NBBS path selection framework

by T. E. Tedijanto
R. O. Onvural
D. C. Verma
L. Gün
R. A. Guérin

This paper describes the path selection function in Networking BroadBand Services (NBBS), which is IBM's architecture for high-speed, multimedia networks. The distinguishing feature of a multimedia network is its ability to integrate different applications with different traffic characteristics and service requirements in the network, such as voice, video, and data. In order to meet their service requirements, it is necessary for the network to provide unique quality-of-service (QOS) guarantees to each application. QOS guarantees, specified as multiple end-to-end performance objectives, translate into path and link constraints in the shortest path routing problem. For a general cost function, shortest path routing subject to path constraints is known to be a nonpolynomial- (NP-) complete problem. The NBBS path selection algorithm, a heuristic solution based on the Bellman-Ford algorithm, has a polynomial order of complexity. The algorithm finds a minimum hop path satisfying an end-to-end delay (or delay variation) constraint, that in most cases also optimizes a load balancing function. To reduce the number of path constraints, other QOS requirements such as packet loss ratio are implemented as a link constraint. The notion of primary and secondary links is used to minimize the long-term overall call blocking probability by dynamically limiting the hop count of a given path. The path selection algorithm developed for point-to-point connections is described first, followed by its extension to the case of point-to-multipoint connections.

Path selection function is at the core of every routing protocol. In a packet-switching network, path selection determines the path each packet takes from the originating end station to its destination. Paths are selected to satisfy a constraint and to optimize some criteria. A routing protocol is designed around a path selection algorithm in which the type of information exchanged, how often it is exchanged, and how it is used depend on the path selection objectives. A survey of routing protocols used in communication networks is given in Reference 1; unconstrained path selection algorithms used in these routing protocols are discussed in Reference 2.

All shortest path algorithms reported in the literature are variants of Dijkstra,[3] Bellman-Ford,[4] and Floyd-Warshall.[5] These algorithms have a polynomial time complexity and find a path in the network that optimizes a single criterion, such as cost or delay, or a scalar function of a number of criteria.

Shortest path algorithms either can be implemented in a centralized location or may be distributed in the network. In the centralized version, all connections in the network are known at a single location. This might result in better overall optimization of network resources. Its main disadvantages are the single point of failure and scaleability problems. Distributed versions solve these

 0018-8670/95/$3.00 © 1995 IBM

reliability problems but because path selection decisions are made in different locations and because distributing information among them takes time (at least on the order of propagation delays), the overall network throughput may no longer be as good as it is in the centralized decision making.

Path selection algorithms are required to find paths as the connection requests arrive at the network.

The main objective of the path selection algorithms in traditional packet-switching networks is to optimize the overall network utilization. A solution to this problem can be achieved in theory if all the connection requests are known ahead of time. However, it is not possible in practice to predict when and what type of connections arrive in the network. Assuming for a moment that they are known, the optimization problem may be formulated as a multicommodity flow model.[6] The significance of this model is that it can be used to show how to minimize the average delay. For example, for a given static traffic demand matrix, a nonzero proportion of traffic for each origin-destination pair follows a path if and only if the path minimizes the sum of the first derivatives of the delay function.[7]

Based on this result, shortest path algorithms are developed for choosing paths that minimize the increase in the total cost, in which the cost function for each link is the packet delay (which in turn is a function of the current utilization levels). Computing paths serially as connections are set up is an approximation to computing all the paths simultaneously as called for by the original result of the multicommodity flow analysis.

Service requirements of multimedia applications may include a set of quality-of-service (QOS) metrics including maximum end-to-end delay, maximum delay variation (jitter), and packet loss ratio. In the Networking BroadBand Services (NBBS) architecture, these end-to-end QOS requirements are supported on a link-by-link basis, with each link in the network providing link-level QOS guarantees. Link metrics for each link are advertised in the network and are updated as traffic dynamics in the network change. Using this information, the path selection function at the origin nodes (in which connections originate) computes a path that satisfies the given end-to-end QOS requirements. To guarantee the link-level QOS, each link performs connection admission control (CAC) by reserving bandwidth to each network connection multiplexed onto the link and rejecting requests for new connections if requested bandwidth is not available. Given this framework, the total network throughput increases as the number of links used by each connection decreases (to the extent it is feasible).

Reserving bandwidth along each path guarantees meeting the end-to-end loss ratio requirements of applications. The maximum end-to-end delay requirement is incorporated into the path selection function. Hence, the NBBS path selection function minimizes the total number of links used for end-to-end connections using links that are likely to have the requested bandwidth available subject to maximum end-to-end delay along the path being less than or equal to the application requirement.

Furthermore, future connection requests are also taken into consideration toward achieving high (long-term) network throughput. Toward this objective, the NBBS path selection algorithm uses a load balancing function to choose one among the paths with similar characteristics (i.e., minimum hop, maximum end-to-end delay, and bandwidth availability).

Jaffe[8] and Lee et al.[9] proposed approximate solutions to the shortest path problems with multiple constraints. The basic idea behind their proposals is to transform the constrained optimization problem to an unconstrained problem or to a tractable constrained problem that has similar properties. In the case of NBBS, the constrained optimization problem of minimizing the number of hops along the end-to-end path is subject to an additive constraint (i.e., delay or delay variation). Although the original problem is NP complete, due to the special nature of the objective function, this algorithm finds the optimal path with a polynomial time complexity.

In addition to applications that require end-to-end QOS guarantees, NBBS also integrates best-effort applications, i.e., those that do not require explicit

QOS guarantees. Path selection for these applications is similar to that used in traditional data networks (e.g., References 1 and 7).

One important characteristic of multimedia applications is that they tend to take place within a group of more than two users. To support group communications, NBBS supports multicast through point-to-multipoint as well as multipoint-to-multipoint connections.[10] Path selection for point-to-multipoint and multipoint-to-multipoint connections (hereafter called unidirectional or bidirectional trees) has been referred to in the literature as Steiner tree problems[11] and is known to be NP-complete. Heuristic solutions to Steiner tree problems have been proposed in the literature.[12–14] These heuristics were developed on undirected graphs (unlike our problem, which requires directional connections) as unconstrained optimization problems. These heuristics are extended in References 15 and 16 to take into account the end-to-end delay constraints of applications. The NBBS point-to-multipoint path selection algorithm is a direct extension of the NBBS point-to-point path selection algorithm that extends the basic concepts of the heuristic approach proposed in Reference 15 to handle the path constraint.

NBBS path selection, both for point-to-point and point-to-multipoint, has a polynomial order of complexity. This is crucial since the wide ranges of traffic characteristics and QOS requirements of applications make path computation expensive in terms of processing and memory requirement. Yet, these algorithms have real-time or near-real-time requirements to support on-demand connections (i.e., find paths as the connection requests arrive at the network).

Other components of NBBS related to the path selection function such as the topology distribution, control point spanning tree, bandwidth management, congestion control, and network connection management functions are presented in other papers in this issue.[10,17,18] The remainder of this paper is organized as follows. Bandwidth management framework related to connection admission control at the links is reviewed next. This computation is needed to prune the network graph by deleting links that do not have available bandwidth to support the new connection. The use of delay and connection priorities and their interaction with the available bandwidth in the network is also discussed in this section. Point-to-point and point-to-multipoint path selection algorithms are discussed in the subsequent two sections. Finally, we conclude with a summary of the unique features of the NBBS path selection framework and further research.

Bandwidth computation

In this section, we focus on NBBS bandwidth management aspects that are relevant to path selection; the interested reader is referred to Reference 18 for the details on NBBS bandwidth management. Bandwidth computations are performed during path selection to determine if a link can accommodate the requested connection based on the traffic characteristics of the connection and the current reservation levels in the network.

An NBBS link supports four delay priorities:[10] real-time-1 (RT1), real-time-2 (RT2), nonreal time (NRT) and best effort (BE). These four delay priorities are implemented as four queues of descending service priorities in the order they are listed above. Delay priority RT1 is intended for real-time applications with constant bit rate and stringent delay/jitter requirements such as voice circuit emulation and the asynchronous transfer mode (ATM) constant bit rate (CBR) service. Delay priority RT2 is intended for real-time applications in which the amount of traffic submitted to the network varies over time (variable bit rate sources). These applications also have less stringent delay/jitter requirements than those that use the RT1 queue. NRT delay priority is intended for nonreal-time applications that are less sensitive to end-to-end delay. Finally, BE delay priority is intended for applications that do not require explicit QOS guarantees such as the ATM Forum's unspecified bit rate (UBR) and available bit rate (ABR) services. Each delay priority queue is served in a first-in first-out (FIFO) manner while scheduling among the delay queues is head-of-line (HOL). Accordingly, the higher priority connections affect the services of lower priority connections at the transmission link while lower priority connections are transparent to higher priority connections.

There is a link manager at each switch, one for each link, that keeps track of two aggregate bandwidth reservation levels: one for RT1 + RT2 and another for NRT. Each of these aggregates is represented by a vector called *link metric*. Link metrics are distributed throughout the network using the control point (CP) spanning tree. This allows every node

to know the current reservation levels at each link. In particular, given a new connection request, each node updates the link metrics based on the characteristics of the new connection and determines if the link can admit the connection request or not. If a link is determined to have sufficient bandwidth to admit the new connection, that link is included in the network graph used during path selection; otherwise, it is excluded, i.e., the network is pruned.

As a means to provide different levels of network availability to different applications, NBBS allows the network operator to assign holding and preemption priorities to each connection.[11] The holding priority of a connection determines the ability of that connection to hold onto network resources it has reserved, whereas the preemption priority determines the ability of the connection to take away resources from other connections when the network is congested. The NBBS path selection function first attempts to find a path without preempting connections already established in the network. However, if no path is available, it starts searching for links carrying connections that can be preempted, i.e., connections with lower holding priorities. Throughout the rest of this paper, we assume that the network graph has been pruned based on bandwidth requirements and connection preemption priorities.

Point-to-point path selection

Load balancing in the network avoids early link saturation, which may result in a greater number of connections routed over longer, and therefore costlier, alternate paths.[19,20] Another advantage of load balancing is that it tends to minimize the end-to-end delay in packet-switched networks.

The NBBS path selection algorithm finds the minimum hop path that is capable of supporting the service requirements of the connection (i.e., loss ratio and either end-to-end delay or delay variation) while attempting to balance the load among paths with similar characteristics and to control when and how calls can be routed over costlier (longer) paths.

The motivation for favoring minimum hop paths is to minimize the amount of network resources allocated to a connection, allowing more connections to be supported, and therefore improving the total network throughput. This is inspired from heuristics used in circuit-switched networks where direct paths are favored. It is noted, however, that even in circuit-switched networks, doing so does not always yield optimal routes. This simple rule has, however, generally been observed to generate "reasonable" network throughput and it is used in NBBS. It is also noted that favoring minimum number of hops often conflicts with the load balancing criterion (a heavily loaded two-hop path is selected over a lightly loaded three-hop one). Next we present the NBBS path selection framework in a more precise way.

NBBS path selection finds the minimum hop path that supports the service required.

Denote a directed graph representing the network (after being pruned as described in the previous section) by G = (V, E), where V and E are the set of vertices and the set of edges, respectively. Vertices represent NBBS nodes and edges represent unidirectional links. Given the source and destination vertices, the point-to-point path selection algorithm computes two unidirectional paths—one for each direction—separately.

Let d: $E \rightarrow R^+$ and w: $E \rightarrow R^+$ be the delay and load balancing functions, respectively. The delay function $d(l)$ represents the maximum delay or jitter guaranteed across link l. The load balancing function $w(l)$ is an increasing function of the load on link l, akin to the first derivative of a delay-type function. Specifically,

$$w(l) = \frac{C_l}{(C_l - B'_l)(C_l - B_l)}$$

where $B_l = \max\{B_{l,RT}, B_{l,NRT}\}$ is the current reserved bandwidth and $B'_l = \max\{B'_{l,RT}, B'_{l,NRT}\}$ is the reserved bandwidth if the new connection were to be added onto the link. Then, the path selection

problem is to find a path P connecting the origin and destination vertices $p, q \in V$ that minimizes

$$\left(\sum_{l \in P} 1, \sum_{l \in P} w(l) \right)$$

subject to

$$\sum_{l \in P} d(l) \leq d_{\max}$$

where the vector ordering is such that

$$(x_1, y_1) < = (x_2, y_2) \text{ if } x_1 < x_2 \text{ or}$$
$$x_1 = x_2 \text{ and } y_1 < = y_2$$

NBBS path selection algorithm is an extension of the Bellman-Ford algorithm[4] and consists of the following two steps (p and q denote origin and destination vertices, respectively):

1. Compute the smallest number of hops, h^*, from p to q that satisfies the delay constraint $d_{\max}$ using the modified Bellman-Ford-Ford algorithm as follows. Starting from $h = 1$, compute the minimum delay $D(h, n)$ from origin vertex p to each vertex n in exactly h hops, i.e.,

   $$D(h, n) = \min_{l \in I_n} D(h - 1, m(l)) + d(l),$$

   where I_n is the set of all links terminating at vertex n and $m(l)$ is the originating vertex of link l. ($D(0, p)$ is initialized to 0 and $D(0, n)$ to ∞ for $n \neq p$.) The above operation is repeated with $h = h + 1$ unless $D(h, q) \leq d_{\max}$, in which case $h^* = h$.

2. Find an $h^* - hop$ path satisfying the delay constraint that "locally" minimizes the load balancing weight as follows. Let L denote the set of links that are part of an $h^* - hop$ path satisfying the delay constraint. Starting from the destination vertex q, links terminating at q that are in L can be identified. Link l is such a link if it terminates at q and satisfies $D(h^* - 1, m(l)) + d(l) <= d_{\max}$. Among these links, a link with minimum load balancing weight can be determined. Let l_1 denote this link. Links in L that terminate at $m(l_1)$ can now be identified, i.e., link l such that it terminates at $m(l)$ and satisfies $D(h - 2, m(l)) + d(l) <= d_{\max} - d(l_1)$. Among the links belonging to L and terminating at $m(l_1)$, one that has the smallest load balancing weight can be determined. The process is repeated until the origin vertex is reached.

The pseudocode for the algorithm is given below:

```
Set D(0, p) = 0, D(0, n) = ∞ for n ≠ p, h = 0
Repeat
  h = h + 1
  For n ∈ V
    Set D(h, n) = min{D(h − 1, m(l)) + d(l)|l ∈ I_n}
Until D(h, q) ≤ d_max or h ≥ h_max
If D(h, q) > d_max
  Stop/*No feasible path is found*/
Else
  Set d'_max = d_max, n = q
  Repeat
    Set h = h − 1, w_min = ∞
    For l ∈ I_n
      If D(h, m(l)) + d(l) ≤ d'_max and w(l) < w_min
        Set w_min = w(l), l_min = l
    Set P = l_min, P/* P is the desired path*/
    Set d'_max = d'_max − d(l_min)
    Set n = m(l_min)
Until h = 0
```

Further comments about the path selection are:

1. The pruning of the network graph described in the previous section does not have to be done separately from the path selection algorithm itself. In particular, it can be performed as the algorithm progresses by introducing conditions at appropriate places.
2. A predetermined hop count $h_{\max}$ can be used to limit the number of hops along the end-to-end connection.
3. Note that the algorithm described above guarantees that the delay requirement is satisfied if a path is found. Among all paths that satisfy the delay constraint, this path is the one with the minimum hop count. The load balancing weight is minimized at each hop moving from the destination to the origin vertex.

In cases where the reduced graph formed by those links that are part of a minimum hop path satisfy the delay constraint is sparse, the end-to-end load balancing weight is more likely to be minimized.

Primary and secondary links

The path selection function performs a preliminary connection admission control by rejecting a connection request if no path that meets both the end-to-end delay and bandwidth requirements can be found. Certain feasible paths (paths meeting both the QOS and CAC criteria) may still be rejected if some of the constituent links are likely to be better used by future connections. In particular, resources assigned to a connection along a path may later be put to a better use and allow the network to carry more connections that would otherwise have been rejected. If that is the case, it is preferable to block the first connection despite the fact that it could have been accepted. Several approaches have been proposed, mostly in the context of circuit-switched networks, to deal with this problem. For example, the shadow cost[21,22] or state dependent[23,24] approaches attempt to forecast the impact on the network revenue of accepting a new connection on a given path. Connections are accepted only if they are expected to have a positive impact on the network revenue.

NBBS best-effort traffic is supported by a rate-based mechanism and path selection.

In addition to the network throughput aspect, there are also fairness reasons that may lead to the rejection of some connections even when a feasible path has been identified. For example, in the simple case of three nodes A, B, C in tandem, traffic from B and destined to C should not be allowed to shut off all the traffic from A to C despite the fact that it requires fewer resources (one versus two links). In packet-switched networks, this problem is known as starvation of some nodes and special attention must be given to avoid this problem.

The NBBS routing algorithm addresses these issues by introducing a concept similar to that of trunk reservation in circuit-switched networks[19,20,25] to avoid both the potential instability and unfairness problems as the network load increases. Accordingly, the use of longer alternate paths is prevented when link loads exceed certain thresholds. This attempts to ensure that excess traffic is carried only when there are enough idle resources. The approach relies on the use of primary and secondary paths associated with each origin and destination pair (OD pair) in the network.

For a given OD pair, primary paths are the minimum hop paths. Links that belong to the primary paths of an OD pair are identified as primary links for that pair, while all other are deemed secondary links. Note that not all links on a secondary path are necessarily secondary links, as some links may also belong to primary paths for the same OD pair. Associated with each link are the primary and secondary load thresholds (reservable capacities, R), which are used to determine if the path can support the connection. A primary path is acceptable if the loads on all its (primary) links are below the levels corresponding to primary load thresholds, and the load levels on all its secondary links (there must be at least one, otherwise the path would not be secondary) are below the secondary load thresholds. Primary and secondary load thresholds are typically set at 100 percent and 95 percent respectively of the maximum reservable link capacity, and have been found effective in maintaining throughput while preserving fairness at high network loads.[26]

For a given OD pair, primary links can be identified using an algorithm very similar to the NBBS path selection algorithm. Note that whether a path connecting a given OD pair is primary or not depends on the QOS and bandwidth requirements of the connection request. For example, consider an OD pair connected by one slow link with a 10 megabits per second (Mbps) total capacity. The same pair is also connected by two faster links, each with 50 Mbps total capacity. The single slow link is primary for a connection requiring, for example, 5 Mbps bandwidth, while the two faster links are secondary links. For a connection requiring more than 10 Mbps, however, the two faster links are primary since the single slow link cannot be used.

The steps used to find all primary links given an OD pair (p, q) and a connection with given delay and bandwidth requirements are summarized as follows:

Prune the network graph as described in the previous section assuming that the connection is the only one in the network. Find h^* as described in Step 1 of the path selection algorithm. Find all the primary links as those that are part of an $h^* - hop$ path satisfying the delay constraint as follows: Let us denote the set of such links by L_p and let $\bar{D}(h, n)$ be the minimum delay from an immediate vertex n to the destination vertex q in exactly $h - hops$ using links belonging to L_p. In general, a link l with end points m and n, is in L_p and is $h^* - h$ *hops* away from q if

$$D(h-1, m) + d(l) \leq d_{\max} - \bar{D}(h, n)$$

Starting from the destination vertex q, links terminating at q that are in L can be identified. $\bar{D}(1, n)$ can then be computed for every intermediate vertex n one hop away from q. Links belonging to L that are one hop away from q can be identified and $D(2, n)$ can be computed using the Bellman-Ford equation. These steps are repeated until the origin vertex is reached.

A pseudocode of the algorithm to find all primary links is given below:

```
Set D(0, p) = 0, D(0, n) = ∞ for n ≠ p, h = 0
Repeat
  h = h + 1
  For n ∈ V
    Set D(h, n) = min{D(h − 1, m(l)) + d(l)|l ∈ I_n}
Until D(q, h) ≤ d_max or h ≥ h_max
If D(q, h) > d_max
  Stop/*No feasible path is found*/
Else
  Set h* = h
  Set D̄(h, n) = ∞ for h = 0, . . . , h*, n ∈ V
  Set D̄(0, q) = 0
  Repeat
    Set h = h − 1
    For n ∈ V
      For l ∈ I_n
        If D(h, m(l)) + d(l) ≤ d_max − D̄(h + 1, n)
    Set L_p = L_p ∪ {l}
    Set D̄(h, m(l)) = min{D̄(h, m(l)),
                         D̄(h + 1, n) + d(l))},
Until h = 0
```

The complexity of the algorithm

The computational complexity of both the path selection algorithm and the algorithm to find the primary links is similar to that of the Bellman-Ford shortest path algorithm. From the pseudocode, it can be seen that the number of iterations of the main steps is a multiple of

$$h_{\max} \sum_{n \in V} |I_n|,$$

where $|I_n|$ is the number of links terminating at vertex n. Assuming $|I|_{\max}$ is the maximum number of links that can terminate in a vertex, the number of iterations is bounded by a multiple of $h_{\max}|V||I|_{\max}$ which is actually linear with $|V|$. In general, the worst-case time complexity of the algorithm is $O(V^3)$, if no assumption on the network topology is made.

Path selection for best-effort connections

Best-effort connections do not require any explicit QOS from the network. Local area network (LAN) data traffic is a typical example of this service class. Implicitly, however, it is required to reduce the loss ratio this traffic will see in the network as much as possible.

Best effort service in NBBS uses nonreserved delay priority (i.e., the lowest delay priority). A consequence of this is that the nonreserved queue is served only when there is no other packet waiting in the other queues for transmission.

NBBS uses two mechanisms to support best-effort traffic in the network: an end-to-end rate-based mechanism that regulates the rate at which traffic is submitted to the network, and path selection. Rate regulation allows the source to increase or decrease its submission rate to the network based on the measured delay along the path of the traffic. Hence, if the path is not heavily utilized by the higher priority traffic, then a best-effort source is allowed to increase its traffic submission rate. If the path is highly utilized, then these sources are forced to reduce their rates.

Rate-based control of best-effort traffic regulation achieves the objective of reducing the loss ratio of the connection at the expense of possibly delaying the traffic at the source. In order to reduce this delay, the path selection function attempts to find a path in the network that reduces the possibility of long delays. This is achieved by allocating the

best-effort connections along the paths that are utilized lightly.

Link utilization that defines the percentage of time the link is busy transmitting traffic is a link metric advertised by each link and distributed to every node in the network. The metrics are updated every time the link utilization at a link changes significantly. Source nodes use this information to find the least utilized path in the network at the time the best effort connection request arrives. The path selection function in this case is an application of the Bellman-Ford shortest path algorithm that uses link utilization as the link metric and attempts to find the lightly utilized path that is likely to be able to support the best-effort traffic.

Point-to-multipoint path selection

The term *multimedia* is used to refer to concurrent presentation of two or more applications such as voice, data, video, and image. Examples of multimedia applications include teleconferencing, entertainment video, medical imaging, advertising, and education.

Multimedia applications are distinguished in various ways:[27] First, there are requirements for "synchronization" among various information types, which can range from a coarse synchronization such as sequencing the transmission of various objects (e.g., image followed by voice followed by image and so on) to a more precise synchronization such as synchronization of voice to the speaker's lip motion. Second, there are performance restrictions on the end-to-end delay value, referred to as latency, as well as instantaneous variations in latency, referred to as jitter. Furthermore, multimedia applications typically take place between a group involving more than two users. The communication within the group is usually symmetric, that is, any group member may send information to any other group member. Accordingly, the network should provide efficient multicast transmission capabilities to support multimedia applications.

In general, there are three methods that can be used to establish connections among a group of users:

- Establish point-to-point connections from each node that generates traffic to all other group members.
- Construct a single tree with full duplex links that provides a path from each group member to all the other group members.
- Construct as many trees with half duplex links as there are group members that generate traffic that provides a point-to-multipoint path from the root (i.e., one of the members) of the tree to all the other group members.

The first method requires as many as N^2 connections to be established with N being the number of group members. This approach imposes a significant processing burden to the network, particularly as the number of group members increases. Each individual connection in this approach is managed separately, increasing the complexity of the management functions. In addition, the amount of network resources used for a single application—connection identifiers, table entries used at the intermediate nodes, as well as the total bandwidth used by the point-to-multipoint connection—increases artificially, limiting the effective utilization in the network.

The second method minimizes the network resources used to establish multipoint connections, and routing in the network becomes much simpler when, potentially, a single identifier is used for routing at the intermediate nodes. However, constructing such a tree is a nonpolynomial- (NP-) complete problem,[28] when the bandwidth requirements of each connection are taken into consideration. Another problem with this approach is that when a new node is to be added to the group, the existing tree may no longer be capable of supporting the additional traffic generated by the new group member. This may require a new tree to be established every time a new node joins the group.

The third method provides a point-to-multipoint communication capability on a single tree, thereby requiring the establishment of N trees. Point-to-multipoint connections arise naturally in various multimedia applications as well, in which the communication is not symmetric, e.g., video distribution. Using these trees reduces the complexity of multipoint connections compared with the first method and eliminates the problems associated with new members joining the group in the second method. Hereafter, point-to-multipoint trees are referred to as *multicast trees*.

A multicast tree is a collection of half duplex transmission links spanning the nodes on which the

group members reside. Messages entering the tree from one group member, referred to as the *root*, are routed and copied as necessary by the intermediate nodes to be delivered to all group members. In the simplest case, it is possible to construct a multicast tree with a given root, if one exists, by forming a spanning tree on a directed graph that originates at the root. The multicast tree can then be constructed by trimming the tree to eliminate all leaf nodes that are not group members. However, it is necessary to meet the service requirements of applications along the links on this tree. In addition, the network provider's objective is to maximize the effective utilization of its network resources, thereby minimizing the amount of resources used for connections. When an optimization criterion is introduced to the construction of a multicast tree, then the problem, known as the Steiner tree problem, becomes NP-hard if the number of group members is less than the total number of nodes in the network.

Next, we present an algorithm that constructs a point-to-multipoint tree, which attempts to minimize the total number of links used to construct the tree and satisfy end-to-end delay requirements of applications. This algorithm can be used in two ways: tree creation and tree extension. In creating a tree, the algorithm starts at the origin node and constructs a multicast tree spanning all group members. In the case of extending the tree, the algorithm starts with the existing tree and extends it to new nodes joining the group.

The input to the algorithm is the maximum end-to-end delay required by the application, the network topology, and the (initial) set of group members that the communication will involve. The output of the algorithm is the multicast tree, providing a point-to-multipoint connection from the root to all the other group members.

Multicast tree algorithm

The algorithm is based on the following procedure: Start with an initial tree and extend it out to one of the other group members to form a larger tree, so that the number of links used to extend the tree are minimized while satisfying the delay constraint. This step is repeated until either all the group members are included in the multicast tree or it is found out that some group members cannot be reached. In reaching out to a destination node, the procedure finds the first group member with a minimum hop count away from the current tree that satisfies the maximum end-to-end delay constraint from the root. As a result, the final multicast tree approximately minimizes the total number of links used to provide a point-to-multipoint connection, while guaranteeing the maximum end-to-end delay constraint of the application. Furthermore, if there exists a feasible tree (i.e., that satisfies the delay requirement), the algorithm is guaranteed to find it. The steps of the algorithm can be summarized as follows:

Step 0. Initially, only the root node belongs to the tree.

Step 1. Initialize the length of the root to every node, where the length of a path to node i is defined as the total delay of going from the root to node i over tree links. A node that belongs to the tree is considered to be a zero hop away from the tree. The lengths of the nodes that do not belong to the current tree are initialized to infinity for all hop counts.

Step 2. Find a group member j that is a minimum hop away from the current tree and satisfies the end-to-end delay constraint.

Step 3. Trace the path from the root to node j and determine if there is a loop in the current tree. Loop, in this context, refers to a cycle in the tree obtained without considering the directions of links.

Step 4. If there is a loop, then eliminate the loop by applying the minimum spanning tree algorithm to the current tree (in fact the current graph is no longer a tree) and eliminate all leaf nodes that are not group members.

Step 5. If there is any group member that does not belong to the current tree, go to Step 1. Else, STOP.

Conclusions

Path selection function in broadband networks is a complex service necessary to provide service guarantees to different applications with different QOS requirements. In NBBS, providing service guarantees is a comprehensive solution that includes the bandwidth management and the path selection functions.

Finding an optimum path subject to one or more constraints is, in general, an NP-complete problem. That is, there is no known algorithm that can solve the problem in polynomial time. Accordingly, the problem needs to be addressed heuristically. In NBBS, the loss requirements of connections is determined by the bandwidth management function that produces the amount of bandwidth needed to support the connection on a particular link. This value is checked in the path selection function to prune the network graph by eliminating links that cannot support the connection (i.e., cannot provide the required loss ratio guarantee). The current NBBS algorithm is then used to find a minimum hop path that satisfies one additive constraint (i.e., delay or jitter, but not both simultaneously). In addition, the path selection algorithm attempts to balance the load over equally desirable paths (i.e., with the same number of hops while satisfying the delay constraints) to achieve high network throughput. Other features of the algorithm include connection and preemption priorities and the use of primary and secondary paths.

The path selection framework also includes a heuristic developed to construct a multicast tree. The algorithm produces a unidirectional tree that provides a point-to-multipoint connection from a node to a group of nodes with end-to-end delay constraints with a polynomial time complexity. Various properties of the multicast algorithm include the guarantee of the construction of a multicast tree on a graph with directional links (if there exists one), minimizing (approximately) the number of links used to construct the tree, and the guarantee of the maximum end-to-end delay requirements of applications.

Emerging multimedia applications require the solution to multiconstraint optimization problems that would include delay, jitter, and administrative weight as the constraints. Satisfying all these QOS requirements simultaneously appears to be much more difficult than the single constraint optimization problem. As these are all NP-complete problems, the proposed solutions would be necessarily heuristics. The current NBBS path selection framework is being extended to address these requirements with an algorithm that works fairly well in most cases with a polynomial worst-case time complexity.

Acknowledgments

The authors gratefully acknowledge the valuable contributions of members of the NBBS team to the NBBS path selection framework.

Cited references

1. A. Ephremides, "The Routing Problem in Computer Networks," *Communication and Networks*, I. F. Blake and H. V. Poor, Editors, Springer-Verlag, New York (1987), pp. 299–324.
2. N. Deo and C. Pang, "Shortest Path Algorithms: Taxonomy and Annotation," *Networks* **14**, 275–323 (1984).
3. E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik* **1**, 269–271 (1959).
4. L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ (1962).
5. R. W. Floyd, "Algorithm '97, Shortest Path," *Communications of the ACM* **5**, 345 (1962).
6. L. Fratta, M. Gerla, and L. Kleinrock, "The Flow Deviation Method: An Approach to Store and Forward Communication Network Design," *Networks* **3**, 97–133 (1973).
7. P. A. Humblet and S. R. Hollaway, *Algorithms for Data Communication Networks—Parts I and II*, Motorola Codex, 20 Cabot Blvd., Mansfield, MA 02048.
8. J. M. Jaffe, "Algorithms for Finding Paths with Multiple Constraints," *Networks* **14**, 95–116 (1984).
9. K. J. Lee, D. Towsley, and M. Choi, "Distributed Algorithms for Minimum Delay Routing with Constraints in Communication Networks," *Proceedings IEEE Infocom '87* (1987), pp. 188–199.
10. 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).
11. P. Winter, "Steiner Tree Problems in Networks: A Survey," *Networks* **17**, 129–167 (1987).
12. L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica* **15**, 141–145 (1981).
13. J. Plesnik, "A Bound for the Steiner Tree Problem in Graphs," *Mathematica Slovaca* **31**, 155–163 (1981).
14. H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Tree Problem in Graphs," *Mathematica Japan* **24**, 573–577 (1980).
15. A. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicasting for Multimedia Applications," *Proceedings IEEE Infocom '92*, Florence, Italy (1992), pp. 2078–2085.
16. A. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Transactions on Networking* **1**, No. 3, 286–292 (1993).
17. M. Peyravian, R. Bodner, C.-S. Chow, and M. Kaplan, "Efficient Transport and Distribution of Network Control Information in NBBS," *IBM Systems Journal* **34**, No. 4, 640–658 (1995, this issue).
18. 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).
19. E. W. Wong and T.-S. Yum, "Maximum Free Circuit Routing in Circuit-Switched Networks," *Proceedings IEEE Infocom '90*, San Francisco (1990), pp. 934–937.
20. G. R. Ash, "Use of a Trank Status Map for Real-Time DNHR," *Proceedings 11th International Teletraffic Con-*

gress, M. Akiyama, Editor, 4.4A-4.1-4.4A-4.7, Elsevier Science Publishers (North Holland), Kyoto, Japan (1985).
21. F. P. Kelly, "Routing in Circuit-Switched Networks: Optimization, Shadow Prices, and Decentralization," *Advanced Applied Probability* **20**, 112–144 (1988).
22. F. P. Kelly, "Fixed Point Models of Loss Networks," *Journal of Australian Mathematical Society*, Ser. B, 31-2, 204–218 (1989).
23. T. J. Ott and K. R. Khrishnan, "State Dependent Routing of Telephone Traffic and the Use of Separable Routing Schemes," *Proceedings 11th International Teletraffic Congress*, M. Akiyama, Editor, 5.1.A-5.1-5.1.A-5.6, Elsevier Science Publishers (North Holland), Kyoto, Japan (1985).
24. K. R. Khrishnan and T. J. Ott, "Forward Routing: A New State Dependent Routing Scheme," *Proceedings 12th International Teletraffic Congress*, M. Bonatti, Editor, Elsevier Science Publishers (North Holland), Torino, Italy (1989), pp. 1026–1032.
25. J. M. Akinpelu, "The Overload Performance of Engineered Networks with Nonhierarchical and Hierarchical Routing," *Bell Systems Technical Journal* **63**, No. 7, 1261–1281 (1984).
26. H. Ahmadi, J. S.-C. Chen, and R. A. Guérin, "Dynamic Routing and Call Control in High Speed Integrated Networks," *Proc. Workshop Sys. Eng. Traf. Eng.*, International Teletraffic Congress, ITC-13, Copenhagen, Denmark (1991), pp. 397–403.
27. R. O. Onvural, *ATM Networks: Performance Issues*, 2nd edition, Artech House, Norwood, MA (1995).
28. M. R. Garay and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP Completeness*, W. H. Freeman and Co., San Francisco, CA (1979).

Accepted for publication July 13, 1995.

Theodore Tedijanto *IBM Networking Hardware Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709.* Dr. Tedijanto received B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Maryland in 1984, 1986, and 1990, respectively. While a member of the IBM family (1990–1995), he worked in the Networking Architecture group at Research Triangle Park. His area of interest includes traffic management and route selection algorithms for high-speed networking architectures. Dr. Tedijanto is an active member of IEEE and continues as a member of the ATM Forum, working on ATM routing and traffic management. He has published a number of papers in various journals and conference proceedings in the areas of queuing theory and bandwidth management.

Ralf O. Onvural *IBM Networking Hardware Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (electronic mail: onvural@vnet.ibm.com).* Dr. Onvural is a senior engineer at IBM's Research Triangle Park facility in the Networking Architecture organization and manages the Networking Technology Architecture department that develops network control services for ATM networks. He is also IBM's venue owner for the ATM Forum and has been attending the forum meetings since February 1993. Dr. Onvural organized several international conferences on high-speed networks, in general, and ATM networks, in particular. He has published in various journals and conferences, and has edited five books. He is also the author of the book *Asynchronous Transfer Mode Networks: Performance Issues.*

Dinesh C. Verma *IBM Research Division, Thomas J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, New York 10532 (electronic mail: verma@watson.ibm.com).* Dr. Verma is a research staff member in the Advanced Networking Laboratory. He received his B.Tech. in computer science from the Indian Institute of Technology at Kanpur, India in 1987, and his Ph.D. from the University of California at Berkeley in 1991. He is interested in network control architecture and software for high-speed networks, performance analysis, and multimedia applications.

Levent Gün *IBM Networking Hardware Division, P.O. Box 12195, Research Triangle Park, North Carolina 27709.* Dr. Gün received a B.A. in mathematics and a B.S. in electrical engineering from Bogazici University, Turkey, in 1983. He earned M.S. and Ph.D. degrees in electrical engineering from the University of Maryland in 1986 and 1989, respectively. While a member of the IBM family (1989–1994), Dr. Gün worked in the Networking Architecture group at Research Triangle Park. In addition, he held the position of Adjunct Assistant Professor in the Operations Research Department at the University of North Carolina, Chapel Hill. His interest continues in the development and analysis of high-speed networking architectures. He has published over a dozen papers in various journals and conference proceedings in the areas of computational probability, queuing theory, and stochastic control. He is an active member of IEEE and ORSA.

Roch A. Guérin *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (electronic mail: guerin@watson.ibm.com).* Dr. Guérin received the Diplome d'Ingénieur from the École Nationale Supérieure des Télécommunications, Paris, France, in 1983, and his M.S. and Ph.D. from the California Institute of Technology, both in electrical engineering, in 1984 and 1986, respectively. Since August 1986 he has been with IBM at the Thomas J. Watson Research Center, where he now manages the Broadband Networking group in the Advanced Networking Laboratory. His current research interests are in the area of modeling, architecture, and quality-of-service issues in high-speed networks. In particular, he is interested in developing techniques to map network-level performance measures to corresponding quantities at the application level, and in understanding issues related to the interactions between different networking technologies and protocols. He is also interested in understanding how the new capabilities and flexibility available from high-speed networks can translate into better services to applications. Dr. Guérin is a member of Sigma Xi and the IEEE Communications Society, and is an editor for the *IEEE/ACM Transactions on Networking.* He was an editor for the *IEEE Transactions on Communications* and the *IEEE Communications Magazine.*

Reprint Order No. G321-5585.