Multicast Protocols and Algorithms

George C. Polyzos and George Xylomenos
Athens University of Economics and Business, Athens, Greece

Abstract

Multicast refers to the transmission of data from one sender to an arbitrary set of receivers, a service useful for applications such as group conferencing, media distribution, and update of replicated databases. From an application viewpoint, the importance of multicast is that it allows adressing a set of receivers via a common identifier. From a network viewpoint, the importance of multicast is that when the transmission paths between the sender and the receivers share some links, multicast can conserve resources by only sending the data once over each such link. The most important issues raised by multicast are routing data to multiple destinations with minimal delay and/or duplication, handling feedback from possibly numerous receivers in a scalable manner, and providing appropriate quality-of-service for data delivery to each receiver.

I. Introduction

One way to characterize communication is by the number of parties involved. The traditional communication modes are *unicast*, i.e., one-to-one, and *broadcast*, i.e., one-to-all. Between these two extremes we find *multicast*, the transmission of a message or datastream to an arbitrary set of receivers, i.e., one-to-many. Multicast can be seen as a unifying communication mode, as it is a generalization of both unicast and broadcast. Multicast is examined separately, however, because the specification of receivers as a set introduces features and complications that are not present in traditional unicast and broadcast. A more general term is *multipoint communication*, which implies many-to-many bidirectional data exchange.

The multicast model of communication is ideal for applications where data and control are partitioned over multiple entities. Examples include updating replicated databases, contacting any one of a group of distributed servers of which the composition is unknown (more appropriately termed *anycast*), and interprocess communication between multiple cooperating processes. The prototypical multicast applications, however, are real-time interactive multimedia conferencing and near real-time media distribution to multiple receivers.

Multicast efficiency is a fundamental requirement for the success of many group applications. Selective multicast replaces indiscriminate broadcasting to everyone, reducing the waste of resources caused by transmitting information to all receivers. To be more economical than unicast, multicast must conserve resources via *sharing*: Instead of transmitting information from a sender to each receiver separately, routes to receivers that share links must carry the information only once over each shared link. We can picture a multicast route as a *tree* rooted at the sender with a receiver at each leaf and, possibly, some receivers on internal nodes. This tree must be designed to maximize link sharing and thus minimize resource consumption.

Besides the obvious issue of how to construct multicast routing trees, multicast also raises other issues related to the extension of unicast mechanisms to a multicast context. For example, the sender in a reliable transport protocol can recover from communication errors based on error reports from the receiver. By simply extending this mechanism to multicast, we run the risk of *feedback implosion*, when many receivers send such reports toward the sender, thus swamping the network and the source with control information. In addition to the scalability issues raised by this approach, another issue is how the sender should react when conflicting reports arrive from different receivers.

II. Multicast Models

The difference between multicasting and separately unicasting to several destinations is best captured by the Internet host-group model of Cheriton and Deering [1]: A host-group is a set of network entities sharing a common identifying multicast address. All group members receive any data packets addressed to this multicast address. The senders have no knowledge of group membership and may or may not belong to the group, corresponding to *closed* or *open* groups, respectively. Multicast messages on the Internet are sent on a best-effort basis, like unicast messages; i.e., they may be reordered, lost, or duplicated.

This definition allows group behavior over time to be unrestricted in multiple dimensions; it may have local (LAN) or global (WAN) membership, be transient or persistent, and have static or dynamic membership. From the sender's point of view, the multicast service interface is identical to unicast; only the address differs. Therefore, it is the network's responsibility to manage the multicast communication, transparently to the users. This extra work compared with unicast is expected to result in a more efficient usage of resources, which is the primary motive for network providers to support multicast in the first place.

The host-group model imposes specific requirements for the implementation of the multicast service. First, there must be a means for routing packets from a sender to all group members, which implies that the network must locate all members of the group and make appropriate routing arrangements without any assistance from the sender. Second, as group membership is dynamic, the network must continuously track membership during a session's lifetime, which may range from short to very long periods of time. Tracking is required both to start forwarding data to new group members and to stop the wasteful

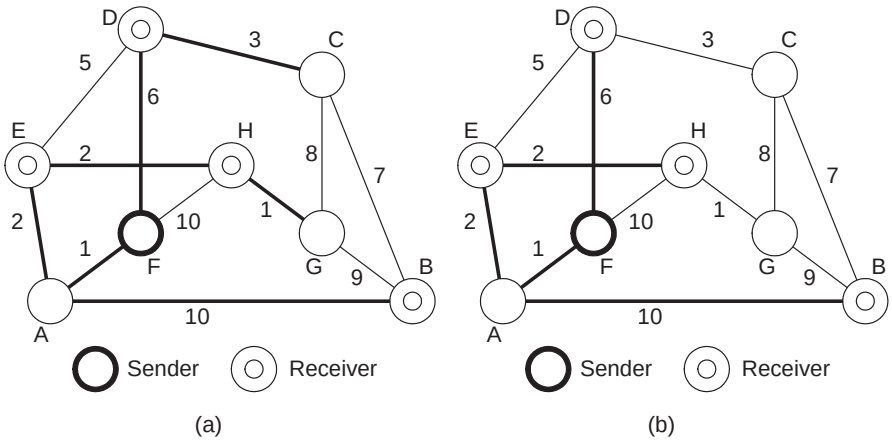

Fig. 1. (a) The broadcast tree formed by the shortest paths from the sender to all nodes. (b) The multicast tree obtained by pruning all links that do not lead to receivers from the broadcast tree.

transmission of data to members that have left the group. This dynamic nature of multicast groups has a considerable impact on multicast routing.

It should be noted that the Internet host-group model is by no means unique for multicasting. Some applications require delivery of messages addressed to a group to be *atomic*, that is, each message sent to a multicast group must be received by either all receivers in the group or none at all. Atomicity further implies that all received messages must be processed in the same order by all receivers, i.e., that multicasts are also totally ordered. As atomic multicast is complex to implement, it is usually built on top of a simpler multicast facility, such as the one offered by the Internet.

III. Multicast Routing Algorithms

Unicast routing attempts to minimize either transmission cost or delay, depending on the metric used for optimization. Although these goals seem different, from an algorithmic point of view, they are both equivalent to finding shortest paths over a network with cost-labeled links; link costs may stand for either transmission cost or delay. A shortest path algorithm finds optimal routes between one node (the sender) and *all* other nodes in the network. Two common examples of such algorithms are those due to Dijkstra and Bellman–Ford. An optimal route minimizes the sum of the costs of all links included in the route. The union of all these routes forms a *shortest path tree* rooted at the sender. As this is a broadcast tree, as shown in Fig. 1(a), a straightforward (but not optimal) solution to the multicast routing problem is to prune off that tree all links that do not lead to any members of the group, as shown in Fig. 1(b).

The advantage of these algorithms is that they are easy to implement and deploy, as they are direct extensions of existing ones. Each path is optimal by definition, regardless of changes in group membership, and this optimality comes essentially for free, because shortest paths need to be computed anyway for unicast routing. The disadvantage of these algorithms is that they concentrate on pairwise optimizations between the sender and each receiver and only conserve resources as a side effect, when paths happen to overlap. For large networks with widely dispersed group members, either the scale of the network or the continuous network changes will necessarily restrict the use of these algorithms to subnetworks, requiring a hierarchical routing technique to support global multicasting.

To achieve the economies promised by multicasting, optimization must be viewed from the perspective of the entire distribution tree. This requires building trees that exploit link sharing as much as possible, duplicating packets only when paths diverge, so as to minimize the total distribution cost even at the expense of serving some receivers over longer paths. In algorithmic terms, what is needed is a minimal cost tree that reaches all receivers, possibly using additional nodes on the way. This is equivalent to the *Steiner tree* problem, analyzed by Hakimi [2]. In this problem, a cost-labeled graph and a set of nodes, the Steiner points, are given and a minimal cost tree is sought connecting all Steiner points, including both the sender and all receivers, as shown in Fig. 2(a).

If all nodes were Steiner points, the above problem would coincide with the *spanning tree problem*, which can be solved efficiently. Unfortunately, the Steiner tree problem belongs to the class of NP-complete problems, as shown by Garey et al. [3]. Fortunately, approximation algorithms exist for it with proven constant worst-case bounds and very good average behavior. Trees built with the heuristic by Kou et al. [4] have at most twice the cost of Steiner trees, whereas simulations of realistic network topologies described by Kabada and Jaffe [5] have shown their cost to be within 5% of the optimum.

The advantage of Steiner tree algorithms is their overall optimality with respect to a single cost metric, such as transmission cost. Their disadvantages are also important, however: These algorithms must be run in addition to the unicast routing algorithm, and they suffer from scaling problems for large networks. Furthermore, optimality is generally lost after changes in group

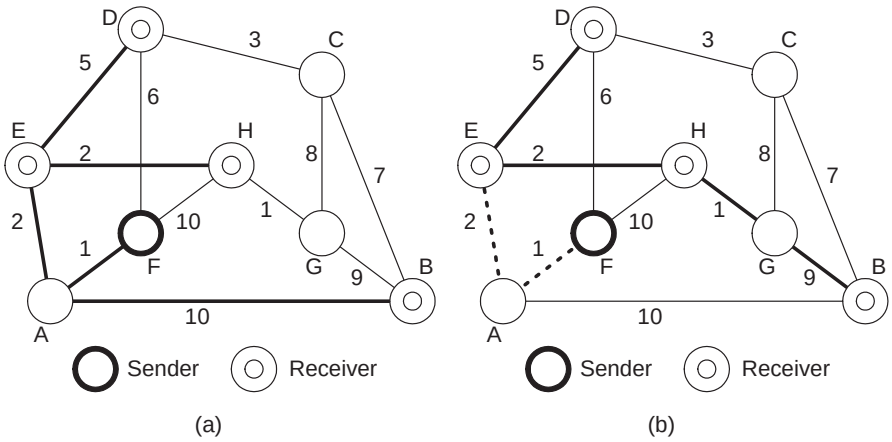

Fig. 2. (a) The Steiner tree obtained by minimizing the overall cost from the sender to all receivers. (b) The core based tree formed by the shortest paths from the core (node H) to all receivers. The sender uses the shortest path to the core for data transmission.

membership and network reconfigurations, unless the tree is repeatedly recomputed from scratch. Approaches for extending these algorithms to deal with changes in group membership without tree recomputation include extending the existing tree in the cheapest way possible to support new group members and pruning redundant links when group members depart. The quality of the tree will deteriorate over time after several such modifications, eventually leading to the need for tree recomputation. Thus, Steiner tree algorithms are best suited to static or slowly changing environments because changes eventually lead to expensive tree recomputation to regain optimality.

Shortest path trees and Steiner trees are optimal with respect to a sender; therefore a separate tree must be built for each sender in both cases. A different approach is to employ a *center-based tree*, which, instead of being rooted at the sender, is rooted at the topological center of the receivers. A single center-based tree serves as a common infrastructure for all senders; therefore, maintenance of the tree is greatly simplified and nodes belonging to the tree need only maintain state for one shared tree rather than for many source-rooted trees. Even though such a tree may not be optimal for any one sender, it may be an adequate approximation for all of them together. Unfortunately, the topological center of the receivers, apart from being hard to find (this problem is also NP-complete), is not even permanent in a dynamic multicast environment.

A more practical proposal is to abandon the topological center as the root of the tree, keeping the basic idea of a single shared multicast tree for all senders to a group. In this approach, routing is performed by defining one or more arbitrarily selected *core* (or *rendez-vous*) points to serve as the basis for tree construction for a group. All senders transmit their data to the core using an optimal (in the unicast sense) route, and the core uses a shortest path tree to distribute these data to all group members, as shown in Fig. 2(b). As in any shortest path tree, merging of paths is exploited whenever possible, but it is not an explicit goal of the routing calculations. Due to the concentration of paths around the core, though, common paths are expected to arise. Although this approach uses an underlying unicast routing algorithm, it is independent of it.

The disadvantage of this approach is that a single shared multicast tree, especially if it is rooted at an arbitrary node, is not optimal in any strict sense. The advantages of shared multicast trees are numerous, however. First, the shared tree means that this approach scales well in terms of maintenance costs as the number of senders increases. Although there is still a tree emanating from each sender, these trees merge near the core and the distribution mesh is common from there on. Second, the trees can be made efficient by choosing appropriately the core points. Third, routing is performed independently for each sender and receiver, with entering and departing receivers influencing only their own path to the core points of the shared tree. This last property means that network and group membership dynamics can be dealt with without global recomputation. Finally, the independence from specific unicast routing schemes, coupled with the scalability of the shared trees, makes this approach ideal for use on large networks. The core points may even be selected to facilitate hierarchical routing; i.e., a top-level tree can distribute data to the core point of each subnetwork, and each core point can then distribute data to the group members in its subnetwork.

Despite the differences between the approaches discussed above, simulations have shown that even simple multicast routing using shortest path trees is not significantly worse in terms of total tree cost from the optimal solutions. For realistic network topologies, Doar and Leslie [6] have found that the cost of a shortest path tree is less than 50% larger than that of a near-optimal heuristic tree, whereas path delays for heuristic trees are 30% to 70% larger than shortest path delays. As shortest path trees are easily built and modified using the underlying unicast routing algorithm and they never deteriorate in terms of delay, but simply vary in their inefficiency in terms of total cost, an application prepared to accept this overhead can avoid special multicast tree construction and maintenance methods by simply employing the shortest paths.

A similar cost versus simplicity tradeoff is involved when using shared trees, for all senders to a group. With shared trees,

optimality is hard to achieve and even harder to maintain; a simple approach is to choose the best core point among group members only. With this limitation, when path delay is optimized, simulations show that delays are close to 20% larger than with shortest paths, and tree cost is about 10% lower than that of shortest path trees. Furthermore, even though a single tree minimizes state and maintenance overhead, this approach suffers from traffic concentration, exactly due to the single tree used, because it routes data from all senders through the same links around the core. Simulations show that delay-optimal member-centered shared trees can cause maximum link loads to be up to 30% larger than in a shortest path tree.

IV. Feedback Control

When the basic service offered by the network is a best-effort one, as in the Internet, generalizing it for multicast is straightforward: Just send the data along the multicast routing tree without providing any guarantees with respect to reliability, throughput, or delay. Many applications, however, cannot be satisfied by such a service; therefore, mechanisms such as flow, congestion, and error control have to be provided on top of this best-effort service. These mechanisms depend on feedback to the sender, which is based on either network- or receiver-generated reports.

Error control ensures that packets transmitted by the sender are received correctly. Packets may be received corrupted (detected by error-detection codes), or they may be lost (detected by missing sequence numbers). Flow control assures that the sender does not swamp the receiver with data that cannot be consumed in time. Congestion control limits the transmission rate of the sender to avoid overloading the intermediate network nodes on the way to the receiver. Although error and flow control require feedback from the receiver, congestion control would be best served by feedback from the intermediate nodes themselves. In best-effort networks like the Internet, however, it is only the receivers that provide feedback about packet losses to the sender, thus leading to confusion between error-induced and congestion-induced losses.

In the unicast case, flow, error, and congestion control rely on feedback from a unique receiver. For example, loss reports may cause the retransmission of lost packets. With multicast, however, this approach faces the feedback-implosion problem: If all receivers respond with status information, they will swamp the sender with, possibly conflicting, reports. Ideally, senders would like to deal with the multicast group as a whole, not on an individual receiver basis, following the host-group model. The sender cannot simply treat all receivers identically, though, because this requires either ignoring the feedback of some receivers or wasting resources by satisfying the worst-case receivers. For example, the sender could retransmit only packets lost by all receivers, thus ignoring some losses, or retransmit any packets lost by any receiver, thus duplicating some packets.

As there is no evident solution to this problem, several approaches exist emphasizing different goals. The simplest approach is to ignore the problem at the network and simply provide a best-effort service. Delegating the resolution of these problems to higher layers may be an adequate solution in many cases, because these layers may have additional information about application requirements and be able to implement more appropriate mechanisms than what is possible inside the network. Even in this case, though, higher layers will have to implement one of the alternative approaches discussed below.

A second solution sacrifices the host-group model's simplicity by keeping per-receiver state at the sender during multicasts. After transmitting a multicast packet, the sender waits until a stable state is reached before sending the next one. For example, in error control, retransmissions may be made until all receivers receive the data. To economize on resources, retransmissions may be multicast when many receivers lose a packet, or unicast when few do. To reduce the risk of feedback implosion, receivers should use negative rather than positive acknowledgments, i.e., send responses only when problems occur, rather than to confirm that packets were received correctly. Furthermore, these negative acknowledgments may be multicast to all receivers after waiting for a random period of time, so as to suppress identical negative acknowledgments from multiple receivers, as suggested by Towsley et al. [7].

Even with these optimizations, the scalability of such schemes is doubtful for large and widely dispersed groups, even when errors, overflows, and congestion are very rare, because the sender remains solely in charge of all receivers. In addition, with these schemes the service provided to a group member is the lowest common denominator, which may be the slowest or most overloaded receiver, or the slowest or most congested link. While more sophisticated variations of this approach exist, their complexity and inefficiency makes them appropriate only for specific applications.

A third solution is to distribute the feedback control mechanism over the entire multicast tree, so as to avoid propagating the receiver's feedback all the way to the sender. In a hierarchical scheme, the intermediate nodes may either respond directly to feedback from downstream receivers or merge their feedback into a summary message and recursively propagate it upstream. If the added complexity of making local decisions on each node (not only group members) is acceptable, this approach narrows down the impact of problems to specific parts of the tree, relieving the sender from dealing with individual receivers. Note that even though this scheme avoids feedback implosion, the problem of dealing with possibly conflicting requests remains.

An alternative non-hierarchical method for distributed feedback control, targeted especially to error control, is to let all receivers and senders cooperate in handling losses, as proposed by Floyd et al. [8]. When receivers discover a loss, they multicast a retransmission request, and anyone that has that message can multicast it again. Both requests and replies can have local scope, if the network supports it, so as to avoid burdening the entire group. To avoid feedback implosion, these requests and replies are sent after a fixed delay based on the distance from the source of the message or the request, respectively, plus a randomized delay. The result is that most duplicate requests and replies are suppressed by the reception of the first one.

By varying the random delays, the desired balance between recovery delay and duplicates can be achieved, and in contrast to hierarchical schemes, only group members participate in recovery.

A fourth solution is for the sender to act based on an estimation of the average conditions across the group. A scalable feedback mechanism for this estimation has been proposed by Bolot et al. [9]: It first estimates the number of receivers in a group and then what the average quality of reception is, using probabilistic techniques. This method can be used to detect congestion problems and adapt the transmission rate (to relieve congestion) or the error redundancy factor (to increase the chances of error recovery). In a refinement of this approach, proposed by Cheung and Ammar [10], the sender splits the receivers into groups according to their capabilities and only sends to each group the data that it can handle. This scheme prevents very fast or very slow receivers from dragging the whole group toward one extreme case.

Finally, another approach (mostly orthogonal to the above) tries to minimize the need for feedback by taking preventive rather than corrective action. For error control, this is achieved by using *forward error correction* (FEC) rather than error detection codes and retransmissions. For flow and congestion control, this is achieved by reserving resources in advance so that both receivers and intermediate nodes can support the sender's data rate. Although FEC imposes considerable processing and transmission overhead, it requires no additional network mechanisms. Resource reservations, however, require additional network mechanisms to set up and maintain the resources for each session.

V. Multimedia Multicasting

A common use of multicasting is for multimedia communication, i.e., the exchange of multiple interdependent media types, such as text, audio, and video. As continuous media, i.e., audio and video, require considerable transmission bandwidth, the economies promised by multicasting are especially attractive in this context. The issues arising when multimedia are combined with multicasting are treated more extensively by Pasquale et al. [11].

A. Host and Network Heterogeneity

Several representational formats exist for each media type, and each participant in a multicast group may support a different set of formats. In unicast, translation is equally effective at either the sender, or the receiver. In multicast, translation at the sender would require the stream to be duplicated and translated for each different type of receiver, preventing link sharing over common paths, placing excessive load on the sender, and requiring the sender to be aware of each receiver's capabilities, thus violating the host-group model. Translation at the receiver is the most economical and scalable approach in this case, because it fully exploits sharing and moves responsibilities away from the sender.

As continuous media impose heavy demands on both networks and hosts, it is likely that not all receivers will be able to receive all of a sender's traffic. This argues in favor of prioritization of the traffic generated through *hierarchical* coding. Hierarchical or *layered* coding techniques decompose a signal into independent or hierarchically dependent components, subsets of which can be used to provide partial reconstruction of the original. Receivers can thus choose only those parts of the media that they can use or are most important to them. For example, a high-resolution component of a video could be dropped from a congested subnetwork, allowing low-resolution components to be received and displayed in that subnetwork, without impacting uncongested subnetworks.

To avoid complicating the host-group model, each component of a hierarchically coded stream may be transmitted to a different multicast group, making the choice of a particular component equivalent to subscribing to the corresponding group. Based on this approach, Vicisano et al. [12] have proposed a purely receiver-driven congestion control scheme, where each receiver estimates the capacity of the network based on packet losses and only subscribes to as many groups as can be realistically delivered by its subnetwork. The sender periodically doubles its transmission rate for each component so as to enable the receivers to decide whether improved network conditions allow the reception of additional media components.

B. Resource Reservations

For interactive multimedia applications to be practical, the network must be able to provide some type of bandwidth and delay guarantees. If any such guarantees are to be provided, resources must be reserved at the various network nodes traversed. The exact nature of the reservations depends on the required service guarantees and the approach taken toward satisfying them. In any case, the first component of any resource reservation scheme is a specification model for describing flow characteristics; this depends heavily on the model of service guarantees supported by the network. The second component is a protocol for communicating these specifications to the receivers and reserving resources along the transmission path so as to support the requested services.

The simplest unicast approaches to resource reservations are source-based. A setup message containing the flow specification is sent to the destination, with the intermediate nodes committing adequate resources for the connection, if available. Resources are normally overallocated early on in the path, so that even if nodes encountered further along the path are short on resources, connection setup may still succeed. After the setup message reaches its destination, and assuming the connection can be admitted along the path, a response message is returned on the reverse path, allowing intermediate nodes to relax any excessive commitments made on the first pass.

Similarly, for multicast, there must be a way for senders to notify receivers of their properties, so that appropriate reservations may be made. In a homogeneous environment, reservations should be made once on each outgoing link for all downstream receivers, so as to minimize resource usage. Reserved resources may even be shared among multiple senders to the same group. However, receiver and network heterogeneity often prohibits use of this simplistic scheme, because the amount of resources that are available at each part of the multicast tree may be quite different. A modified scheme is to allocate resources as before during the first message's trip and then have all receivers send back their relaxation (or rejection) messages. Each node that acts as a junction only propagates toward the source the most restrictive relaxation among all those received. However, as paths from such junctions toward receivers may have committed more resources than are now needed, additional passes will be required for convergence or resources will be wasted.

An alternative is to abandon reservations during the sender's setup message, instead reserving resources based on the modified specifications returned by the receivers. Again, resource reservations are merged on junction points, but as these requests are expected to be heterogeneous, each junction will reserve adequate resources for the most demanding receiver and reuse them to support the less demanding ones. This approach supports both heterogeneous requests and resource conservation, thus maximizing the possibility for a new session to be admitted. As this mechanism converges in one pass, the reservation state in the switches can be periodically refreshed, turning the fixed state of a static connection into adaptive state suitable for a dynamic environment. Therefore, this mechanism can accommodate both group membership changes and routing modifications without involving the sender.

C. Quality-of-Service Routing

When multicast is used for multimedia communications, link sharing can lead to considerable economies in transmission bandwidth, but routing must also take into account two additional factors: Delay constraints, particularly for interactive applications, and media heterogeneity. Separate handling of media streams allows using the most effective coding technique for each stream. The question arises then of whether the same or separate distribution trees should be used for each stream. Considering the load that continuous media put on network links, separate trees seem preferable. Thus, each media stream could ask for the appropriate quality-of-service (QoS) parameters and get routed accordingly, with receivers choosing to participate in any subset of these trees. On the other hand, the management overhead of multiple trees per source may be prohibitive, whereas routing each media stream separately may complicate inter-media synchronization.

Turning to delay constraints, assuming that we use delay as the link cost during routing, we already saw that the shortest path tree and the Steiner tree are different: The former minimizes individual path delays, whereas the latter minimizes overall distribution delay and maximizes link sharing. As the global tree metric and the individual receiver-oriented metrics are potentially in conflict, we cannot hope to optimize both. We can, however, try to optimize the global metric subject to the constraint that the individual metrics are tolerable. As interactive applications can be characterized by upper bounds on end-to-end delay, it is reasonable to design the tree to optimize total cost while keeping individual paths within some bound. Normally, all receivers are satisfied by the same delay bound, as this is determined by human perception properties.

This problem is essentially a version of the Steiner tree problem with additional constraints on the paths. Even though it is also NP-complete, fast heuristic algorithms that are nearly optimal have been developed, for example, by Kompella et al. [13]. Almost identical formulations are obtained when the constraints are delay *jitter*, i.e., the variation of delay, or a probabilistic reliability constraint. For example, in the case of independent link losses, a loss probability can be assigned to each link. By using logarithms, the reliability metric can be calculated in linear form between a source and each destination by adding the logarithms along the path. Thus, the problem reverts to tree cost minimization with a constraint on an additive path-based metric. Finally, the constraint may be a link capacity that must not be exceeded. Again, heuristic algorithms exist to solve this variant of the problem.

VI. Multicast Protocols on the Internet

A. The IP Multicasting Model

The Internet, due to its open architecture, has been extensively used as a testbed for multicast algorithms and protocols. IP multicasting is based on special (class D) multicast IP addresses. By simply using a class D address as the destination of a *datagram*, i.e., an IP packet, it is multicast to all group members rather than unicast. To achieve multicasting in a wide-area network, such as the Internet, a mechanism is needed to keep track of the dynamic membership of each group and another mechanism is needed to route multicast datagrams from a sender to these group members without unnecessary duplication of traffic. IP multicasting implements these mechanisms in two parts: Local mechanisms track group membership and deliver multicasts to group members within a local network, and global mechanisms route datagrams between local networks.

In each local network, at least one router acts as a multicast router. A multicast router keeps track of local group membership and is responsible for forwarding multicasts originating from its local network toward other networks, as well as for delivering multicasts originating elsewhere to the local network. The delivery of multicast datagrams to local receivers, as well as the reception of local multicasts by the router for subsequent propagation to other networks, depend on the underlying network

technology. Therefore, the information needed within the local network regarding group membership in order to achieve multicast delivery may vary.

In contrast, cooperation among multicast routers for the delivery of multicast datagrams between networks is based on a network-independent interface between each network and the outside world. The information needed to decide whether multicasts should be delivered to target networks is whether at least one group member for a destination group is present there, regardless of the information the multicast router needs for local purposes. A multicast router uses the list of groups present on its attached local networks along with information exchanged with its neighboring routers to support wide-area multicasting. Based on this interface, alternative algorithms can be used for global routing without affecting local mechanisms. Conversely, as long as this interface is provided by the local mechanisms, they can be modified without affecting global routing.

B. Global Mechanisms

A variety of global, wide-area, multicast routing mechanisms exist. The earliest one, proposed by Deering and Cheriton [14], is the distance vector multicast routing protocol (DVMRP). The original version of DVMRP is a variant of the truncated reverse path broadcasting algorithm. Routers construct distribution trees for each source sending to a group, so that datagrams from the source (root) are duplicated only when tree branches diverge toward destination networks (leaves). To construct the tree, each router identifies the first link on the shortest path from itself to the source, i.e., on the shortest *reverse* path, using the Bellman–Ford distance vector unicast routing algorithm. Datagrams arriving from this link are forwarded toward downstream multicast routers, i.e., those routers that depend on the current one for multicasts from that source. A broadcast distribution tree is thus formed, with datagrams reaching all routers. As each router knows which groups are present in its local networks, redundant datagrams are not forwarded there and the tree is truncated at the lowest level. The latest version of DVMRP implements the improved reverse path multicasting algorithm, where links leading to networks with no members for a group are pruned off the tree and are grafted back when members appear for these groups. Although initially all data are broadcast, eventually the tree becomes a real multicasting one.

Another protocol proposed by Moy [15], multicast open shortest path first (MOSPF), extends Dijkstra's link state unicast routing algorithm. Routers flood their membership lists among them, so that each one has complete topological information concerning group membership. Shortest path multicast distribution trees from a source to all destinations are computed on demand as datagrams arrive. These trees are real multicast ones, but the flooding algorithm used to construct them introduces considerable overhead.

A radically different approach is the core-based tree (CBT) protocol proposed by Ballardie et al. [16], which employs a single tree for each group, shared among all sources. This tree is rooted on an arbitrarily chosen router, the *core*, and extends to all networks containing group members. It is constructed from leaf network routers toward the core as group members appear; thus, it is composed of shortest reverse paths. Sending to the group is accomplished by sending toward the core; when the datagram reaches any router on the tree, it is relayed toward tree leaves. Routing is thus a two-stage process that can be suboptimal, as datagrams may be sent away from the receivers during the first stage.

As both shortest path trees and center-based trees have advantages and disadvantages, the protocol independent multicast (PIM) protocol proposed by Deering et al. [17] provides both. PIM supports two modes, the dense mode and the sparse mode. The dense mode is similar to DVMRP but independent of the underlying unicast routing algorithm. The sparse mode starts similarly to CBT, constructing a shared tree for all receivers using a core, called a *rendez-vous* point in PIM, but it allows paths to individual receivers to be switched to shortest delay ones upon receiver request.

Networks supporting IP multicasting may be separated by multicast unaware routers. To interconnect such networks, tunnels are used, i.e., virtual links between two endpoints, composed of a, possibly varying, sequence of physical links. Multicasts are relayed between routers by encapsulating multicast datagrams within unicast datagrams at the sending end of the tunnel and decapsulating them at the other end. Multicast routers may choose to forward through the tunnels only datagrams that have time-to-live (TTL) values above a threshold, so as to limit multicast propagation across networks.

For unicast routing to scale, the Internet is divided into autonomous systems (AS), i.e., areas that internally run a single routing protocol, probably different for each AS. The border routers of each AS run a common routing protocol to achieve global unicast routing. Similarly, although multicast routing within an AS can use any of the above protocols, a common protocol, such as the border gateway multicast protocol (BGMP) proposed by Kumar et al. [18], must be used among the border routers of each AS to achieve global routing. BGMP constructs shared trees between those ASs containing group members, using as the core the AS where the multicast group was originally created, thus bridging the multicast routing protocols used within each AS.

C. Local Mechanisms

Unlike global mechanisms, only a single set of local mechanisms exists. These local multicasting and group management mechanisms are based on shared-medium broadcast-based networks, such as Ethernet. Delivery is straightforward on such networks, because each host can listen to all messages and select only those with the appropriate addresses. As an optimization,

class D IP addresses may be mapped, if possible, to native multicast addresses so as to filter datagrams in hardware rather than in software.

On these networks, multicasts with local scope do not require any intervention by the multicast router, whereas externally originated multicasts are directly delivered to the local network by the router. The router monitors all multicast transmissions on the local network so that it may forward to the outside world those for which receivers exist elsewhere. The router does not need to track individual group members; the only information needed to decide whether an externally originated multicast must be delivered to the local network is whether at least one group member exists in the network. Therefore, the multicast router only requires a local group membership list.

The Internet group management protocol (IGMP) provides a mechanism for group management well suited to broadcast networks, because only group presence or absence is tracked for each group. In the original version of IGMP, the multicast router periodically sends a query message to a multicast address to which all local receivers listento. Each host, on reception of the query, schedules a reply to be sent, after a random delay, for each group in which it participates. Replies are sent to the address for the group being reported, so that the first reply will be heard by all group members and suppress their transmissions. The multicast router monitors all multicast addresses, updating its membership list after receiving each reply. If no reply is received for a previously present group for several queries, the group is assumed absent. When a host joins a group it sends several unsolicited reports to reduce join latency if it is the first local member of the group. No explicit action is required when a host leaves a group, as group presence eventually times out.

During the time interval between the last host leaving a group and the router stopping multicast delivery for that group, called the *leave latency*, local transmissions to the group are wasted. To reduce this phenomenon, in the latest version of IGMP a host must send a leave message when abandoning a group if it was the last host to send a report for that group. As this last report may have suppressed other reports, the router must explicitly probe for other group members by sending a group-specific query to trigger membership reports for this group. The router can only assume the group absent if no reports arrive after several such queries. Group-specific queries may use much shorter response intervals than general queries, so as to minimize leave latency.

For networks consisting of point-to-point links, such as dialup links between home users and their Internet Service Providers, only a single member per multicast group can exist at the end-user side. An extension to IGMP for such networks, proposed by Xylomenos and Polyzos [19], uses only explicit join and leave messages from the end-user side to the multicast router, thus reducing both join and leave latency, as well as avoiding periodic queries and reports.

D. Related Protocols

In addition to protocols ensuring multicast delivery, many other protocols, directly or indirectly related to multicast, exist on the Internet. The multicast address-set claim (MASC) protocol, proposed by Kumar et al. [18], allows the entire range of class D IP multicast addresses to be distributed between ASs, so as to avoid addressing conflicts when multicast groups originating in different networks choose the same address.

The session announcement protocol (SAP) is used to announce the existence of multicast sessions along with their addresses, so that interested receivers may join the group. The session description protocol (SDP) describes the media formats comprising a session so that interested receivers will know what to expect after joining the group.

If the receivers require QoS guarantees, they may use the resource reservation protocol (RSVP) by Zhang et al. [20], to signal their requirements to the sender. When RSVP requests from multiple receivers meet on the way to the sender, they are merged into a single reservation that satisfies the most demanding request. As RSVP reservations are periodically refreshed, dynamic reservation modifications and network reconfigurations are supported.

E. Unresolved Issues

Although multicasting is widely considered to be a valuable service, it is still not universally supported over the Internet. Many explanations for this phenomenon are proposed by Diot et al. [21], including security and scalability issues. Although the traditional security issues raised by unicast, such as data confidentiality and integrity, are also valid for multicast, in the multicast context, they are more difficult to address. For example, secure group communication can be provided by using independent end-to-end secure unicast channels between all pairs of participants, albeit by negating the link sharing advantages of multicast. In the host-group model adopted by the Internet, group membership is unknown to the sender; therefore, it is impossible to setup security associations between the sender and the receivers without tracking additional information.

Another issue that became apparent when multicast started becoming popular is that the amount of forwarding state required at each multicast router for global multicasting does not scale well, because separate entries are needed for every multicast group, even if a single tree is used for delivery. If shortest path trees are used instead, the number of forwarding entries must be multiplied by the number of senders to the group. In unicast routing, this problem is solved by aggregating the forwarding state based on the fact that networks with similar unicast IP addresses are usually geographically close; thereforem routers only need a single aggregate entry for many different IP addresses, pointing in the appropriate direction. Unfortunately, multicast groups may have members everywhere on the Internet; therefore, this type of aggregation is not generally possible. Various other aggregation methods have been proposed, as described by Zhang and Mouftah [22].

REFERENCES

[1] D. R. Cheriton and S. E. Deering, Host groups: A multicast extension for datagram internetworks, *Proc. of the Data Communications Symposium*, Vol. **9**, 1985, 172–179.

[2] S. L. Hakimi, Steiner's problem in graphs and its implications, *Networks*, **1**: 113–133, 1971.

[3] M. R. Garey, R. L. Graham, and D. S. Johnson, The complexity of computing Steiner minimal trees. *SIAM J. on Appl. Math*, **34**: 477–95, 1978.

[4] L. Kou, G. Markowsky, and L. Berman, A fast algorithm for Steiner trees. *Acta Informatica*, **15**: 141–145, 1981.

[5] B. K. Kabada and J. M. Jaffe, Routing to multiple destinations in computer networks. *IEEE Trans. on Commun.*, **31**: 343–351, 1983.

[6] M. Doar and I. Leslie, How bad is naive multicast routing? *Proc. of the IEEE INFOCOM*, Vol. **12**, 1993, 82–89.

[7] D. Towsley, J. Kurose, and S. Pingali, A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE J. on Select. Areas Commun.*, **15**: 398–406, 1997.

[8] S. Floyd, V. Jacobson, S. McCanne, C.G. Liu, and L. Zhang, A reliable multicast framework for light-weight Sessions and application level framing. *Comput. Commun. Rev.*, **25**(4): 342–356, 1995.

[9] J. C. Bolot, T. Turletti, and I. Wakeman, Scalable feedback control for multicast video distribution in the Internet. *Comput. Commun. Rev.*, **24**(4): 58–67, 1994.

[10] S. Y. Cheung and M. H. Ammar, Using destination set grouping to improve the performance of window-controlled multipoint connections. *Comput. Commun.*, **19**: 723–736, 1996.

[11] J. C. Pasquale, G. C. Polyzos, and G. Xylomenos, The multimedia multicast problem. *Multimedia Syst.*, **6**(1): 43–59, 1998.

[12] L. Vicisano, J. Crowcroft and L. Rizzo, TCP-like congestion control for layered multicast data transfer. *Proc. of the IEEE INFOCOM*, Vol. **17**, 1993, pp. 996–1003.

[13] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, Multicast routing for multimedia communication. *IEEE/ACM Trans. Networking*, **1**(3): 286–292, 1993.

[14] S. E. Deering and D. R. Cheriton, Multicast routing in datagram internetworks and extended LANs. *ACM Trans. Comput. Syst.*, **8**(2): 85–110, 1990.

[15] J. Moy, Multicast routing extensions for OSPF. *Commun. ACM*, **37**(8): 61–66, 1994.

[16] A. Ballardie, J. Crowcroft, and P. Francis, Core Based Trees (CBT) – An architecture for scalable inter-domain multicast routing. *Comput. Commun. Rev.*, **23**(4): 85–95, 1993.

[17] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, The PIM architecture for wide-area multicast routing. *IEEE/ACM Trans. Networking*, **4**: 153–162, 1996.

[18] S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D. Estrin and M. Handley, The MASC/BGMP architecture for inter-domain multicast routing. *Comput. Commun. Rev.*, **28**(4): 93–104, 1994.

[19] G. Xylomenos and G. C. Polyzos, IP multicast group management for point-to-point local distribution. *Comput. Commun.*, **21**(18), 1645–1654, 1998.

[20] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, RSVP: a new resource ReSerVation Protocol. *IEEE Network*, **7**(5): 8–18, 1993.

[21] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, Deployment issues for the IP multicast service and architecture. *IEEE Network*, **14**(1): 78–88, 2000

[22] B. Zhang and H. T. Mouftah, Forwarding state scalability for multicast provisioning in IP networks. *IEEE Communications*, **41**(6): 46–51, 2003.