking Laboratory. He received his B.S. in electrical engineering from the State University of New York at Buffalo in 1985, his M.S. in computer engineering from Syracuse University in 1989, and his Ph.D. in computer science from Boston University in 1992. His research interests include performance analysis of communication networks, high-speed network architecture, and the enablement of multimedia applications on wide area networks.

Reprint Order No. G321-5583.