Multicast Routing and Its QoS Extension: Problems, Algorithms, and Protocols

Bin Wang and Jennifer C. Hou, The Ohio State University

Abstract

Multicast services have been increasingly used by various continuous media applications. The QoS requirements of these continuous media applications prompt the necessity for QoS-driven, constraint-based multicast routing. This article provides a comprehensive overview of existing multicast routing algorithms, protocols, and their QoS extension. In particular, we classify multicast routing problems according to their optimization functions and performance constraints, present basic routing algorithms in each problem class, and discuss their strengths and weaknesses. We also categorize existing multicast routing protocols, outline the issues and challenges in providing QoS in multicast routing, and point out possible future research directions.

Multicast services have been increasingly used by various continuous media applications. For example, the multicast backbone (MBone) of the Internet has been used to transport real time audio and video for news, entertainment, and distance learning. Instead of sending a separate copy of the data to each individual group member, a multicast source sends a single copy to all the members. An underlying *multicast routing* algorithm determines, with respect to certain optimization objectives, a multicast tree connecting the source(s) and group members. Data generated by the source flows through the multicast tree, traversing each tree edge exactly once. As a result, multicast is more resource-efficient, and is well suited to applications such as video distribution.

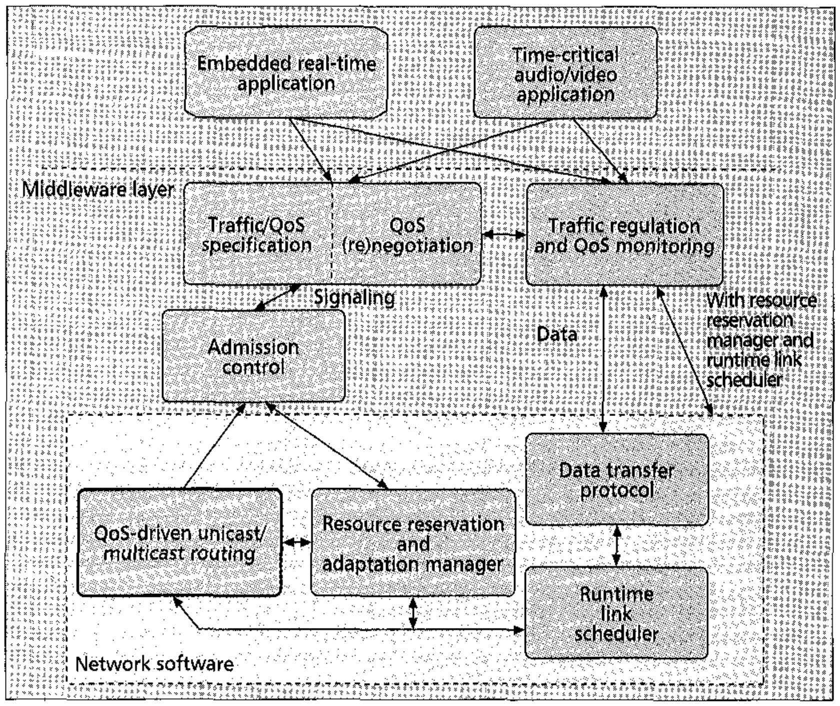
With fast development of hardware technologies, commercialization of the Internet, as well as the increasing demand for quality of service (QoS) fueled by emerging continuous media applications, offering guaranteed and better than best effort services will add to the competitive edge of a successful service provider. The notion of QoS was proposed to capture the qualitatively or quantitatively defined performance contract between the service provider and the user applications. QoS provisioning entails the development of several essential techniques: definition and specification of QoS, design of *QoS-driven* (also called *constraint-based*) unicast/multicast routing protocols, packet scheduling algorithms for link sharing, as well as resource reservation and management. Figure 1 gives a modular collection of, and the relationship among, these techniques for QoS provisioning.

In an effort to provide QoS, a number of services have been defined (e.g., *QoS-guaranteed* and *controlled load* services in the integrated services architecture [1]). A Resource Reservation Protocol (RSVP) has been developed to provide receiver-initiated fixed/shared resource reservations for unicast/multicast data flows [2]. Numerous rate-based packet scheduling algorithms have been proposed and analyzed to provide service differentiation and fair link sharing. To avoid per-flow state and queue management, the differentiated services architecture, which relies on packet tagging and lightweight router support, has also been proposed [3, 4] to provide *premium* and *assured* services [5].

Not until recently has the issue of QoS support for multicast routing been extensively addressed. QoS-driven multicast routing amounts to finding the best distribution tree, with respect to certain performance-related constraints, to better utilize network resources and to support the QoS requirements of applications. It is an indispensable component in a QoS-centric network architecture. For example, RSVP in integrated services relies on a unicast/multicast routing protocol to provide a unicast route/multicast tree. If the located route/tree does not have sufficient resources for RSVP to reserve, RSVP incurs reservation errors, an alternative route/tree has to be identified, and the reservation process is retried. An alternative option is to establish the connection at degraded QoS. It would be desirable if this trial-and-error process could be avoided and RSVP provided a route/tree with sufficient resources in the first place. Similarly, in differentiated services, a *bandwidth broker* [5] in each domain is

The work reported in this article was supported in part by NSF under Grant No. ANI-9804993 and No. ANI-9903160. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the funding agencies.

responsible for negotiating service-level agreements between neighboring domains and setting up the corresponding service profiles. The bandwidth broker also interacts with the (intra/interdomain) routing protocols to set up appropriate routes. It would be desirable if bandwidth brokers were provided with QoS-satisfying routes/trees.


Embedded real-time application
Time-critical audio/video application
Middleware layer
Traffic/QoS specification
QoS (re)negotiation
Traffic regulation and QoS monitoring
Signaling
Admission control
Data
With resource reservation manager and runtime link scheduler
Data transfer protocol
QoS-driven unicast/ multicast routing
Resource reservation and adaptation manager
Runtime link scheduler
Network software

■ Figure 1. *A modular collection of network techniques for QoS provisioning.*

The problem of providing QoS in multicast routing is difficult due to a number of reasons. First, distributed continuous media applications such as teleconference, video on demand, Internet telephony, and Web-based applications have very diverse requirements for delay, delay jitter, bandwidth, and packet loss probability. Multiple constraints often make the multicast routing problem intractable. Second, there are many practical issues that have to be taken into account when a routing algorithm is incorporated as part of a multicast routing protocol (e.g., state collection and update, handling of dynamic topology and membership changes, tree maintenance, and scalability). Figuring in QoS further complicates the protocol design process. Moreover, one has to consider how to collect/maintain QoS-related state at minimal cost, how to construct a QoS-satisfying route/tree in the presence of aggregated imprecise state information, and how to maintain QoS across routing domains.

In this article we provide a survey of recent advances in multicast routing algorithms/protocols, with an emphasis on QoS issues. We first give an overview of multicast routing, and classify multicast routing problems according to their objective functions and tree or link constraints. We present multicast routing algorithms in each class of the routing problems. We then categorize, based on how multicast trees are structured, existing multicast routing protocols and discuss the components that comprise multicast routing protocols. Following that, we present the issues and challenges in providing QoS support to multicast routing protocols, and discuss the various solution approaches in the literature. We conclude this article by pointing out several possible future research directions.

Multicast Routing Problems

The Network Model

As far as multicast routing is concerned, a network is usually represented as a weighted digraph $G = (V, E)$, where V denotes the set of nodes and E the set of communication links connecting the nodes. $|V|$ and $|E|$ denote the number of nodes and links in the network, respectively. Without loss of generality, only digraphs are considered in which there exists at most one link between a pair of ordered nodes. Associated with each link are parameters that describe the current status of the link. For example, one may define a link-delay function $d: E \rightarrow R^+$ which assigns a nonnegative weight to each link in the network. The value of $d(\ell)$ is a measure of the delay that packets experience on link $l \in E$, and takes into account the queuing delay, transmission time, and propagation delay. One may also define the bandwidth available on an outgoing interface as a link parameter. These parameters are collectively termed *link state*, and are usually maintained by a node. Similarly, associated with each node are parameters representing the current status of the node (e.g., buffer space available), which can be independently measured for each outgoing interface or aggregately measured for the node. These parameters are termed *node state*. The collection of the local node/link states maintained in the network is termed global *network state*.

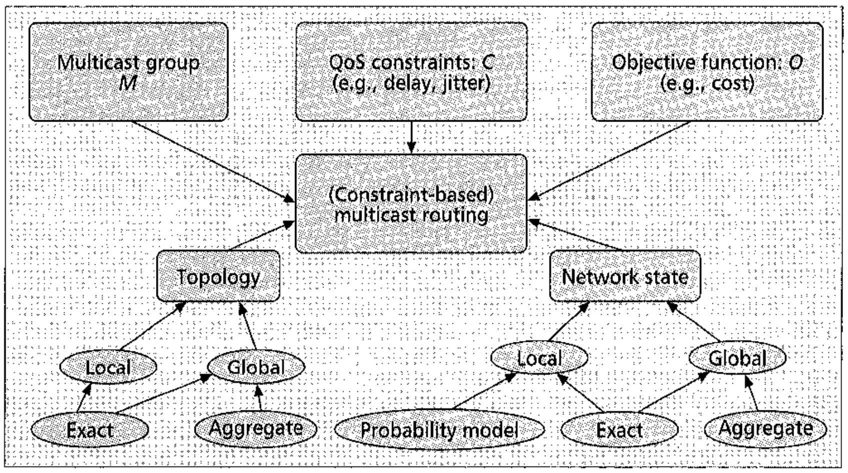
Let $M \subset V$ be a set of nodes involved in a group communication. We call set M a multicast group with each node $v \in M$ a group member. Packets originating from a source node v_s have to be delivered to a set of receiver nodes $M - \{v_s\}$. Depending on the number of sources and receivers in a multicast group, the communication paradigm may be one-to-one (unicast), one-to-many, or many-to-many. In particular, in the most general many-to-many paradigm, a group member may be a source, a receiver, or both. A multicast tree T is a subgraph of G that spans all the nodes in M. T may include relay nodes, non-group-member nodes along a path in the tree. We use $P_T(v_s, v_d)$ to denote that the path from a source node v_s to a receiver node $v_d\, M \in \{v_s\}$ in the tree T.

Classification of Multicast Routing Problems

Given a multicast group M and possibly a set of optimization objective functions O, multicast routing is a process of constructing, based on network topology and network state, a multicast tree T that optimizes the objective functions (Fig. 2). In the case of constraint-based multicast routing, a set of constraints C in the form of end-to-end delay bound, interreceiver delay jitter bound, minimum bandwidth, packet loss probability, and/or a combination thereof is given. The resulting multicast tree must provide not only reachability from source(s) to a set of destinations, but also certain QoS merits on the routes found in order to satisfy the constraints.

Objective Functions and Constraints — The optimization objectives sought are usually defined in the form of minimizing the cost of a multicast tree, where the cost may be the total bandwidth used and/or a monotonically nondecreasing function of network utilization. The constraints imposed can be classified into two categories:

• Link constraints are restrictions on the use of links for route


Multicast group M
QoS constraints: C (e.g., delay, jitter)
Objective function: O (e.g., cost)
(Constraint-based) multicast routing
Topology
Network state
Local
Global
Local
Global
Exact
Aggregate
Probability model
Exact
Aggregate

■ Figure 2. *The various components of multicast routing.*

selection. For example, one may request that the bandwidth or buffer available on a link be greater than or equal to a predetermined value.

- Tree constraints are either:
  –Bounds on the combined value of a performance metric along each individual path from the source to a receiver in a multicast tree (e.g., the end-to-end delay bound on the paths from the source to all the receivers).
  –Bounds on the difference of the combined value of a performance metric along the paths from the same source to any two different receivers, such as the interreceiver delay jitter bound defined as the difference between the end-to-end delays along the paths from the same source to any two different receivers. In the case of heterogeneous QoS, a different constraint may be imposed for each receiver.

Depending on how a tree constraint is derived from the corresponding link metrics, tree constraints can be further classified into the following three types (let $m(\ell)$ be defined as the performance metric for link ℓ):

- Additive tree constraints: For any path $P_T(u, v) = (u, i, j, \ldots, k, v)$, we say the tree constraint is additive if
  $$m(u, v) = m(u, i) + m(i, j) + \cdots + m(k, v). \quad (1)$$
  For example, the end-to-end delay d(u, v) from node u to node v, is additive and is equal to the sum of individual link metric d(i, j) along the path $P_T(u, v)$.
- Multiplicative tree constraints: We say the tree constraint is multiplicative if
  $$m(u, v) = m(u, i) \times m(i, j) \times \cdots \times m(k, v). \quad (2)$$
  For example, the probability, $1 - P_L(u, v)$, for a packet to reach node v from node u along $P_T(u, v)$ is multiplicative and is equal to the product of individual link metric $1 - P_L(i, j)$ along the path $P_T(u, v)$.
- Concave tree constraints: We say the tree constraint is concave if
  $$m(u, v) = \min[m(u, i), m(i, j), \ldots, m(k, v)]. \quad (3)$$
  For example, the bandwidth $b(u, v)$, available along a path from node u to node v, is concave and is equal to the minimum bandwidth among the links on path $P_T(u, v)$.

Classification of Multicast Routing Problems — Depending on the link/tree constraints imposed and the objective function used, a multicast routing problem can be formulated as

1)A link-constrained problem: A link constraint is imposed to construct feasible multicast trees (e.g., bandwidth-constrained routing).
2) A multiple-link-constrained problem: Two or more link constraints are imposed to construct feasible trees (e.g., bandwidth- and buffer-constrained routing).
3)A tree-constrained problem: A tree constraint is imposed to construct feasible multicast trees (e.g., delay-constrained routing).
4)A multiple-tree-constrained problem: Two or more tree constraints are imposed to construct feasible multicast trees (e.g., delay- and interreceiver-delay-jitter-constrained routing).
5)A link- and tree-constrained problem: A link constraint and a tree constraint are imposed to construct feasible multicast trees (e.g., delay- and bandwidth-constrained routing).
6) A link optimization problem: A link optimization function is used to locate an optimal multicast tree (e.g., maximization of the link bandwidth over on-tree links in a multicast tree).
7)A tree optimization problem: A tree optimization function is used to locate an optimal multicast tree (e.g., minimization of the total cost of a multicast tree). This is also known as the *Steiner tree* problem.
8)A link-constrained link optimization problem: A link constraint is imposed and a link optimization function is used to locate an optimal multicast tree that fulfills the constraint (e.g., the bandwidth-constrained buffer optimization problem).
9)A link-constrained tree optimization problem: A link constraint is imposed and a tree optimization function is used to locate an optimal multicast tree (e.g., the bandwidth-constrained Steiner tree problem).
10)A tree-constrained link optimization routing problem: A tree constraint and a link optimization function are used to locate an optimal multicast tree (e.g., the delay-constrained bandwidth optimization problem).
11)A tree-constrained tree optimization routing problem: A tree constraint and a tree optimization function are used to locate an optimal multicast tree (e.g., the delay-constrained Steiner tree problem).
12)A link and tree constrained tree optimization routing problem: Link and tree constraints and a tree optimization function are used to locate an optimal multicast tree (e.g., the bandwidth- and delay-constrained tree optimization problem).

Problems 1 and 2 are tractable, because link constraints can be fulfilled by removing from the network topology links that do not meet the selection criteria. Wang and Crowcroft [6] proved that the problem of finding a path subject to two or more independent additive and/or multiplicative constraints in any possible combination is NP-complete. The only tractable combinations are the concave constraint and the other additive/multiplicative constraints. As a result, problems 3, 5, and 10 are polynomial time solvable, while problem 4 is NP-complete. A solution algorithm of polynomial time complexity to problem 6 has been proposed in [7]. Problem 8 reduces to problem (6) if links that do not meet the link constraint are removed from the network topology. Hence, they are also polynomial time solvable. Finally, problem 7 (the Steiner tree problem) and problems 9, 11, and 12 (the constrained Steiner tree problems) have been proved to be NP-complete in [8]. Table 1 gives a summary of these problems.

Multicast Routing Algorithms

In this section we summarize several multicast routing algorithms that can be used to solve the problems classified above. A taxonomy of these multicast routing algorithms is given in Table 2.

	No optimization	Link optimization	Tree optimization
Null constraint		(6) Link optimization Polynomial time complexity	(7) Tree optimization NP-complete complexity
Link constraint	(1) Link-constrained Polynomial complexity (2) Multiple-link-constrained Polynomial complexity	(8) Link-constrained link optimization Polynomial time complexity	(9) Link-constrainedtree optimization NP-complete complexity
Tree constraint	(3) Tree-constrained Polynomial time complexity (4) Multiple-tree-constrained NP-complete complexity	(10) Tree-constrained link optimization Polynomial time complexity	(11) Tree-constrained tree optimization NP-complete complexity
Link and tree constraints	(5) Link- and tree-constrained Polynomial time complexity		(12) Link- and tree-constrained tree optimization NP-complete complexity

■ Table 1. *A taxonomy of multicast routing problems.*

Shortest Path Tree

A shortest path algorithm minimizes the sum of the weights on the links along each individual path from the source to a receiver in the multicast group. If the unit weight is used, the resulting tree is a least-hop tree. If the weight represents the link delay, the resulting tree is a least-delay tree. The Bellman-Ford and Dijkstra algorithms [9] are the two best known shortest path algorithms; both are exact and run in polynomial time. Shortest path algorithms can be used to solve tree-constrained (e.g., delay-constrained) problems.

Minimum Spanning Tree

A minimum spanning tree is a tree that spans all the group members and minimizes the total weight of the tree. The well-known centralized minimum spanning tree algorithm is Prim's algorithm [9], and a distributed version was proposed by Gallager *et al.* [10]. In Prim's algorithm the tree construction starts from an arbitrary root node and grows until the tree spans all the nodes in the network. In each step a least-cost edge connecting an off-tree node to the partial tree is added to the tree. The algorithm is greedy since the tree is augmented with an edge that contributes the minimum amount possible to the tree's total cost. Minimum spanning tree algorithms can be used to solve tree optimization problems.

Steiner Tree

The Steiner-tree-based problem aims to minimize the total cost of a multicast tree, and is known to be NP-complete [23, 24]. If the multicast group includes all nodes in the network, the Steiner tree problem reduces to the minimum spanning tree problem. Unconstrained Steiner tree algorithms can be used to solve tree optimization problems. However, they do not attempt to fulfill the tree constraints on an end-to-end basis, and hence may not be well-suited for applications with such requirements. Winter [23] and Hwang [24] did extensive surveys on both exact and heuristic Steiner tree algorithms. Bauer [25] and Salama [26] gave excellent reviews on most recent solution algorithms to the Steiner tree problem. We summarize one representative solution approach to this problem below.

KMB Heuristic — The KMB heuristic was proposed by Kou, Markowski, and Berman (hence the name) [11]. KMB applies Prim's minimum spanning tree algorithm to the *complete distance graph*, where the complete distance graph is a graph that contains all the nodes in the network and has an edge between every pair of nodes representing the shortest path between them. The heuristic works as follows:

- It creates a complete distance graph H from the original network topology G.
- It finds the minimum spanning tree U for the graph H.
- It builds a connected subgraph V by converting every node of U into its equivalent shortest path.
- It applies the minimum spanning tree algorithm to subgraph V to create a spanning tree T.
- It prunes T of nonmulticast leaves until no non-multicast leaves remain.

It has been shown that the KMB heuristic finds a tree whose cost is within twice the cost of the corresponding Steiner tree [11]. In addition to KMB, Takahashi *et al.* proposed a heuristic that constructs a tree whose cost is also within twice that of the Steiner tree [12], and Bauer and Verma proposed a distributed algorithm for solving the Steiner tree problem [13].

Constrained Steiner Tree

The Steiner tree problem has been extended to include other side constraints, for example, delay, delay jitter, or a combination thereof. These problems are also NP-complete, and heuristic algorithms are sought. Most algorithms in this category, except those in [18, 19], are centralized and source-initiated.

Zhu's Algorithm — Zhu *et al.* [15] proposed a heuristic algorithm, called the *Bounded Shortest Multicast Algorithm* (BSMA), to solve the delay-constrained tree optimization problem. They defined link cost as a function of link utilization. They also defined a superedge of a tree as the longest simple path whose internal nodes (excluding the end nodes on the path) are relay nodes, and each relay node connects exactly two tree edges. The algorithm starts by computing a least-delay tree rooted at a given source and spanning all group members. It then iteratively replaces superedges in the tree with cheaper superedges not in the tree, while not violating the delay constraint, until the total cost of the tree cannot be further reduced. Cheaper superedges are located by using a kth shortest path algorithm. BSMA always finds a delay constrained multicast tree if one exists because it starts with a least-delay spanning tree.

Kompella's Centralized Algorithm — Kompella *et al.* [16] proposed a heuristic algorithm, called the *KPP heuristic*. They assumed that the link delay, $d(u, v)$, of link (u, v) and the delay constraint D are integers, while the link cost, $C(u, v)$, of link (u, v) may take any positive real value. They defined:

- A *constrained cheapest path* between two nodes u and v as the least-cost path from node u to node v that has delay less than D
- A *closure graph* G' of a graph $G = (V, E)$ as a complete

graph over the nodes in V, with edges representing constrained cheapest paths

Given the source s and a multicast group M, KPP first computes a delay-constrained closure graph G' over $\{s\} \cup M$ using dynamic programming. The constrained cheapest path from node u to node v is then located. Then KPP uses Prim's algorithm [9] to obtain a minimum spanning tree of the closure graph G'. Starting with the source node, the tree is incrementally expanded by adding edges one at a time until all the receiver nodes are included. The edge selected each time is the one that:

- Connects an on-tree node and an off-tree node
- Does not violate the delay constraint
- Minimizes a selection function

KPP proposed two selection functions: one is the link cost, and the other strikes a balance between cost minimization and delay minimization. Finally, KPP replaces the edges in the minimum spanning tree with paths in the original graph G. Loops, if any, are removed.

Haberman's Algorithm — Haberman *et al.* [17] considered the Steiner tree problem under the delay and delay jitter constraints. The algorithm first constructs a *reference tree*, T_R, of least-cost paths from source node s to all receiver nodes. Second, for each

	Algorithm	Centr./dist.	Initiator	Tree type	Complexity	Problem solved
Shortest path tree	Dijkstra [9]	Centralized	Source	Source	$O(\|E\|\log\|V\|)$	Tree constrained
Minimum spanning tree	Prim [9]	Centralized	Source	Source	$O(\|E\|\log\|V\|)$	Tree optimization
	Gallager [10]	Distributed	Receiver	Source	$\|V\|\log_2\|V\|^{(1)}$	Tree optimization
Steiner tree	Kou [15]	Centralized	Source	Source	$O(\|M\|\|V\|^2)$	Tree optimization
	Takahashi [16]	Centralized	Source	Source	$O(\|M\|\|V\|^2)$	Tree optimization
	Bauer [17]	Distributed	Receiver	Source	$O(D \cdot \|M\|)^{(2)}$	Tree optimization
	Maxemchuk [18]	Centralized	Source	Source	$O(\|M\|\|V\|^2)$	Tree optimization
Constrained Steiner tree	Zhu [19]	Centralized	Source	Source	$O(k\|V\|^3\log\|V\|)$	Delay-constrained tree optimization
	Kompella [20]	Centralized	Source	Source	$O(\|V\|^3\Delta)^{(3)}$	Delay-constrained tree optimization
	Haberman [21]	Centralized	Source	Source	$O(kl\|M\|\|V\|^4)^{(4)}$	Delay-/delay-jitter-constrained tree optimization
	Kompella [22]	Distributed	Source	Source	$O(\|V\|^3)^{(5)}$	Delay-constrained tree optimization
	Jia [23]	Distributed	Source Receiver	Source	$O(2 \cdot \|M\|)^{(6)}$	Delay-constrained tree optimization
	Bauer [24]	Centralized	Source	Source	$O(\|M\|\|V\|^2)$	Degree-constrained tree optimization
Maximum bandwidth tree	Shacham [7]	Centralized	Source	Source	$O(\|E\|\log\|V)$	Link optimization
Miscellaneous	Rouskas [25]	Centralized	Source	Source	$O(kl\|M\|\|V\|^4)^{(7)}$	Delay-/delay-jitter-constrained
	Chen [26]	Distributed	Source Receiver	Source	$O(\|M\|\|E\|)^{(8)}$	Bandwidth-/delay-constrained

(1) Message complexity: $O(|V|\log_2|V| + |E|)$.

(2) D is the one-way trip time over the longest path between two nodes in the network or the diameter of the network. Message complexity: $O(|M||V|)$.

(3) Δ is the delay requirement. The time complexity is polynomial if Δ is a bounded integer.

(4) k is the number of paths in the initial least-cost path tree; l is the number of paths tried when adding a multicast group member.

(5) Message complexity: $O(|V|^3)$.

(6) Message complexity: $O(2 \cdot |M|)$.

(7) k and l are constants in the algorithm. A larger k or l results in a higher probability of finding a feasible tree and a higher overhead.

(8) Message complexity: $O(|E|)$.

■ Table 2. *A taxonomy of multicast routing algorithms.*

receiver node $d_i \in M$, the algorithm attempts to construct a tree, T_i, that initially contains the path in T_R from node s to node d_i. Then the algorithm augments T_i by adding "good" paths from on-tree nodes to off-tree receiver nodes, until all the receiver nodes are included. If more than one feasible tree is eventually constructed, the one with the least cost is selected.

Kompella's Distributed Algorithm — Kompella *et al.* [18] proposed a distributed heuristic algorithm to construct delay-constrained Steiner trees. The algorithm requires that every node maintain a distance vector of the minimum delay to every other node in the network. It starts with a tree that initially contains the source node, and augments the tree by adding receivers one at a time, until all the receivers are included in the tree. The approach used to select a receiver for inclusion is as follows. The source node s multicasts a `find` message via the partial tree. Upon receipt of a `find` message, a node locates an outgoing link that connects to an off-tree receiver, does not violate the delay constraint, and minimizes a selection function. The node then sends back to the source a `response` message that contains the identity of the candidate link. Upon receipt of all the responses, node s decides the best link ℓ which minimizes the selection function, and instructs that link ℓ be added to the tree. The algorithm requires multiple passes of control messages.

Jia's Distributed Algorithm — Jia presented another distributed algorithm to solve the delay-constrained tree optimization problem [19]. It is assumed (perhaps unrealistically) that the least-cost path between two nodes is always the shortest-delay path between them.

To facilitate distributed implementation, a table is used to keep track of the following information for each receiver d:
- Whether or not node d is currently on-tree
- The on-tree node, n, at which a tree branch should be grafted to connect to node d; the tree branch is a shortest path P from node n to node d that incurs the least cost and fulfills the delay constraint
- The least cost incurred in P

The algorithm starts with a tree that contains only the source node. For each receiver node d, the source node s constructs a least-delay path P from itself to node d. If such a least-delay path P satisfies the delay constraint, node s records in the table itself as the on-tree node at which a tree branch (i.e., P) is grafted to connect to node d. Node s then selects a receiver whose least-delay path incurs the least cost (let the receiver be denoted receiver j), composes a `setup` message that carries this table and the cumulative delay from node s, D (which is initialized to 0), and sends the message to receiver j. The message is sent, hop by hop, along the shortest path to receiver j.

Upon receipt of a `setup` message, an intermediate node u updates and remembers parameter D. In addition, node u checks for each currently off-tree receiver, i, if a constraint-satisfying path with a smaller cost (than the one currently recorded in the table) exists. If so, for each such receiver i, node u updates the table to reflect that a tree branch to receiver i should be grafted from node u. After the setup message reaches receiver j, receiver j selects as the next receiver to join the multicast tree the one whose least-delay path to an on-tree node incurs the least cost.

Bauer's Algorithm — By imposing constraints on the number of outgoing links that can be used for a multicast group (which was termed the *copying ability*) at each individual node, Bauer and Varma proposed a node-degree-constrained Steiner tree algorithm [20]. They proposed to modify six existing unconstrained Steiner tree heuristics. All the heuristics have a common property: a multicast tree is constructed by connecting different components. Each heuristic merges two components of a graph by the shortest path between two components. In the degree-constrained case, one or more such shortest paths may exhaust the allowable degree of a node. Thus, the heuristics are modified as follows. When a node's degree constraint is violated by a partial tree, the node and its remaining edges are eliminated from further consideration. This modified topology may alter the shortest path information for the remaining algorithm steps. As a consequence, modified heuristics must reevaluate the shortest paths between nodes when nodes and/or edges are eliminated. The authors also proposed an alternative heuristic, called *shortest path heuristic with iteration* (SPH-R). Construction of a tree begins with an arbitrary starting point, and an edge that is closest to the partial tree is added, one at a time. The shortest path heuristic is repeatedly applied to the network graph for different starting points. SPH-R terminates when it generates a solution.

Maximum Bandwidth Tree

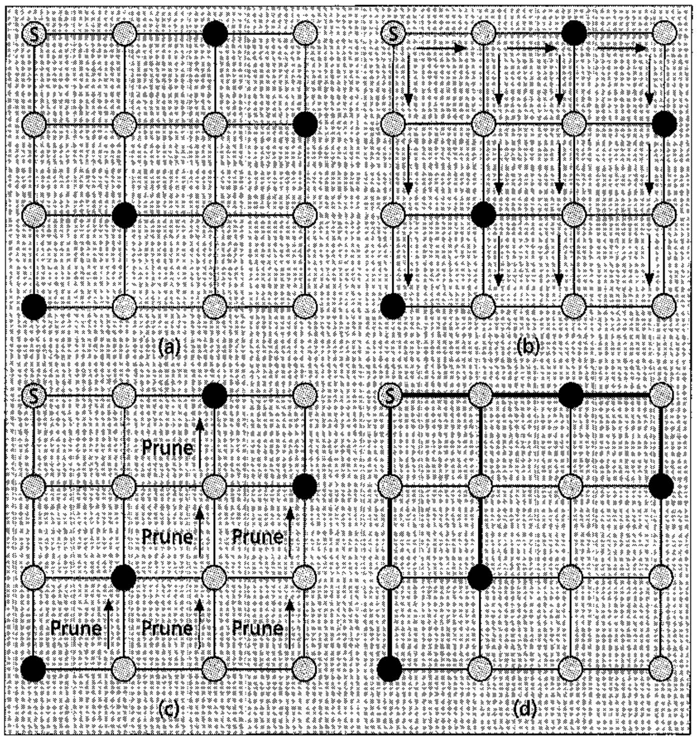
Shacham proposed a maximum bandwidth tree algorithm for distributing hierarchically encoded data [7]. It uses a Dijkstra-like algorithm to compute the maximum single-path bandwidth to all destinations. Their algorithm works as follows. First, they compute the maximum available bandwidth paths to all receivers from the source. The set of links connecting the nodes on the paths to the receivers form a maximum bandwidth tree by construction. Second, receivers are classified into different categories according to their receiving capabilities. A quality value is assigned for each layer of data. The satisfactory level of a receiver is measured by summing up the quality value over all intended layers that are received. The rate at which each receiver will receive is then determined to maximize the sum of the satisfactory level of all receivers. This optimization procedure gives, for each individual receiver, the intended rate at which it will receive from the source. The link bandwidth will then be allocated appropriately on the maximum bandwidth tree. The maximum bandwidth tree algorithm solves the link optimization problem.

Miscellaneous Trees

Rouskas *et al.* studied and proposed a heuristic algorithm to the problem of constructing source-based multicast trees to meet the delay and interreceiver delay jitter constraints [21]. Chen *et al.* proposed a distributed receiver-initiated probe-based multicast routing algorithm to construct a multicast routing tree with certain QoS requirements [22].

Tree Rearrangement in Response to Member Join/Leave — A multicast group member may join or leave a multicast session dynamically. It is thus important to ensure that member join/leave will not disrupt the ongoing multicast session, and the multicast tree after member join/leave will still remain near-optimal and/or satisfy the QoS requirements of all on-tree receivers. If a multicast tree is reconstructed each time a member joins or leaves, on-tree nodes may not switch to the new tree simultaneously, and a seamless transition may not be possible. One may handle dynamic member join/leave by incrementally changing the multicast tree. When a new member intends to join the distribution tree, a tree branch connects the new member to the nearest tree node. When a member leaves the multicast group, only the corresponding tree branch is torn down. This incremental change approach suffers because the quality of the multicast tree maintained may deteriorate over time in terms of, for example, total tree cost.

Several researchers addressed the multicast tree rearrangement issue, among which the *edge-bounded algorithm* (EBA) [27],



■ Figure 3. *The tree construction process in DVMRP; dark circles are group members: b) periodically datagrams are forwarded using RPM across the entire internetwork; c) a leaf router sends a prune message back toward the source if there are no group members on its directly attached leaf subnetwork; d) the resulting multicast tree.*

Bauer and Varma's algorithm [28], Narvaez's algorithm [29], and Sriram's algorithm [30] have received the most attention. The main idea is to define and monitor a certain damage index to the multicast tree as members join/leave, and trigger tree rearrangement when the index exceeds a certain threshold.

Multicast Routing Protocols

After a thorough treatment of multicast routing algorithms, we are now in a position to summarize and classify, based on how a tree is constructed, several existing multicast routing protocols into two categories: *source-based* and *core-based* (Table 3).

The Source-Based Multicast Tree Approach

A tree rooted at a source node is constructed and connected to every member in the multicast group. Data packets originating from the source node are sent to all the destination nodes via the links of a multicast tree. The Distance-Vector Multicast Routing Protocol (DVMRP) [37], Multicast Extensions to Open Shortest Path First Protocol (MOSPF) [38], Protocol Independent Multicast Dense Mode (PIM-DM) [39], and very recently Explicitly Requested Single-Source Multicast (EXPRESS) [40] fall in the category of source-based trees.

DVMRP — DVMRP constructs source-based multicast trees using the Reverse-Path Multicast (RPM) algorithm. In RPM, upon arrival of a datagram from a source node s, a router forwards the datagram on the other interfaces only if the interface ℓ on which the datagram arrives lies on the shortest path back to source s.

As shown in Fig. 3, DVMRP forwards datagrams based on the RPM paradigm, and periodically datagrams for a (source, group) pair are forwarded across the entire internetwork (Fig. 3b). A leaf router sends a `prune` message back toward the source if there are no group members on its directly attached leaf subnetwork (Fig. 3c). Each DVMRP router then updates the forwarding table accordingly. Periodically the prune state times out, and the process of forwarding datagrams across the entire internetwork and trimming tree branches based on `prune` messages received repeats. To reduce the join latency, if a router that previously sent a `prune` message for a (source, group) pair discovers new group members on its subnetwork, it sends a `graft` message to the group's upstream router, which then modifies the prune state accordingly. `Graft` messages may cascade back toward the source to graft the branch to the multicast tree. DVMRP operates independent of unicast routing. It maintains its own unicast information through exchange of distance vectors with multicast-capable neighbors. A DVMRP router relies on the receipt of "poison reverse" updates to maintain a list of dependent routers and to determine leaf routers.

MOSPF — In OSPF, each router within a routing domain keeps topological and state information of this domain. This is achieved through link-state advertisement (LSA) flooding. An MOSPF router makes use of this feature, uses IGMP [41] to monitor multicast group membership on directly attached subnetworks and floods group-membership LSA to all the other routers. An MOSPF router builds a shortest-path tree rooted at the source using Dijkstra's algorithm. After the tree is built, group membership information is used to prune those branches that do not lead to subnetworks with group members. The result is a pruned shortest-path tree rooted at the source. The MOSPF router then determines its position in the shortest-path tree and creates a forwarding table. The forwarding table is not periodically refreshed, but only changes when the network topology or group membership changes.

PIM-DM — PIM has two operation modes: dense mode and sparse mode. Dense mode refers to an environment where group members are relatively densely packed and bandwidth is plentiful. Sparse mode refers to an environment where group members are distributed across many regions of the network, and bandwidth is not necessarily widely available. PIM-DM is similar to DVMRP in that it employs the RPM algorithm and uses `graft` messages to add a previously pruned branch to the tree. PIM-DM is different from DVMRP in that it requires the presence of a unicast routing protocol to provide unicast routing information and to adapt to topology changes.

EXPRESS — Holbrook *et al.* [40] proposed an extension to IP multicast to support large-scale single-source applications. This extension is based on the *channel* model of multicast in which a multicast channel is identified by a tuple (S, E), where S is the source address and E a multicast address. Only source host S may send to (S, E). The channel model eliminates the necessity for global multicast address allocation.

The authors described a realization of the channel model called *explicitly requested single-source multicast* (or EXPRESS) on top of the IP multicast. It uses an EXPRESS count management protocol (ECMP) to both maintain the multicast tree and support source-directed counting. ECMP generalizes the join/leave function by using a count to count the number of subscribers in a subtree. A new subscriber sends a `subscriberID Count` message to the next hop on the shortest

Type	Protocol	Uni/bidirectional	Algorithm	Rely on unicast routing?(1)	Computation	Distribution tree	QoS techniques that can be used
Source-based	DVMRP	Unidirectional	RPM	No	On demand	Is periodically flushed and rebuilt	
	MOSPF	Unidirectional	Dijkstra's	Yes	On demand	Changes only in response to network topology and membership change	Guerin [36, 37]
	PIM-DM	Unidirectional	SPT(2)	Yes	On demand	Is soft-state based	Guerin [36, 37]
	EXPRESS	Unidirectional	SPT	Yes	On demand	Is soft-state based	Tyan [38, 39], Carlberg [40], Banarjea [41]
Core-based	PIM-SM	Unidirectional	SPT	Yes	On demand	Is soft-state based	Tyan [38, 39], Carlberg [40], Banerjea [41]
	CBT/SM	Bidirectional	SPT	Yes	On demand	Is periodically refreshed but does not change in response to network topology change	Tyan [38, 39], Carlberg [40], Banerjea [41]

(1) For routing information collection
(2) SPT: shortest path tree

Table 3. *A classification of multicast routing protocols.*

path to the channel source. The `subscriberID Count` message propagates hop by hop until it reaches the source or an on-tree router. A host unsubscribes by sending a zero `Count` message upstream. Count values kept at the routers are updated when a subscriber joins/leaves. When topology change causes a router to select a different upstream router for a channel, it sends a current `Count` message to the new upstream router and a zero `Count` message to the old upstream router, unsubscribing it there. A router selects either TCP or UDP mode for ECMP on each interface. As a result, ECMP can be deployed on an end system host that supports IP multicast without changing the host operating system. With TCP-based ECMP, it is not necessary to send a periodic refresh for long-lived channels. A single periodic per-connection keep-alive detects TCP failures. This allows ECMP to support a large number of channels, since only one message is required to initiate subscription and one to end it, and per-channel timers are eliminated. Multisource applications can be built on top of EXPRESS channels by using multiple channels, one per source, or allowing several sources to share a channel by relaying packets through the channel's source host.

The Core-Based Multicast Tree Approach

One node for each group is selected as the core (or termed a rendezvous point, RP, in [39, 42]) for the group. A tree rooted at the core is then constructed to span all the group members. The Core Based Tree (CBT) protocol [43], PIM Sparse Mode (PIM-SM)[1] [39, 42], and very recently Simple Multicast (SM) [44] are representatives of core-based trees.

PIM-SM -- In PIM-SM (Fig. 4), there is only a single active RP for each multicast group. A receiver that wishes to join a multicast group contacts (via IGMP `query/report` messages) its directly attached router. The local router then creates a forwarding cache for the (*, group) pair and explicitly joins the distribution tree by sending a unicast `PIM-join` message to the group's RP. An intermediate router forwards the `PIM-join` message and creates the (*, group) pair. When a source host first transmits a multicast packet to a group, the local router encapsulates the packet in a `PIM-register` packet and unicasts it to the RP. The primary RP then transmits a `PIM-join` message back to the source router. Upon receipt of the `PIM-join` message from the RP, the source router then ceases to encapsulate data packets in `PIM-registers` but forwards them in the native multicast format to the RP.

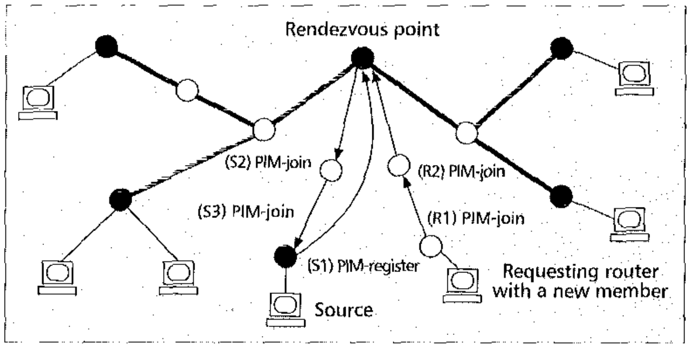
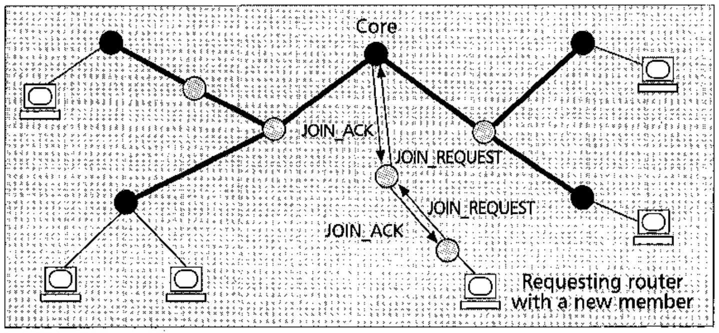


Figure 4. *The member join procedure in PIM-SM.*

In the steady state each router sends periodic PIM-join/prune messages, for each active PIM route entry, to capture state, topology, and membership changes. A `PIM-join/prune` message is also sent on an event-triggered basis each time a new route entry is established for some new source (note that some damping function may be applied, e.g., a short delay to allow for merging of new join information). `PIM-join/prune` messages do not elicit any form of explicit

[1] *PIM-SM, however, includes a mechanism to switch to a source-specific shortest-path tree when the data rate of a source exceeds some threshold.*



■ Figure 5. *The member join procedure in CBT.*

acknowledgment; routers recover from lost packets using the periodic refresh mechanism.

A PIM-SM router that originally receives datagrams from the shared tree may switch to a source-based tree if the data rate from the source exceeds a predefined threshold. The router (say router R) sends a `join` message toward the source. Upon receipt of the `join` message, source router s adds the interface (on which the `join` message arrives) to its (s, group) entry. After a router on the path from s to R receives datagrams directly from node s (rather than from RP), it sends a prune message to the RP so that the RP does not forward datagrams belonging to (s, group) henceforth.

CBT — Similar to PIM-SM, a host first expresses its interest in joining a group by contacting its local router, which then sends a `join-request` message to the next hop on the shortest path toward the group's core router (Fig. 5). The `join-request` sets up transient join state in the routers it traverses. The `join-request` message travels hop by hop toward the core until it reaches the core or an on-tree router, at which point a `join-acknowledgment` message is sent back along the reverse path. When a router receives a `join-acknowledgment` message, it updates its forwarding cache to reflect the fact that it now becomes an on-tree router, and forwards the `join-acknowledgment` message back to the requesting router.

"Tree maintenance" in CBT is achieved by having each downstream router periodically send a CBT "keepalive" message (i.e., `echo-request`) to its parent router on the tree. The receipt of a keepalive message over a valid child interface prompts a response (i.e., `echo-reply`). If no response is forthcoming before the corresponding timer expires, the router sends a `quit-notification` message upstream, and flushes all of its downstream branches by sending `flush-tree` messages, allowing them to individually rejoin if necessary. When a member leaves the group, if the local router to which the leaving member is attached does not have any other directly attached members or downstream on-tree routers, the router sends a `quit-notification` message to its parent router on the tree and deletes the corresponding forwarding cache.

During data transmission, data packets flow from any source to its parent and children, and the parent router forwards packets to all the children other than the source and to its parent until data packets reach the core. Data packets are then sent down all the other branches. To accommodate the situation in which a source is not on the multicast tree, the local router to which the sender host is attached encapsulates the data packet and unicasts it to the core, where it is decapsulates and disseminated over the tree.

SM — Perlman *et al.* [44] proposed a multicast routing protocol, called Simple Multicast, that extends CBT and works both within and between domains. SM [44] resembles CBT in terms of member join/leave, data transfer, and tree maintenance. The major difference between CBT and SM is how they resolve the multicast address allocation problem. SM identifies a group by the 8-byte combination of a core node C and the multicast address M. The identity of a multicast group is carried in the join-request messages and data messages. As a result, M no longer has to be unique across the Internet; it only has to be unique per core node. This eliminates the need for coordinated multicast address allocation across the Internet. EXPRESS uses a similar addressing concept in its channel model.

A Comparison of the Two Approaches — From the viewpoint of network management, the CBT approach offers more favorable scaling characteristics than the source-based approach by a factor of the number of active sources. A router does not have to maintain information about each source for each group and needs, instead, a single entry for each group. Besides, routers that are not on a multicast tree do not have to be involved in the group membership maintenance activities, such as sending `prune` messages to trim branches that do not lead to group members, as in DVMRP, or flooding group-membership LSAs, as in MOSPF.

The price core-based multicast routing has to pay, however, is that the resulting multicast tree may be suboptimal with respect to some source(s), resulting in increased delay. Also, a CBT may concentrate traffic from multiple sources on a few links that are part of the CBT. When the reservation model is not in the shared mode (e.g., the RSVP fixed filter reservation style for multiple senders), the network bandwidth of certain on-tree links may have been exhausted by group members that are already on-tree. If a group member whose shortest route to the core contains these on-tree links joins the tree, the QoS of either the existing on-tree members or the new member may not be met. Finally, mechanisms such as the bootstrap mechanism reported in [45] are needed to support core management, which encompasses selection, distribution, and dynamic placement of core routers.

Issues in Multicast Routing Protocols

In addition to the multicast routing algorithms used to construct distribution trees (which we treated at length earlier), there are several other issues one must consider to devise an operational, scalable multicast routing protocol.

Collection and Update of State Information — To provide input to the multicast routing algorithm used, each node in the network has to keep either global or partial network state. The form in which the state is kept may be exact, probabilistic [46, 47] or aggregate [48, 49]. The network state is collected by using either distance vector protocols (e.g., RIP [50]) or link state protocols (e.g., OSPF [51]). In the former protocols, each node exchanges the state information by sending distance vectors to its neighbors. A distance vector is indexed by the nodes in the network, and each entry consists of the distance of the shortest path to a node and the next hop on the shortest path to that node. In the latter protocols, each node exchanges the state information by flooding LSA messages to all the other nodes within a routing domain so that each node knows the network topology and the state of every link. Multicast protocols may or may not rely on an underlying unicast routing protocol to collect state information. For example, PIM relies on a unicast routing protocol to provide routing table information, while DVMRP maintains its own routing information through exchange of distance vectors with

DVMRP-capable neighbors. Because of nonnegligible network delay, asynchronous exchange of state among network nodes, the state kept at a node serves only as an approximation of the current network state.

Computation of Routes/Trees — In addition to the control message overhead incurred in tree/state maintenance, the operational cost of routing protocols also includes the computation overhead incurred to compute the route/multicast tree. Depending on *when* the computation is performed, one may classify the route/tree computation mechanisms into two categories: *on-demand* routing and *precomputed* routing. In on-demand routing, a route is computed whenever a connection request arrives. In precomputed routing, routes for connection requests are computed a priori and cached. Sometimes multiple alternative, disjoint routes may be cached for each destination. Upon arrival of a connection request, a route is selected either randomly or with respect to certain criteria. Compared to on-demand routing, the computational overhead for precomputed routing is relatively low, but the route provided may not reflect the current network state. To balance route quality and computational overhead, on-demand routing may be combined with precomputed routing.

State and Tree Maintenance — To be robust to message loss, many Internet protocols have adopted the *soft state* approach to tree/state maintenance: the state kept at each router periodically times out. For example, as discussed earlier, a DVMRP router periodically deletes its prune state, which causes the next datagram to flow across the entire internetwork and routers that are not interested in receiving the datagrams to send `prune` messages on the interface on which the datagram arrives. In most core-based multicast routing protocols (e.g., PIM-SM, CBT, SM), the soft state is periodically refreshed by certain state-refresh messages. A state is deleted if no matching refresh messages arrive before the expiration of its associated timer. The major difference between PIM-SM and CBT/SM is that although CBT/SM uses state-refresh messages to maintain states, it does not tear down on-tree routes if the underlying unicast route from a group member to the core has changed in adapting to load changes. That is, once a CBT tree is built, it never changes. In contrast, a PIM-SM tree adapts to underlying unicast route changes. To remedy this CBT deficiency, Tyan *et al.* [33, 34] proposed to include in CBT `echo-request/reply` messages the state information collected by the underlying unicast protocol. An on-tree CBT router is then able to know, upon receipt of such a message, whether or not the tree branch needs to change. If yes, the CBT router may simply flush the subtree below it and ask downstream group members to individually rejoin the multicast tree.

Scalability — As networks grow large and interconnect with other networks, the size of the routing tables and the amount of periodic update messages will continue to grow. This makes efficient update and storage of state information in a large network difficult. Without effective approaches to deal with the scalability problem, the processing and memory capabilities of routers will eventually be depleted as networks continue to grow. One approach to handling massive amounts of state update/storage and improving scalability is to construct a single tree shared by all the sources in a multicast group. This is exactly the main driving force for designing core-based multicast routing protocols. Another approach is to represent state information in a certain aggregate form [48, 49] and/or to have some form of hierarchical routing [52, 53]. More on the latter will be discussed next.

Issues in Providing QoS in Multicast Routing

As discussed earlier, QoS-driven unicast/multicast routing is an indispensable component in a QoS-centric network architecture. However, there are several challenging issues that must be solved before QoS-driven multicast routing can be deployed in real networks. In this section we present these issues and discuss some of the solution approaches.

Locating a QoS-Satisfying Route

The first issue in QoS-driven multicast routing is to locate a QoS-satisfying route. Discussion on this issue is differentiated between source-based trees and core-based trees.

Locating a QoS-Satisfying Route in Source-Based Trees — Guerin *et al.* proposed a QoS extension to OSPF, with the minimum bandwidth requirement as the QoS parameter [31]. Although the extension focuses on unicast flows, it can be extended with moderate modification to multicast flows as well. Each node in the network runs the algorithm either periodically or on demand, and computes paths from the current node to all possible destinations for all possible bandwidth values. The results are stored in a QoS routing table that is a $K \times H$ matrix, with K being the number of destinations and H being the maximum allowable number of hops for a path. The $(n; h)$ entry of the table is built during the hth iteration of the algorithm, and consists of two fields:

- *bw*: the maximum available bandwidth, on a path of at most h hops between the node that runs the algorithm (which we term the computing node) and destination node n.
- Neighbor: the next hop on the h or less hops path to destination node n, whose available bandwidth is *bw*

Let the available bandwidth on the link from node n to node m be denoted $b(n, m)$. The routing table is first initialized with all *bw* fields set to 0 and neighbor fields cleared. Next, the entries in the first column (which correspond to the one-hop paths) are modified as follows: the *bw* field is set to the value of the available bandwidth on the direct link (if any) from the current node. The neighbor field is set to the neighbor of the computing node.

The algorithm iterates for at most H iterations. At the hth iteration, the algorithm first copies column $h - 1$ into column h. In addition, the algorithm keeps a list of entries that changed their *bw* value during the $(h - 1)$th iteration. The algorithm then looks at each link (n, m) where n is a node whose bw value in the $(n; h)$ entry changed in the previous iteration, and checks the maximum available bandwidth on an (at most) h-hop path to node m whose final hop is that link. This amounts to taking the minimum between the *bw* field in the $(n; h - 1)$ entry and the link metric value $b(n; m)$ kept in the topology database. If this value is higher than the present value of the bw field in the $(m; h)$ entry, a better path with larger bandwidth has been found for destination node m and with at most h hops. The *bw* and neighbor fields of the $(m; h)$ entry are then updated to reflect this new value. This records the next hop from the computing node on the best path identified thus far for destination node m and with h or less hops.

In conjunction with the QoS extension, Guerin *et al.* [32, 54, 55] have also proposed a QoS path management mechanism that aims at allowing management, through the RSVP protocol, of paths selected by their proposed QoS-driven OSPF, and have extended the interface between RSVP and routing to support a broader range of routing mechanisms than allowed by the current recommended interfaces.

Locating a QoS-Satisfying Route in Core-Based Trees — In CBT approaches, the shortest path from a requesting router to the core may not be the best QoS route, due to the fact that traffic from multiple sources may concentrate on a few on-tree links near the core and exhaust the network resources on them. To provide QoS in CBT routing, it is imperative for a requesting router to locate, based on certain QoS metrics, a path to an on-tree router. Once a requesting router identifies such a path, it initiates the member join process by sending a request on the selected path toward the on-tree router. Several approaches have been proposed, which we summarize below.

Local Search with Bidding — Complementary to CBT, Carlberg *et al.* proposed the Yet Another Multicast (YAM) protocol [35, 56] in which a new router which intends to join a multicast tree does a `bid-order` broadcast with limited scope using the time-to-live (TTL) field. On-tree routers that receive the broadcast message become candidate routers and return `bid` messages. The `bid` messages contain static router information (e.g., link capacity and propagation delay), based on which the new router locates the best on-tree router.

There is one potential problem: if the value of TTL in the `bid-order` broadcast messages is set too small, the messages may reach no qualified on-tree routers and all the `bid` messages returned (if any) do not identify qualified routes. To solve this problem, an expanded ring search is used by increasing the TTL each time until either a qualified path is located or an upper bound of the TTL value is reached. The determination of an appropriate TTL bound is a trade-off among the message overhead, setup time, and probability of successfully locating a path.

Multicast Tree Search — Banerjea *et al.* extended YAM and proposed the QoSMIC protocol in which both local search with bidding and multicast tree search are used to locate routes [36]. Specifically, a joining router broadcast `bid-order` messages and at the same time sends a `multicast-join` message to a manager router (which may or may not be the core router). If the manager router has sufficient knowledge of the tree and the network topology, it sends `bid-order` messages to a set of selected candidate on-tree routers; otherwise, the manager router joins the multicast group and multicasts a `bid-order` message on the tree. On-tree routers which receive `bid-order` messages from either the management router or the requesting router then respond by unicasting bid messages to the joining router. `bid` messages collect *dynamic* QoS information (e.g., available bandwidth and current link delay) on their way to the requesting router. The multicast tree search approach may incur, in the worst case, message overhead on the order of the multicast group size. This makes the protocol not scalable to large multicast groups.

Both protocols are QoS-sensitive, but do not address how *end-to-end* QoS is achieved; they only focus on finding the best route (or the best on-tree router) along which (at which) a new member joins the tree.

Maintaining Desired QoS on a Multicast Tree

An approach that complements YAM and QoSMIC and provides end-to-end QoS is to conduct admission control when a join request arrives at the core (RP) or an on-tree router. Tyan *et al.* [33, 34] proposed an extension to the core-based multicast routing protocols to facilitate the deployment of additive, multiplicative, and concave QoS. In their extension, each join request message carries, in addition to the interface information, the QoS parameters of interest. When a join request reaches the core or an on-tree router, the core/on-tree router conducts a set of *eligibility* tests to verify whether or not a new member can join a multicast tree at adequate QoS, while not violating the existing QoS provided to the other on-tree members. Only after the join request survives the eligibility tests will an acknowledgment message be sent back.

Tyan *et al.* considered the many-to-many multicast paradigm in which each member in the multicast group can be a source in addition to being a receiver. To capture the main idea, we take the end-to-end delay bound as an example. An upper bound D is imposed on the end-to-end delay along any path from a source to any receiver in a multicast group. Each on-tree router u keeps the following states for each downstream interface i:

- $d_i^{max}(u,*)$: the maximum delay among the on-tree paths from router u to the downstream on-tree group members reachable on interface i (recall that downstream is defined with respect to the core)
- $d_i^{max}(*, u)$: the maximum delay among the on-tree paths from all the downstream on-tree source group members to router u reachable on interface i

They also define the maximum outgoing/incoming delay between router u and all its downstream on-tree group members *except those reachable on interface* ℓ as

$$d^{\max}_{I\setminus\{\ell\}}(u,*) \triangleq \max_{i\in I\setminus\{\ell\}} d_i^{\max}(u,*) \text{ and}$$

$$d^{\max}_{I\setminus\{\ell\}}(*,u) \triangleq \max_{i\in I\setminus\{\ell\}} d_i^{\max}(*,u),$$

where I is the set of downstream interfaces of router u. Let $T_s(u)$ denote the subtree rooted at router u; then each on-tree router only keeps the state information for $T_s(u)$.

When a join request message from a joining router v arrives at an on-tree router u on interface i, router u checks if

$$d(u, v) + d^{max}_{I\setminus\{i\}}(*, u) \le D. \tag{4}$$

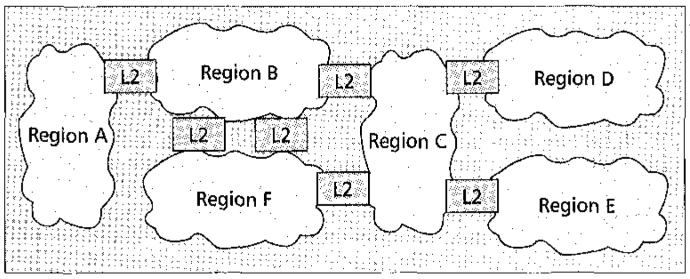
If the joining member v is also a source, router u further checks if

$$d(v, u) + d^{max}_{I\setminus\{i\}}(u, *) \le D. \tag{5}$$

Both parameters $d^{max}_{I\setminus\{i\}}(u, *)$ and $d^{max}_{I\setminus\{i\}}(*, u)$ are obtained from the states kept at router u. Both parameters $d(v, u)$ and $d(u, v)$ can be carried in the join request message and updated as the message travels from router v to router u. If Eq. 4 holds, it implies the QoS requirement of the new member v is fulfilled by all the source group members in $T_s(u)$. Similarly, if Eq. 5 holds, it implies that the QoS requirements for the group members in $T_s(u)$ are not violated due to the join of the new source member v.

If Eq. 4 (or 5) does not hold, the join request is immediately rejected. Otherwise, router u forwards upstream to its parent router w the join request with updated cumulative delay information ($d(w, u) + d(u, v)$, and $d(v, u) + d(u, w)$ if router v is also a source). Upon receipt of the join request on interface i, router w conducts the eligibility test (i.e., Eqs. 4–5, except that u is replaced by w in the expressions) to verify that the QoS is met on the subtree $T_s(w)$. If the eligibility tests succeed, router w forwards the join request upstream with updated cumulative delay information; otherwise, it sends a rejection reply message downstream on the interface on which the join request arrives.

The process repeats until the join request is either rejected at some upstream on-tree router or forwarded to, and approved by, the core, whichever occurs first. In the latter case, the core sends back a join acknowledgment message along the reverse on-tree path, and each on-tree router w on the path between router u and the core updates its state. Finally, router u sends back a join-acknowledgment message.



■ Figure 6. *Hierarchical DVMRP.*

Following the same line of development, Tyan *et al.* also modify the member leave procedure to notify the on-tree routers between the on-tree router at which the tree branch is torn down and the core of the (possible) need to update their states. Based on the soft state concept, they also devise a state update/refresh procedure which can be readily integrated with the tree maintenance mechanism that already exists in most receiver-initiated multicast routing protocols (e.g., *echo-request* and *echo-reply* in CBT). Their QoS extension can be applied to multicast routing protocols with explicit receiver-initiated member join/leave procedures and soft state update/refresh procedures (e.g., PIM-SM, CBT, and SM).

Handling Heterogeneous QoS

In a heterogeneous environment, the perceived quality may vary among users. For example, in the context of video distribution, video quality perceived depends on the bandwidth/buffer capabilities of receivers. Attempting to unify the perceived quality results in a dilemma: if all users who wish to participate are allowed to be in the session regardless of their constraints, the quality of the session is driven by the quality of the least capable receiver. On the other hand, if the more capable users insist on a certain minimum quality, receivers that cannot participate at that level may have to be excluded.

To accommodate different users' requirements, two types of approaches have been proposed:

- A multicast source may vary the transmission rate by using rate-adaptive coding [57] or combining a layered compression algorithm with a layered transmission scheme [7, 58–61]. In the latter approach, multimedia data are encoded into a number of layers that can be incrementally combined to provide progressive refinement. Lower layers encode coarse information, while high layers encode details. The different layers of the data are striped across multicast groups, and receivers add/drop layers by joining/leaving the corresponding multicast groups. In this case, QoS is not directly supported by multicast routing protocols, but instead by the end systems.
- A multicast routing protocol may identify a multicast tree that provides users of heterogeneous capabilities and QoS requirements with different levels of QoS. A receiver-initiated multicast routing protocol (e.g., PIM-SM, CBT, SM, and EXPRESS) may use this approach, since receivers may specify their capabilities in the join request, and the protocol could take these requirements into account in the routing decision. Tyan *et al.* extended their work to support heterogeneous QoS among receivers [33, 34]. For example, if D_v denotes the delay bound imposed by a receiver router v, the eligibility test (Eqs. 4–5) is modified as

$$d(u, v) + d^{max}_{I\backslash\{i\}}(*, u) \le D_v, \text{ and} \tag{6}$$

$$d(v, u) \le D_w - d(u, w), \forall w \in T_s(u). \tag{7}$$

Note that Eq. 7 can be rewritten as $d(v, u) \le \min_{w \in T_s}(u)\, D_w - d(u, w)$. The per-interface state kept at each on-tree router u is:

- $d_i^{max}(*, u)$
- The minimum laxity defined as the minimum difference between D_w and $d(u, w)$ among all downstream on-tree group members w reachable on interface i, that is, min $w \in T_s(u)$, reachable on interface i $(D_w - d(u, w))$.

Interdomain and Hierarchical Routing

The growing sizes of networks and multicast groups give rise to the interdomain routing and scalability problem. To deal with the former problem, multicast source distribution protocol (MSDP) [62] is under development by the IETF. Also, the IETF recently created a working group on Border Gateway Multicast Protocol (BGMP) [63] to solve the interdomain multicast routing problem in the long run. For the latter (scalability) problem, one natural solution is to have some form of hierarchical routing: nodes are organized into clusters or areas, and each of these areas are single addressable entities from the viewpoint of higher-level areas. The topology within an area is transparent to the nodes outside the area. Each node only needs to know the explicit details about routing packets to destinations within its own area, but knows nothing about the detailed topological structure of any of the other areas. The protocol running between the individual areas (domains) maintains information about the interconnection of the domains, but not about the internal topology of each domain.

The first operational hierarchical routing protocol is hierarchical DVMRP, proposed by Thyagarajan and Deering [52]. The network is divided into a number of individual routing domains, as in the hierarchical network model (Fig. 6). Routers internal to a domain are termed L1 routers, and execute their own instance of the multicast routing protocol. Each region is required to have at least one "boundary router" (termed an L2 router) that provides interregional connectivity. Another protocol (e.g., an instance of DVMRP) is used as the L2 protocol, and is responsible for routing between the individual domains. With DVMRP as the L2 protocol, an interdomain multicast delivery tree is constructed based on the (domain_ID, group) pair.

In spite of the advantage of reducing communication, computation, and storage overheads, hierarchical routing imposes several difficulties in QoS-driven constraint-based routing. First, it introduces imprecision in the representation of state information (which we discuss next). Second, it may be difficult for a node to locate a QoS-satisfying route because of its partial, incomplete view of network topology and state. Efficient schemes must be devised to allow higher-level entities appropriately aggregate state information and construct a QoS-satisfying route/tree in a hierarchical fashion. For example, a node may construct a skeleton route/tree, and as control messages traverse the route/tree, the lower-level routing details are then filled in by intermediate higher-level nodes as is done in hierarchical routing in asynchronous transfer mode (ATM) private network to network interface (PNNI) [53]. There are a few pieces of work in the hierarchical multicast routing. Most notable are Murthy and Garcia-Luna-Aceves's hierarchical distance-vector routing algorithm, called *hierarchical information path-based routing* (HIPR), that provides a distributed implementation of Dijkstra's shortest path algorithm running over a hierarchical graph [64], and Behrens and Garcia-Luna-Aceves's Area-based hierarchical Link-Vector Algorithm (ALVA), where routers propagate incremental information only about those links that they actually use to reach any destination (thus the name *link-vector*), and all routers keep a partial topology [65]. The shortest path algorithm is then used to compute the multicast tree based on that partial topology.

QoS-Driven Multicast Routing with Imprecise State Information

To provide input to the multicast routing algorithm used, each node in the network has to keep either local or partial network state. Several state update policies have been proposed to determine when a state update should be triggered. As summarized in [66], the most commonly used policies are:

- Relative change or threshold-based triggers (i.e., a state update is triggered when the amount of change in the state variable exceeds a threshold)
- Absolute change or class-based triggers (i.e., a state update is triggered when the state changes from one class to another)
- Timer-based triggers (i.e., a state update is periodically triggered)

To control the protocol overhead and to limit it to a tolerable level, large clamp-down timers are used to limit the rate of updates. The accuracy of network state is also affected by, for example, the scope of an update message, and the types of value advertised (exact state values or quantized values). There is a fundamental trade-off between the accuracy of state information and the protocol message overhead. Moreover, in large interconnected networks, the growth in the state information makes it practically impossible to maintain accurate knowledge about all nodes and links. Instead, the state information is usually aggregated in a certain hierarchical manner, and the aggregation process inherently decreases the information accuracy and introduces imprecision. The imprecise state information kept at each node imposes difficulty in QoS provisioning. Most of the work in QoS routing in the presence of imprecise state information is still in the theoretical development stage.

Guerin and Orda investigated the problem of QoS routing when the state information is inaccurate and expressed in some probabilistic manner [49]. Their objective was to identify a path that is mostly likely to satisfy the delay requirement, which they achieved by decomposing the end-to-end requirement into local delay constraints and deriving tractable, near-optimal solutions for a certain class of probability distributions. Orda also considered the same problem, but in the context of networks with rate-based schedulers (i.e., networks that employs fair queuing scheduling disciplines for link sharing) [67]. Chen *et al.* considered a simplified probability model for link parameters (i.e., link parameters are allowed to distribute uniformly over an interval) [46]. They then proposed a distributed ticket-based probing routing algorithm.

Interaction between QoS-Driven Routing and Resource Reservation

As depicted in Fig. 1, QoS-driven constraint-based routing and resource management are closely related. However, there are divided views on whether constraint-based routing and resource reservation should be integrated or separated. The integrated services model separates routing with resource reservation, and uses RSVP for receiver-initiated resource reservations for unicast/multicast data flows. By separating routing and resource reservation, the task of network management is eased at the expense of the route/tree located by routing possibly not being QoS-satisfying. Several researchers, on the other hand, proposed schemes to combine routing with resource reservation, with the argument that it is more likely to locate QoS-satisfying routes/trees at reduced connection setup latency [68–70]. A number of researchers studied the problem of resource reservation for multicast routing. Firoiu and Towsley proposed to decompose the problem into three subproblems [71]:

- Partitioning of end-to-end QoS requirements into local QoS requirements
- Mapping of local QoS requirements into resource requirements
- Reclaiming of resources allocated in excess

Lorenz *et al.* generalized Firoiu and Towsley's work and studied the problem of how to partition an end-to-end QoS requirement into local requirements, such that the overall cost of the multicast tree is minimized [72]. Instead of distributing QoS requirements along a path evenly or proportionally as in [71], Kodialam *et al.* proposed to associate each receiver with a fixed resource budget and directly distribute the resource budget of a receiver along its path from the source such that an objective function is optimized [73]; for example, the total delay along the paths from the source to all the receivers is minimized subject to the budget constraints at each receiver.

Summary and Future Direction

Multicast routing and its QoS-driven extension are indispensable components in a QoS-centric network architecture. Its main objective is to construct a multicast tree that optimizes a certain objective function (e.g., making effective use of network resources) with respect to performance-related constraints (e.g., end-to-end delay bound, interreceiver delay jitter bound, minimum bandwidth available, and maximum packet loss probability).

In this article we provide an overview of multicast routing algorithms, protocols, and their QoS extension. In particular, we classify multicast routing problems into 12 categories according to their objective functions and QoS constraints, and present routing algorithms in the literature in each problem class. We also categorize existing multicast routing protocols, and outline the issues and challenges in providing QoS in multicast routing. These reported research efforts, however, represent only initial attempts to provide a comprehensive, practical multicast routing framework. Many other issues need to be resolved before QoS-driven multicast protocols and services can be deployed on large-scale networks. These issues include:

- Empirical performance study of QoS-driven multicast routing in large-scale real networks to provide some insights into the trade-off between the design complexity of QoS-driven protocols and the resulting performance improvement. The work reported in [74] may be the first effort in this direction. The focus of this work was assessing the cost and feasibility of QoS routing in IP networks. The initial results show that the two major components of QoS routing costs are processing cost and protocol overhead, and that the costs are within the limits of modern technology.
- Further investigation in exploiting research results of hierarchical routing with imprecise, aggregated state information in QoS-driven multicast routing protocols. We have surveyed some theoretical work that can provide starting points for this direction of research. A continued effort should be made to incorporate these theoretical research results into multicast routing protocols, to study the scalability issue (in terms of protocol overheads), and to allow QoS-driven routing across domains, while having the minimum possible impact on the existing protocol specification.
- Revisiting the issue of whether to reserve resources in the course of multicast routing. Separating routing and resource reservation has been one of the design objectives of RSVP. However, when it comes to constraint-based routing, the status of resource usage and availability has to be taken into account in the routing decision anyway. This leads to the question of whether it is actually more efficient (in terms of reducing redundant functionalities and setup

latency) to marry routing and resource reservation. More investigation should be made to quantify the advantage of such a marriage. If the results do not justify it, studies should be performed to define the extent to which constraint-based routing should take into account resource usage in its decision and what level of QoS assurance (soft or deterministic) it can provide.

- Consideration of other multicast-related issues. *After* the multicast tree is constructed and during the data transmission phase, we have to consider two other QoS-related issues: reliable multicast and multicast congestion control. The former is concerned with retransmission of lost packets to group members in a multicast group, with the objectives of avoiding NACK implosion and duplicate replies, reducing recovery latency, achieving recovery isolation, and being adaptive to dynamic membership changes. The most notable work is reported in [75–80]. The latter ensures that multicast applications, when they are deployed on the Internet, respond to network congestion in a TCP-friendly manner in order to coexist with TCP flows, which constitute the majority of Internet traffic [81]. The multicast congestion control mechanism designed should be TCP-friendly, scalable, require as little router support as possible, and judiciously detect network congestion and respond accordingly, based on (perhaps aggregated) acknowledgments from receivers. The most notable work in this category is reported in [82–89]. Although not directly related to routing, these two issues should be considered, with multicast routing, an integral part of a QoS-centric multicast framework. It is also interesting to look into multicast routing, traffic engineering, and multiprotocol label switching to provide, for example, differentiated multicast services.

The above list represents a set of challenging problems. However, the need for multicast, and QoS-driven multicast in particular, in large-scale networks such as the Internet is expected to intensify in the near future. Efficient, scalable solutions to the above issues and issues discussed in the body of the article are therefore important.

References

[1] Integrated Services Working Group, 1998.
[2] R. Braden *et al.*, "Resource Reservation Protocol (RSVP) — Version 1 Functional Specification," RFC 2205, 1997.
[3] D. D. Clark and J. Wroclawski, "An Approach to Service Allocation in the Internet," draft-clark-diff-svc-alloc-00.txt, 1997.
[4] V. Jacobson, "Differentiated Services Architecture," talk to the Int-Serv WG at the Munich IETF, Aug. 1997.
[5] K. Nichols, V. Jacobson, and L. Zhang, "A Two-Bit Differentiated Services Architecture for the Internet," Internet draft, Nov. 1997, work in progress.
[6] Z. Wang and J. Crowcroft, "QoS Routing for Supporting Resource Reservation," *IEEE JSAC*, Sept. 1996.
[7] N. Shacham, "Multipoint Communication by Hierarchically Encoded Data," *Proc. IEEE INFOCOM '92*, May 1992, pp. 2107–14.
[8] M. R. Carey and D. S. Johnson, *Computers and Intractability*, New York: Freeman, 1979.
[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, 1997.
[10] R. Gallager, P. Humblet, and P. Spira, "A Distributed Algorithm for Minimum-Weight spanning trees," *ACM Trans. Programming Lang. and Sys.*, Jan. 1983, pp. 66–77.
[11] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica*, 1981, pp. 141–45.
[12] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Math. Japonica*, 1980, pp. 573–77.
[13] F. Bauer and A. Varma, "Distributed Algorithms for Multicast Path Setup in Data Networks," Comp. Eng. Dept., UC Santa Cruz, UCSC-CRL-95-10, Aug. 1995.
[14] N. F. Maxemchuk, "Video Distribution on Multicast Networks," *IEEE JSAC*, Apr. 1997, pp. 357–72.
[15] Q. Zhu, M. Parsa, and J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrained Minimal-Cost Multicasting," *Proc. IEEE INFOCOM '95*, 1995, pp. 377–84.
[16] V. P. Kompella, J. C. Pasquale, and G. C. Polyzo, "Multicast Routing for Multimedia Communication," *IEEE/ACM Trans. Net.*, 1993, pp. 286–92.
[17] B. K. Haberman and G. Rouskas, "Cost, Delay, and Delay Variation Conscious Multicast Routing," Tech. rep. TR-97-03, NC State Univ., 1997.
[18] V. P. Kompella, J. C. Pasquale, and G. C. Polyzo, "Two Distributed Algorithms for Multicasting Multimedia Information," *Proc. ICCCN '93*, 1993, pp. 343–49.
[19] X. Jia, "A Distributed Algorithm of Delay-Bounded Multicast Routing for Multimedia Applications in Wide Area Networks, *IEEE/ACM Trans. Net.*, Dec. 1998, pp. 828–37.
[20] F. Bauer and A. Verma, "Degree-Constrained Multicasting in Point-to-Point Networks," *Proc. IEEE INFOCOM '95*, 1995.
[21] R. N. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," *IEEE JSAC*, Apr. 1997, pp. 346–56.
[22] S. Chen and K. Nahrstedt, "Distributed QoS Routing in High-Speed Networks Based on Selective Probing," Tech. rep., CSD, UIUC, 1998.
[23] P. Winter, "Steiner Problem in Networks: A Survey," *Networks*, 1987, pp. 129–67.
[24] F. K. Hwang, Steiner Tree Problems, *Networks*, 1992, pp. 55–89.
[25] F. Bauer, "Multicast Routing in Point-to-Point Networks Under Constraints," Ph.D. dissertation, UC Santa Cruz, 1996.
[26] H. F. Salama, "Multicast Routing for Real-Time Communication on High Speed Networks," Ph.D. dissertation, NC State Univ., Raleigh, 1996.
[27] M. Imase and B. Waxman, "Dynamic Steiner Tree Problem," *SIAM J. Disc. Math*, Aug. 1991, pp. 369–84.
[28] F. Bauer and A. Varma, "ARIES: A Rearrangeable Inexpensive Edge-Based On-Line Steiner Algorithm," *IEEE JSAC*, Apr. 1997, pp 382–97.
[29] P. Narvaez, K. Siu, and H. Tzeng, "New Dynamic SPT Algorithm Based on a Ball-and-String Model," *IEEE INFOCOM '99*, New York, Mar. 1999.
[30] R. Sriram, G. Manimaran, and C. Murthy, "A Rearrangeable Algorithm for the Construction of Delay-Constrained Dynamic Multicast Trees," *IEEE INFOCOM '99*, New York, Mar. 1999.
[31] R. Guerin *et al.*, "QoS Routing Mechanism and OSPF Extensions," Internet draft, 1998.
[32] R. Guerin *et al.*, "QoS Routing Mechanism and OSPF Extensions," Internet draft, Mar. 1997.
[33] H.-Y. Tyan, C.-J. Hou, and B. Wang, "On Providing Quality-of-Service Control for Core-Based Multicast Routing," *IEEE Proc. Int'l Conf. Dist. Comp. Sys.*, June 1999.
[34] H.-Y. Tyan, C.-J. Hou, and B. Wang, "QoS Extension to the Core-Base Tree Protocol," *Proc. NOSSDAV '99*, June 1999.
[35] K. Carlberg and J. Crowcroft, "Yet Another Multicast (YAM) Routing Protocol: Specification, Version 1," unpublished manuscript, 1997.
[36] A. Banerjea, M. Faloutsos, and R. Pankaj, "Designing QoSMIC: A Quality of Service Sensitive Multicast Internet Protocol," Internet draft, 1998."
[37] T. Pusateri, "Distance Vector Routing Protocol," draft-ietf-idmr-dvmrp-v3-07, 1998.
[38] J. Moy, "Multicast Extension to OSPF," Internet draft, 1998.
[39] D. Estrin *et al.*," Protocol Independent Multicast (PIM) Sparse Mode/Dense Mode," Internet draft, 1996.
[40] H. W. Holbrook and D. R. Cheriton, "IP Multicast Channels: EXPRESS Support for Large-Scale Single-Source Applications," *ACM SIGCOMM '99*, 1999.
[41] "Internet Group Management Protocol, version 2 (IGMPv2)," Internet draft, 1996.
[42] S. Deering *et al.*, Protocol Independent Multicast-Sparse Mode (PIM-SM): Motivation and Architecture," Internet draft, 1998.
[43] A. Ballardie, B. Cain, and Z. Zhang, "Core Based Trees (CBT Version 3) Multicast Routing," Internet draft, 1998.
[44] R. Perlman *et al.*, "Simple Multicast: A Design For Simple, Low-Overhead Multicast," draft-perlman-simple-multicast-01.txt, 1998.
[45] D. Estrin *et al.*, "A Dynamic Bootstrap Mechanism for Rendezvous-Based Multicast Routing," *Proc. INFOCOM '99*, Mar. 1999.
[46] S. Chen and K. Nahrstedt, "Distributed QoS Routing with Imprecise State Information," *Int'l. Conf. Comp., Commun. and Networks (ICCCN '98)*, Oct. 1998.
[47] D. H. Lorenz and A. Orda, "QoS Routing in Networks with Uncertain Parameters," *Proc. IEEE INFOCOM '98*, 1998.
[48] W. C. Lee, "Spanning Tree Method for Link State Aggregation in Large Communication Networks," *Proc. IEEE INFOCOM '95*, 1995.
[49] R. Guerin and A. Orda, "QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms," *Proc. IEEE INFOCOM '97*, Apr. 1997.
[50] G. Malkin, "RIP Version 2: Carrying Additional Information," RFC 1723, 1994.
[51] J. Moy, "OSPF Version 2," Networking WG Internet draft, 1992.
[52] A. S. Thyagarajan and S. Deering, "Hierarchical Distance-Vector Multicast Routing," *Proc. ACM SIGCOMM '95*, 1995.
[53] ATM Forum, "Private Network Network Interface (PNNI) Specification v1.0," Mar. 1996.
[54] R. Guerin, S. Kamat, and S. Herzog, "QoS Path Management with RSVP," Internet draft, Mar. 1997.
[55] R. Guerin, S. Kamat, and E. Rosen, "Extended RSVP-Routing Interface," Internet draft, Mar. 1997.
[56] K. Carlberg and J. Crowcroft, "Building Shared Trees Using a One-to-Many Joining Mechanism," *ACM Comp. Commun. Rev.*, Jan. 1997, pp. 5–11.
[57] J. C. Bolot, T. Turletti, and I. Wakeman, "Scalable Feedback Control for Multicast Video Distribution in the Iternet," *Proc. ACM SIGCOMM '94*, Sept. 1994.
[58] D. Taubman and A. Zakhor, "Multi-Rate 3-D Subband Coding of Video," *IEEE Trans. Image Proc.*, Sept. 1994, pp. 572–88.
[59] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-Driven Layered Multicast," *Proc. ACM SIGCOMM '96*, 1996.

[60] X. Li, S. Paul, and M. H. Ammar, "Layered Video Multicast with Retransmission (LVMR): Evaluation of Hierarchical Rate Control," *Proc. IEEE INFOCOM '98*, Mar. 1998.
[61] X. Li, M. H. Ammar, and S. Paul, "Video Multicast over the Internet," *IEEE Network*, Mar./Apr. 1999, pp. 46–60.
[62] D. Farinacci *et al.*, "Multicast Source Discovery Protocol (MSDP)," Internet draft, draft-ietf-msdp-spec-01.txt, Mar. 1999.
[63] IETF BGMP WG, WG mailing list, ftp://catarina.usc.edu/pub/bgmp/mail-archive/, Oct. 1999.
[64] S. Murthy and J. J. Garcia-Luna-Aceves, "Loop-free Internet Routing Using Hierarchical Routing Trees," *Proc. IEEE INFOCOM '97*, 1997.
[65] J. Behrens and J. J. Carcia-Luna-Aceves, "Hierarchical Routing Using Link Vectors," *Proc. IEEE INFOCOM '98*, Mar. 98.
[66] G. Apostolopouloset *al.*, "Quality of Service Based Routing: A Performance Perspective," *Proc. ACM SIGCOMM '98*, 1998.
[67] A. Orda, "Routing with End to End QoS Guarantees in Broadband Networks," *IEEE INFOCOM '98*, Apr. 1998.
[68] C. J. Hou, "Routing Virtual Circuits with Timing Requirements in Virtual Path Based ATM Networks," *Proc. IEEE INFOCOM '96*, Mar. 1996, pp. 320–28.
[69] I. Cidon, R. Rom, and Y. Shavitt, "Multi-Path Routing Combined with Resource Reservation," *Proc. IEEE INFOCOM '97*, 1997.
[70] N. Rao and S. G. Batsell, "QoS Routing Via Multiple Paths Using Bandwidth Reservation," *Proc. IEEE INFOCOM '98*, 1998.
[71] V. Firoiu and D. Towsley, "Call Admission and Resource Reservation for Multicast Sessions," *Proc. INFOCOM '96*, 1996.
[72] D. H. Lorenz and A. Orda, "Optimal Partition of QoS Requirements on Unicast Paths and Multicast Trees," *Proc. IEEE INFOCOM '99*, Mar. 1999.
[73] M. Kodialam and S. H. Low, "Resource Allocation in a Multicast Tree," *Proc. IEEE INFOCOM '99*, Mar. 1999.
[74] G. Apostolopoulos, R. Guerin, and S. Kamat, "Implementation and Performance Measurements of QoS Routing Extensions to OSPF," *Proc. INFOCOM '99*, Mar. 1999.
[75] A. Costello and S. McCanne, "Search Party: An Approach to Reliable Multicast with Local Recovery," *Proc. IEEE INFOCOM '99*, Mar. 1999.
[76] S. Floyd *et al.*, "A Reliable Multicast Framework for Lightweight Sessions and Application Level Framing," *Proc. ACM SIGCOMM '95*, Oct. 1995, pp. 342–56.
[77] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton, "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," *Proc. ACM SIGCOMM '95*, Oct. 1995, pp. 328341.
[78] L.-W. Lehman, S. J. Garland, and D. L. Tennenhouse, "Active Reliable Multicast," *Proc. IEEE INFOCOM '98*, Mar. 1998.
[79] J. Lin *et al.*, "A Reliable Multicast Transport Protocol(RMTP)," *IEEE JSAC*, Apr. 1997.
[80] C. Papadopoulos, G. Parulkar, and G. Varghese, "An Error Control Scheme for Large-Scale Multicast Applications," *Proc. IEEE INFOCOM '98*, Mar. 1998.
[81] J. Mahdavi and S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control," Note in the end2end-internet mailing list, Jan. 1997.
[82] S. J. Golestani, "Fundamental Observations on Multicast Congestion Control in the Internet," *Proc. IEEE INFOCOM '99*, Mar. 1999.
[83] S. J. Golestani and S. Bhattacharyya, "A Class of End-to-End Congestion Control Algorithms for the Internet," *Proc. IEEE ICNP '98*, Oct. 1998.
[84] K.-W. Lee, S. Ha, and V. Bharghavan, "IRMA: A Reliable Multicast Architecture for the Internet," *Proc. IEEE INFOCOM '99*, Mar. 1999.
[85] J. Padhye *et al.*, "A TCP-Friendly Rate Adjustment Protocol for Continuous Media Flows Over Best Effort Network," *Proc. NOSSDAV '99*, June 1999.
[86] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-To-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet," *Proc. IEEE INFOCOM '99*, Mar. 1999.
[87] I. Rhee, N. Ballaguru, and G. N. Rouskas, "MTCP: Scalable TCP-Like Congestion Control for Reliable Multicast," *Proc. IEEE INFOCOM '99*, Mar. 1999.
[88] D. Sisalem and H. Schulzrinne, "The Loss-Delay Adjustment Algorithm: A TCP-Friendly Adaptation Scheme," *Proc. NOSSDAV '98*, July 1998.
[89] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-Like Congestion Control for Layered Multicast Data Transfer," *Proc. IEEE INFOCOM '98*, Mar. 1998.

Biographies

BIN WANG (bwang@ee.eng.ohio-state.edu) received his Bachelor of Engineering degree from Zhejiang University, China, in 1992, and his M.S. degree in computer science from the University of Louisville in 1994. From September 1994 to August 1996 he was a research assistant in the Department of Electrical and Computer Engineering, the University of Wisconsin-Madison. Since September 1996, he has been with The Ohio State University, where he is a Ph.D. candidate in the Department of Electrical Engineering. His research interests include multicast networking, multimedia networking, QoS routing, QoS provisioning for next-generation internetworks, and wireless and mobile communications.

JENNIFER C. HOU [M] received a B.S.E. degree in electrical engineering from National Taiwan University, China, in 1987; and M.S.E degrees in EECS and I&OE in 1989 and 1991, respectively, and a Ph.D. degree in EECS in 1993, all from the University of Michigan, Ann Arbor. From August 1993 to July 1996, she was an assistant professor in the Department of ECE at the University of Wisconsin, Madison. Since August 1996 she has been with the Department of Electrical Engineering at The Ohio State University, where she is currently an associate professor. She was a recipient of the NSF CAREER award from the Advanced Network and Research Infrastructure program, National Science Foundation, in 1996. She has served on the technical program committees of numerous networking, protocol design, real-time, and distributed systems conferences/symposia, and is Program Chair of the IEEE 6th Real-Time Technology and Applications Symposium. Her research interests are in the areas of QoS provisioning in high-speed networks, QoS-driven multicast routing, multicast support to Mobile IP, congestion control, network simulation, and real-time computing.