# The multimedia multicasting problem

Joseph C. Pasquale and George C. Polyzos and George Xylomenos
{pasquale,polyzos,xgeorge}@cs.ucsd.edu
Computer Systems Laboratory,
Department of Computer Science and Engineering,
University of California, San Diego,
La Jolla, CA 92093-0114, USA

***Abstract*—This paper explores the problems associated with the multicasting of continuous media to support multimedia group applications. The interaction between multicasting and the delivery of multiple time-correlated continuous-media streams with real-time delay requirements poses various new and interesting problems in research on communication protocols and architectures. We describe these problems, and identify where the opportunities are for effective solutions, all in the context of providing an overview of the current state of research in multimedia multicasting. The issues we discuss include quality of service, resource reservations, routing, error and traffic control, heterogeneity, and the use of hierarchical coding and open-loop control techniques.**

## I. Introduction

One of the ways communication can be characterized is by the number of receivers targeted by a sender. Traditional communication modes have been one-to-one or *unicast*, and one-to-all or *broadcast*. Among these two extremes we find *multicast*, the targeting of a single data stream to a select set of receivers, which may or may not include the sender.[1] Multicast is actually a generalization of both and a unifying communication mode. This model of communication supports applications where data and control are partitioned over multiple actors, such as updates to replicated databases, contacting one of a group of distributed servers, and interprocess communication among cooperating processes.

*Multimedia* applications communicate using a collection of information formats, such as text and graphics, which can be classified as *discrete* or time-independent media, and audio and video, which can be classified as *continuous* or time-dependent media. Sound and moving images are natural forms of communication between humans; thus, support for continuous media is an important goal for enhancing the scope of communications applications. The proliferation of cost-effective audio and video hardware for existing workstations, increasingly provided as standard equipment, along with the availability of high-speed networks providing the necessary bandwidth for continuous-media communications, is expected to establish multimedia applications as standard tools for collaboration.

Since multimedia enhances the abilities of computer-based communication in supporting human collaboration, it should be expected that such applications can greatly benefit from the use of multicast capabilities. A canonical example of such interactions arises in video conferencing ([81], [14]). In addition, groupware ([29]) and computer-supported cooperative-work ([42]) applications, which naturally fit into the multicast model, can be made more effective by incorporating audio and video capabilities.

Efficient multicasting is a fundamental issue for the success of group applications. While in the past it has been regarded as a feature of limited use, often provided only as an afterthought, it is now recognized as a very desirable service ([62], [74]) for emerging high-speed networks. Since a major market for the emerging Broadband Integrated Services Digital Network (B-ISDN) is expected to be selective video distribution ([87], [89]), with selective multicasting taking the place of indiscriminate broadcasting so as to reduce the waste of resources caused by transmitting too many channels to a limited number of receivers, the interaction between multimedia and multicasting is of special importance.

In this paper, we give an overview of the issues arising from the interaction of multicasting and multimedia communications, and we present some proposed solutions in perspective. Since multicasting and multimedia are by themselves important research topics, we do not attempt to cover each in detail. Instead, we focus on the sub-topics whose interactions pose additional problems to the designer. We also show that there are often special-case solutions that are of great importance in this context. The problem at hand can be summarized as the efficient transmission of large volumes of multiple time-correlated delay-sensitive streams of continuous media, to a dynamic group of destinations, using a packet-switched network.

The remainder of the paper is organized as follows: Section II discusses the nature and needs of applications communicating continuous media over a packet network, giving an overview of continuous media characteristics, required network support and, lastly, a presentation of hierarchical coding techniques. Section III presents the general problems related to multicasting in a packet network, such as group dynamics, routing support, dealing with feedback control and hardware support for multicasting. Section IV brings multimedia and multicasting together, discussing host and network heterogeneity, resource reservations, and extending the previous discussions on routing and feedback control in this context. Finally, in Sect. V we give an overview of the protocols that have been designed to support multimedia and multicasting on the Internet.

[1] If the sender must be a member of the group, the group is called *closed*; otherwise, it is *open*. Here we deal with the more general case of open groups.

## II. Continuous-media applications and packet networks

In this section, we first examine the nature of applications using digital audio and video streams and then proceed to show how the characteristics of continuous media impose requirements for the underlying communications network. Traditionally, audio and video were carried by circuit-switched networks with fixed bandwidth allocations and constant transmission delays, while packet-switched networks were developed to accommodate data sources with bursty traffic, ignoring timing issues. The change of paradigm to that of integrated networks, transporting all forms of information using a packet network raises many issues for packet-network technology. We focus on those aspects of continuous media that have implications for multicasting as well.

### A. Audio and video streams

A distinguishing feature of audio and video is the *volume* of data required for a typical stream, especially compared to the text and graphics streams traditionally carried on packet networks. CD-quality audio, uncompressed television-quality video, and HDTV-quality (high-definition television) video, require rates measured in Mb/s (megabits per second), hundreds of Mb/s, and Gb/s, respectively. Compression, particularly for video streams ([43]), can significantly reduce the bandwidth of a combined (television-quality) video and audio stream to 1.5 Mb/s using MPEG-1 ([39]), while for applications with more modest quality requirements, like video-conferencing, the bandwidth can drop to between 64 and 384 Kb/s ([64]). Considering that these applications may require the simultaneous transmission of multiple data streams, the aggregate data rates are high enough to make resource considerations important, even though transmission bandwidth is constantly increasing in new-generation networks and buffer-memory costs are dropping.

The second aspect that is relevant here is that many audio and video applications are *interactive*, in the sense that data reception is interleaved with playback of the associated media streams, rather than playback following reception completion. This implies a requirement for the provision of bounded *delays* between sender and receiver. In fact, for audio and video to be effectively used in these situations, i.e. without forcing the communicating parties to modify their behavior from that of face-to-face communication, such delays are expected to be small. Studies have determined that a certain amount of delay is imperceptible or, at least tolerable by humans; various guidelines set this tolerance to between 40 and 600 milliseconds ([5], [32], [48]).

A problem related to bounding the maximum transmission delay is that of bounding the delay variance, usually called *jitter* in this context. For most continuous-media streams, information must be presented at regular intervals to avoid distracting the human user, thus there is a need for *intra-media synchronization* ([11]). Jitter is usually smoothed out at the receiver by buffering and delaying the playback time of received data. Although this improves playback quality, it increases the total delay experienced at the receiver, a problem for interactive applications. In addition, it increases memory requirements for buffering, which may be a problem ([33]) considering the amount of data involved, even for very short time periods.

While applications expecting nearly real-time interaction, with delays practically imperceptible by humans, are more challenging, non-interactive applications[2] that maintain the characteristic of interleaved reception and playback pose interesting problems. For instance, video distribution based on the "TV" model expects relatively infrequent interactions with the human viewer (e.g. changing the channel); video distribution based on the "VCR" model will include interactions due to user commands to control information flow (e.g. slow motion) and expect them to take effect immediately (by human reaction-time standards). In both cases, synchronization is required between sender and receiver that depends not only on transmission events but also on simultaneous playback events at the receiving end.

To take into account and exploit the different characteristics of continuous-media applications, and more precisely of their media components, we can classify them into two generic categories ([19]):

1) Intolerant or *rigid*, requiring performance guarantees, either absolute or statistical ones, which have to be somehow provided by the network. In return for network provided guarantees, these applications produce constant quality output.
2) Tolerant or *adaptive*, which expect a consistent quality-of-service at any point in time, but can adjust their operation to adapt to changes in the actual quality-of-service provided, presumably in exchange for lower transmission costs. These applications are able to gracefully adapt their output quality depending on the quality-of-service provided by the network.

In the first case, the network has to arrange in advance for providing the required service guarantees under any circumstances, which may only be possible through explicit resource reservations ([103]). In the second case, the applications themselves will dynamically adapt to service degradations ([12]), with the network only undertaking to provide a minimum level of service quality which will improve when traffic conditions allow it to. The non-guaranteed, *best effort* service provided by connectionless internetworks today properly completes the spectrum of possible service models when added to the two discussed above ([19]). Best effort service was designed for time-independent information streams such as file transfer and electronic mail, and will not be further considered in this paper.

### B. Packet-network support

The traditional transmission medium for continuous-media applications has been a circuit-switched network with a fixed-channel data rate, carrying an analog signal with no provision for error control, thus delivering output whose quality varies depending on instantaneous signal information content and channel quality. The first video coders were designed to output a *constant bit rate* (CBR) stream to match this fixed bandwidth,

[2] As perceived by users, since peer applications may interact without user intervention.

disregarding any information redundancy inherent in the encoded media. Digitization of audio and video streams however opens up the possibility of employing compression techniques that minimize bandwidth requirements by exploiting media-intrinsic and human-perception properties ([43]), thus producing a constant quality but *variable bit rate* (VBR) ([72]) digital stream. This makes packet switching the desired mode of network transport due to the advantages of statistical multiplexing which can realize large economies of scale by supporting constant-quality rather than constant-bandwidth service.

Traditional methods for distributing continuous media combine all of the media components (audio and video) in a single composite wide stream. It is possible to mimic this behavior in a packet-switched environment by appending audio data to video frames ([63]). With this approach, synchronization among continuous-media streams, or *inter-media synchronization* ([11]), is greatly facilitated as each frame is a synchronization point. However, this approach prevents us from treating each media component in the most appropriate way during transmission. By transmitting each type of media (and going further, substreams of each media type) independently, not only is media-specific treatment allowed, but there is also the potential for diminishing the effects of communication problems. For instance, congestion can be controlled by employing a protocol supporting priorities (based on the type of information transferred on each stream), that drops the least-important packets, rather than arbitrary ones, thus maximizing perceived reception quality. Such schemes have been proposed for B-ISDN networks using ATM (asynchronous transfer mode) ([28]). Since degradation of video quality can be tolerated to a greater degree than audio degradation, such a congestion-control scheme can choose to drop video packets first. This can be taken further by using hierarchical coding (see Sect. II-C) and similarly prioritizing the media substreams.

The drawback of treating each stream separately is that most packet networks do not explicitly take into account any timing relations among separate transported streams. Thus, additional protocol mechanisms have to be provided to achieve inter-media synchronization before playback and appropriate synchronization information will have to accompany each stream ([78]). Inter-media synchronization requirements can be quite stringent. For example, video and audio of a person speaking (lip synchronization) must be presented within 80 milliseconds of each other to be unnoticeable ([88]). In contrast, synchronization between time-dependent (audio/video) and time-independent (text/graphics) media is considerably less critical, with differences between 250 and 500 milliseconds among streams being acceptable for most applications ([88]). As a result, the delay and jitter caused by packet switching will have to be accordingly bounded for each separate stream. In contrast, with composite media streams, such synchronization overhead is avoided and delay and jitter have to be bounded for a single stream only.

Many techniques commonly used in packet network protocols are based on the assumption that data communication is not especially delay-sensitive but is highly error-sensitive. This assumption does not hold for the real-time, high volume, interactive traffic discussed above. For instance, error-free transmission, considered essential for most existing data communication applications, is commonly implemented by adding error-detection information to transmitted streams and retransmitting any lost or corrupted data. The delay requirements of continuous-media applications may not permit retransmissions, especially for long-delay transcontinental or satellite links. Thus, the issue now is the actual error rate for the end-to-end transport mechanisms employed and how it can best be dealt with. Since audio and video streams are less sensitive to errors than other data streams, with the perceived reception quality depending heavily on the encoding and compression schemes employed as well as on the channel characteristics, one approach for error control is to rely on increasing transmission overhead by including error-correction rather than error-detection information. For continuous-media streams that can withstand limited data loss, it is more efficient to consider the error and delay requirements as *soft* real-time ones, e.g. to provide statistical delay guarantees, such as a given percentile of packets or bits received correctly within a delay bound ([32], [34], [102]). Note that in this case delayed or corrupted data are equally useless to the application, but their loss is not disastrous, in contrast to traditional data streams where loss is to be avoided and delay is irrelevant. Depending on the model that is chosen for defining the application's quality-of-service requirements, several approaches have been proposed for actually guaranteeing them ([60], [102]).

An issue that has been long debated is the choice between connection-oriented and connectionless services. In many packet networks the connectionless approach has been selected for the network layer, with information traveling in datagrams, since this stateless model seems to better fit the bursty nature of the traffic and is also more resilient with respect to network reconfigurations that lead to routing changes. Additional semantics are added, if required, on the transport layer service, by using an appropriate protocol. However, to support the real-time service guarantees that continuous-media applications need, it may be required to reserve resources or make other advance arrangements before media transmission starts. The relatively high cost for such set-up coupled with the long life of such communication sessions in terms of packets transferred seems to fit connection-oriented models better. Naturally, in connection-oriented networks, such advance preparations can be supported directly by the network layer.

In order to provide service guarantees to continuous-media applications, it may be necessary to depart from the best-effort service provided in many networks. This service is a direct consequence of the fact that every packet presented to the network boundaries is admitted without regard to the impact it may have on the congestion and quality-of-service that existing connections are experiencing. To keep the quality-of-service at the level that these applications have bargained for, admission-control mechanisms will have to be introduced that first determine the effects that a new session will have on the existing ones and then admit the new traffic only if the impact is not negative ([19], [92], [28]). Note that routing decisions, resource reservations, and admission decisions are interrelated, since the network may be able to support a request by making appropriate reservations over some routes only. Again, these consid-

erations argue in favor of adopting some form of connection-establishment scheme, where routing may be fixed in advance, along with resource reservations. However, as many existing connectionless networks use dynamic routing, an effective compromise is to maintain *soft* state which is periodically updated to adapt to changing session requirements and network conditions, rather than *hard* state that remains fixed after the initial set-up, as in a static connection scheme.

During data transfer, traffic-policing mechanisms may have to be used to ensure that applications behave as promised with regard to the data traffic they produce ([19], [102]). In addition, traffic-shaping mechanisms can help regulate the data transmission so that sensitive data meet their deadlines without overrunning the receivers' abilities (flow control) and temporary overload situations in the network are dealt with gracefully (congestion control).

### C. Hierarchical coding for continuous media

Hierarchical coding techniques, also referred to as layered or sub-band coding, split a continuous-media signal into components of varying importance ([56], [41]). The original signal may be reconstructed by aggregating all these components, but even proper (but specific) subsets of these components can approximate it well. A simple form of hierarchical coding may decompose a video frame into a low-resolution component containing one quarter of the pixels and a high-resolution component containing the remaining ones. A receiver that only chooses to (or has to) use a presentation window one quarter of the sender's size, may avoid using the high-resolution component, and can reduce its resource needs by discarding it or, even better, by not receiving it. Thus, with hierarchically-coded streams, the receivers can allocate resources based on their own specifications and priorities without interacting with the source. This has important implications when hosts or networks are heterogeneous and even more so when multicasting is employed. For long-term allocations, this may be done in advance so that the sender can avoid sending the extraneous streams. Temporary resource shortages, whether memory or processing ones, can be dealt with by ignoring some streams, without any explicit negotiations with the sender, and by dynamically degrading the quality of the presented signal. The receiver may even manipulate these streams before presentation in ways not anticipated by the sender ([65]).

Hierarchical coding can be exploited to the benefit of the network infrastructure itself. For high-speed networks, significant congestion-control problems may arise due to the statistical multiplexing of highly bursty signals ([92], [28]). Solutions that avoid reserving resources at peak transmission rates depend on shedding load quickly by dropping some traffic without causing avalanches of retransmissions. With hierarchically-coded continuous media, the less important signal components, as determined by the applications, can be dropped to relieve congestion without causing retransmissions, leading to degradations in quality-of-service but not service interruption. Many proposed congestion-control techniques rely on this feature ([28]) as a last resort. Another relevant aspect of hierarchical coding is that in some schemes the basic low-resolution layers that are essential for signal continuity are highly compressible, thus suggesting a strategy of transmitting these streams with stricter guarantees than the ones for the remaining streams.

The benefits derived from the independence of the streams provided by hierarchical coding must be weighed against two factors. First, separate compression of parts of the signals can be less efficient (in terms of space reduction) than compression of the complete signal. Second, there are performance penalties for splitting the signal into components and later reconstructing it, since hardware and software support is still inadequate for this purpose. Since some international compression standards, such as JPEG ([98]) and MPEG-2 ([93]), support hierarchical coding, this problem may be less important in the future as vendors are pressed to upgrade their system software.[3]

## III. Multicast support in packet networks

While multicasting is increasingly recognized as a valuable service for packet networks, many existing architectures still do not support it directly. A variety of methods for providing such a service are under examination, from both theoretical and experimental standpoints. In this section we look at the main issues in multicasting such as group dynamics, routing support and feedback control, and discuss some of the proposed approaches. Since space does not permit a more detailed exposition of all issues relevant to multicasting, the reader can also refer to specialized surveys ([38]), research compilations ([2]), or start from detailed bibliographies ([16]). Here we try to limit our scope to the issues that are most relevant for continuous-media communications.

Support for multicasting may exist at the physical and data-link layers. Many shared-medium networks such as Ethernet support the option of broadcasting and multicasting packets, and support the corresponding addressing mechanisms. However, processors are often required to perform extra processing when receiving a multicast packet. For example, many Ethernet controllers only support partial-address filtering ([51]). When switches are used in point-to-point networks, we would like the hardware to automatically recognize multicast addresses as such and transmit multicast packets through multiple links copying packets on the fly, as required. ATM switch designs ([95], [104]) increasingly support parallel transmission of multicast cells over multiple links in hardware, increasing peak switching speeds. The emphasis placed on high-speed network switches to support multicasting ([50], [30], [61], [96]) could influence more conventional hardware.

### A. Multicast groups and their dynamics

The difference between multicasting and separately unicasting data to several destinations is best captured by the *host group* model: a host group is a set of network entities sharing a common identifying multicast address, all receiving (traditionally, via best-effort service) any data packets addressed to this multicast address by senders that may or may not be members of the group and have no knowledge of the group's membership ([17]). This definition implies that the behavior of the group

[3]Even though MPEG-1 ([39]) was not designed with hierarchical coding in mind, MPEG-1 streams can be split in prioritized components and treated accordingly ([73], [3]).

over time is unrestricted in multiple dimensions; it may have local (LAN) or global (WAN) membership, be transient or persistent in time, and have constant or varying membership. From the sender's point of view, this model reduces the multicast service interface to a unicast one. This implies that the network software is accordingly burdened with the task of managing the multicasts in a manner transparent to the users. From the network designer's point of view, this extra work is expected to result in a more efficient usage of resources. This is the primary motive for network providers to support multicasting in the first place.

These goals for multicast service impose specific requirements for the network implementation. First, there must be a means for routing packets from a sender to all group members whenever the destination address of a packet is a multicast one, which implies that the network must locate all members of the relevant group and make routing arrangements. Second, since group membership is dynamic, the network must also continuously track current membership during a session's lifetime, which can range from a short to a very long period of time. Tracking is required both to start forwarding data to new group members and for stopping the wasteful transmission of packets to destinations that have left the group. Both tasks must be carried out without assistance from the sending entity as defined by the host-group model. The dynamic nature of multicast groups has important implications for multicast routing.

Another set of issues is concerned with extending the feedback mechanisms employed by unicast-oriented protocols to deal with flow, congestion and error control. Transport-layer protocols such as TCP (see Sect. V) adapt their behavior according to the prevailing network conditions at any given point in time by measuring loss rates as experienced by receivers, especially in networks based on best-effort service. Transport-layer protocol behavior is based on end-to-end exchanges of reports that describe reception statistics; these are then used by the sending side to estimate the network conditions and modify its behavior accordingly. When extending these protocols for multicasting, there is the possibility of *feedback implosion* when many receivers send such reports towards the sender ([20]), thus swamping the network and the source with control information. Apart from the obvious scalability problems of such schemes, there is also the issue of how to adapt the sender's behavior when conflicting reports arrive from the various receivers and how to deal with a changing receiver population in a dynamic environment.

### *B. Multicast routing*

The basic means of conserving resources via multicasting is sharing: instead of transmitting packets from a sender to each receiver separately, we can arrange for routes that share some links to carry each packet only once. 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. The tree can be designed so as to maximize shared links and thus minimize resource consumption. An interesting problem arising from resource sharing in multicasting is how to split the total distribution costs among the receivers, or how to allocate the savings compared to using separate unicasts ([49]). This issue is orthogonal to the problem of what the total costs are, a question also arising in the unicast case. Whether these costs are used for pricing or for informational purposes, they are a primary incentive to use multicasting.

Traditional unicast-routing decisions intend to minimize transmission costs or delay (depending on the interpretation of the metric used), using shortest-path algorithms, with the Dijkstra ([26]) and Bellman-Ford ([7], [37]) algorithms being two common cases. These algorithms find optimal routes between one node (the sender) and *all* other nodes in the network (including all receivers). Thus, a straightforward solution to the multicast routing problem can be based on the *shortest path trees* produced by these algorithms and pruning off any branches that do not lead to any receivers in the group.

In Fig. 1, we show a multicast tree within a sample graph taken from ([58]). The edge labels denote (cost,delay). The tree is obtained by combining the separate optimal *cost* paths from the sender to each of the receivers. The total cost is $21$, the average delay is $2.75$ and the maximum delay is $4$ units. In Fig. 2, the multicast tree is obtained by combining the separate optimal *delay* paths from the sender to each of the receivers. The total cost is $32$, the average delay is $2$ and the maximum delay is $3$ units.

Although details vary according to the base algorithm ([71], [22]), there are some observations that generally apply. On the up side, these algorithms are easy to implement, as direct extensions of existing ones, and thus fast to deploy. Additionally, each path is optimal by definition, regardless of changes in group membership, and this optimality comes essentially for free since shortest paths need to be computed for unicast routing as well. On the down side, these algorithms suffer from poor scalability. Even though they are not affected by membership dynamics, as paths are pairwise independent, they are affected by network dynamics, i.e. link failures and network reconfigurations that can cause them to frequently repeat routing calculations. For large internetworks with widely dispersed groups, either the scale of the network or continuous network changes will restrict use of these algorithms to subnetworks that already use their unicasting counterparts, with a higher-level routing protocol ([90]) forwarding data among these subnetworks.[4]

Cost optimization in multicasting can be viewed from another angle: overall cost optimization for the distribution tree. The shortest path algorithms concentrate on pairwise optimizations between the source and each destination and only conserve resources as a side effect, when paths converge. We can instead try to build a tree that exploits link sharing as much as possible, and by duplicating packets only when paths diverge, minimize total distribution cost, even at the expense of serving some receivers over longer paths. What we need is a tree that reaches all receivers and may use any additional network nodes on the way. This is equivalent to the *Steiner tree* problem, where a cost-labeled graph and a set of nodes, the *Steiner points*, are given and we want a minimal-cost tree connecting all Steiner points, consisting of the sender and the receivers ([45]). In Fig. 3, we see one of the many possible *Steiner* multicast

[4] Similar problems (e.g. processing complexity for Dijkstra and instability for Bellman-Ford) have also forced unicast-routing algorithms to rely on *hierarchical routing* techniques for large networks.

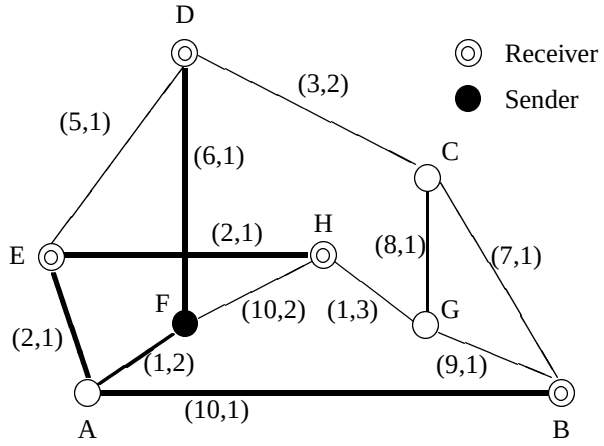


Fig. 1. The multicast tree obtained by combining the separate optimal *cost* paths from the sender to each of the receivers

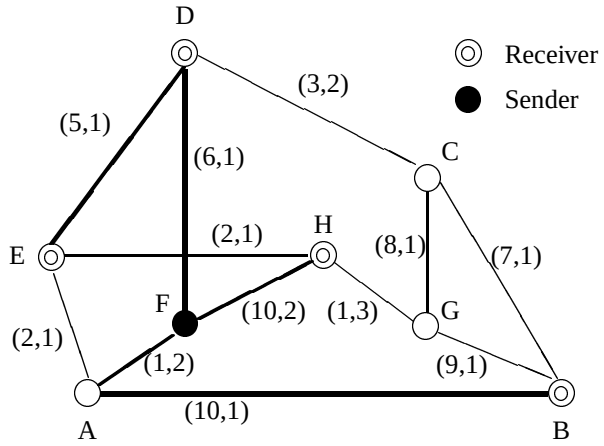


Fig. 2. The multicast tree obtained by combining the separate optimal *delay* paths from the sender to each of the receivers

trees for this graph. For this tree, the total cost is $20$, the average delay is $4.75$ and the maximum delay is $8$ units. Note how simple cost minimization can lead to significant increases in delay.

Despite the fact that this problem is NP-complete ([40]), approximation algorithms exist with proven constant worst-case bounds ([80], [59]). Implementations of such algorithms have been shown to produce low-cost multicast trees with very good average behavior ([97], [99], [55]). As an example, trees built with the KMB heuristic ([59]) have at most twice the cost of Steiner trees, while simulations of realistic network topologies have shown their cost to be within 5% of the optimum ([27]). The advantage of this approach is its overall optimality with respect to a single cost metric, such as transmission cost. However, the disadvantages are also important: the algorithm needs to run in addition to the unicast algorithms, and it will itself have scaling problems for large networks. Furthermore, optimality is generally lost after group membership changes and network reconfigurations if the tree is not recomputed from scratch. Thus, Steiner tree algorithms are best suited to static or slowly changing environments since changes lead to expensive recalculations to regain optimality.

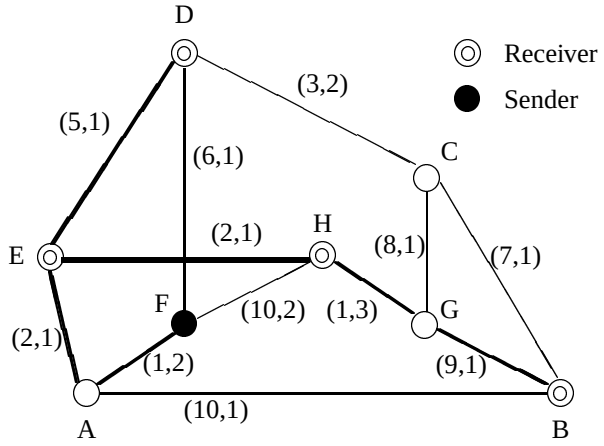


Fig. 3. A *Steiner* multicast tree for this graph

Both approaches discussed above suffer from an inability to maintain their measure of optimality in a large and dynamic network. Approaches for extending these algorithms to deal with changes in group membership without complete tree reconfigurations include extending an existing tree in the cheapest way possible to support a new group member and pruning the redundant branches of the tree when a group member departs ([1], [99]). The quality of the trees after several local modifications of this sort will deteriorate over time, eventually leading to a need for global tree reconfiguration.

A different approach to the routing problem opts for a solution in realistic settings by adopting the practical goal of finding good rather than optimal trees that can also be easily maintained. The departure point for this approach is the *center-based tree* ([97]), which is an optimal-cost tree that, instead of being rooted at the sender, is rooted at the topological center of the receivers. Even though such a tree may not be optimal for any one sender, it can be proven to be an adequate approximation for all of them together. The implication is that one basic tree can serve as a common infrastructure for all senders. Thus, maintenance of the tree is greatly simplified and nodes on the tree need only maintain state for one *shared* tree rather than many source-rooted trees. Since this method has been developed for broadcasting rather than multicasting, the theoretical investigation ([97]) does not hold when we prune the broadcast trees to get multicast ones. In addition, the topological center of the tree, apart from being hard to find[5] will not even be of use in a dynamic multicasting environment.

Practical proposals for multicast routing abandon the concrete optimality claims discussed above, but keep the basic idea of having a single shared multicast tree for all senders to a group. This is a departure from approaches that build one tree for each sender. Routing is then performed by defining one or more *core* ([6]) or *rendez-vous* ([23]) points to serve as the basis for tree construction and adding branches by separately routing packets optimally (in the unicast sense) from the senders to these fixed points and then from there to the receivers. Again, merging of paths is exploited whenever possible, but it is not an explicit goal of the routing calculations. Instead, because of the

[5]The problem is NP-complete ([6]).

concentration of paths around the fixed points, common paths are expected to arise.

As an example, in Fig. 4 we show a *core based tree* ([6]), with node $H$ as the core and optimal *delay* paths from the core to each receiver (this is a *member-centered optimal-delay* tree). All highlighted links except $FH$ are part of a common tree shared by all possible senders. Sender $F$ unicasts its data towards the core again via an optimal *delay* path, using link $FH$. The total cost is 29. The average delay from $F$ is 3.5 and the maximum delay is 5. Using again $H$ as the core, if we try to minimize *cost*, the resulting core tree is identical to the tree in Fig. 3 (this is a *member-centered cost-optimal* tree). All highlighted links except $FA$ and $AE$ in that figure are shared among all possible senders, and $F$ unicasts its data towards the core via an optimal *cost* path, using links $FA$ and $AE$. The delay metrics from $F$ are the same with those given in Fig. 3 as packets do not have to reach the core to be re-routed: multicasting starts when the path from the sender first encounters the shared tree. In that figure, for sender $F$ this point is node $A$.

A single shared multicast tree is not optimal in any strict sense[6] since no attempt is made to find the topological center of the tree, both due to its computational cost and the limited lifetime of any topological center for a dynamic environment. But, the advantages of shared multicast trees are numerous. First, a shared tree for the whole group means that this approach scales well in terms of maintenance costs as the number of senders increases.[7] Second, the trees can be made quite efficient by clever choice of the fixed points. Third, routing is performed independently for each sender and receiver, with entering and departing receivers influencing only their own path to the fixed points of the single shared tree, employing any underlying mechanism available for unicast routing. This last property means that network and group membership dynamics can be dealt with without global recalculations and by using available mechanisms.

In practice, these multicast algorithms are expected to use the underlying unicast algorithms, but are independent of them. Interoperability with different unicast schemes, coupled with the scalability of the shared trees, make these algorithms ideal for use on very large-scale heterogeneous networks. The fixed points can also be selected so as to facilitate *hierarchical* routing for very large internetworks, further enhancing scalability properties.

### C. Feedback control

Whether a network provides a simple connectionless service or a complicated connection-oriented unicast service, generalizing it for multicasting is not trivial. Flow, congestion, and error control depend on feedback to the sender, according to network and receiver-triggered events. For simple network services, no such information is provided by the network itself, but instead higher-layer protocols must exchange end-to-end reports. With more complex network-service models, some of these problems, such as error control, may be dealt with internally in the network, while other problems, such as flow control, are usually left to higher-layer protocols that are better equipped to deal with them.

[6] We examine the costs and tradeoffs in Sect. IV-C.

[7] Actually, there is still a tree emanating from each sender, but all these trees merge near the fixed points and the distribution mesh is common from there on to the receivers.

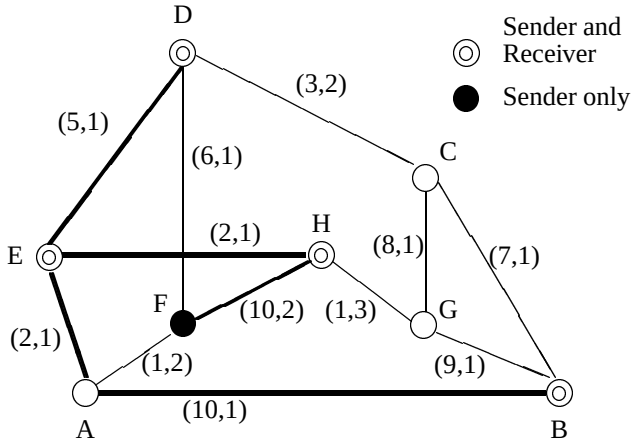


Fig. 4. A *core-based tree* with node $H$ as the core and optimal *delay* paths from the core to each receiver (this is a *member-centered optimal-delay* tree)

*Error control* ensures that packets transmitted by the sender are received correctly at the other end. Packets may be received corrupted (detected by error-detection codes) or they may be lost (detected by missing packet-sequence numbers). *Flow control* assures that the sender does not swamp the receiver with data that cannot be consumed in time. *Congestion control* deals again with the problem of insufficient resources, but this time on the network switches between sender and receiver. Although packets are dropped when they cannot be processed in an intermediate node, in many networks this loss can be detected only by the receiver, resulting in confusion between errors and congestion (lost versus dropped packets). While flow control is clearly an end-to-end issue, error control and congestion control depend on network status and thus they can be dealt with either at the network layer or addressed at the higher layers on an end-to-end basis.

In the unicast case, lost or corrupted packets are retransmitted based on feedback received from the network or the receiver. When packets are multicast, simple feedback schemes face the feedback-implosion problem ([20]): all receivers respond with status information, swamping the sender with possibly conflicting reports. Ideally, senders would like to deal with the multicast group as a whole and not on an individual receiver basis, following the host-group model. However, the sender cannot simply treat all receivers identically, because this would lead to either ignoring the retransmission requests of some receivers, or to wasting resources by retransmitting to all of them.

Since there is no evident solution that satisfies all requirements, several approaches exist emphasizing different goals. The simplest approach of all is to ignore the problem at the network layer and provide a best-effort connectionless service. Delegating the resolution of transmission problems to the higher layers may be an adequate solution in many cases, since they may have additional information about the application requirements and thus can implement more appropriate mechanisms than what is possible at this layer. In Sect. IV-D we discuss how this applies specifically to continuous-media applications.

A second solution sacrifices the host-group model's simplicity by keeping per-receiver state during multicasts. After transmitting a multicast packet, the sender waits until a stable state is reached before sending the next one. For flow control, this slows down the sender enough so as not to swamp the slowest receiver. For error control, retransmissions are made until all receivers receive the data. This may not be possible even after multiple retransmissions, so the sender may have to treat some receivers as special, i.e. by removing them from the group. Retransmissions may be multicast when many receivers lose the initial packet, or unicast when few do. Since feedback implosion is always a possibility, all such schemes should use *negative* rather than *positive acknowledgments*, i.e. send responses when problems occur rather than confirming that packets are received correctly and in time. In a negative acknowledgment scheme, some responsibilities are moved to the receivers, complicating their operation. However, additional opportunities arise, such as multicasting the negative acknowledgements to all receivers after random periods of time to minimize the number of negative acknowledgements returned to the sender ([79]).

Nevertheless, distributing such overhead to all receivers rather than performing everything at the sender can lead to higher throughput rates. However, the scalability of such schemes is doubtful, even for very reliable links and rare congestion or overflow problems. The problem is that the sender is still the control center, and as the number of group members grows, receivers and network paths become more heterogeneous. With these essentially symmetric 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 network link. Sophisticated approaches exist that follow these general directions ([66], [15]), but their complexity and inefficiency makes them appropriate only for applications that require very high reliability and uniform member treatment ([68]). Note that such reliable solutions can be implemented as transport services ([82]) over a simple connectionless network service ([4]).

A third solution is to distribute the feedback control mechanism over the entire multicast tree, and follow a hierarchical scheme. A receiver's feedback need not propagate all the way to the sender. Instead, intermediate nodes may either respond directly or merge the feedback from many downstream receivers to a summary message and then recursively propagate it upwards. In this case, feedback implosion is avoided in terms of messages, but the problem of dealing with possibly conflicting requests remains. If the added complexity of making local decisions on each network node (not only group members) is acceptable, we can narrow down the impact of problems to specific parts of the tree, relieving the sender from dealing with individual receivers. Since the current trend in networks is to minimize node complexity, it seems that the hierarchical approach will be used either for simple protocols that only summarize information forwarded to the sender,[8] or for more complex protocols that are activated only infrequently during a session ([103]).

A non-hierarchical method for distributed feedback control targeted to recovery of lost messages is to let all receivers and senders cooperate in handling losses ([36]), thus extending the sender-oriented model ([79]). When receivers discover a loss, they multicast a retransmission request, and *anyone* that has that message can multicast it again. 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 source of the request respectively, plus a (bounded) randomized delay. The result is that most duplicate requests and replies are suppressed by the reception of the first multicasts. By varying the random-delay intervals, the desired balance among recovery delay and duplicates can be achieved. In contrast to hierarchical schemes, since location-independent multicasts are used, only group members participate but recovery cannot be localized without additional mechanisms.

A fourth solution 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 simple error-detection codes ([85], [9]). For flow and congestion control, this is achieved by reserving resources just before transmission starts ([103]) so that both receivers and intermediate network nodes are able to support the sender's data rate. The motivation for these approaches is that they are better than doing nothing, but simple enough to be implemented efficiently. FEC only requires some processing and transmission overhead and no additional mechanisms in the network ([3]). Resource reservation on the other hand needs additional control mechanisms to set up a session (and often, to maintain it during the session's lifetime). However, these costs will be a small fraction of the total costs (in the case of FEC) and only infrequently incurred (in the case of reservations), compared to other complex feedback mechanisms. The combination of error correction and reservations may be an adequate solution to the expected problems created by the statistical multiplexing of highly-bursty high-bandwidth signals, even though these mechanisms are not expected to be both completely reliable and efficient in resource usage.

## IV. Multicasting continuous media

Having considered the problems of continuous media and multicasting separately, in this section we examine the issues arising from their combination. Here, we emphasize issues such as dealing with heterogeneity and resource reservations that become particularly important in this context. In addition, other issues, such as routing and feedback control, take on new dimensions when multimedia and multicasting are combined, with both novel problems and special-case solutions arising.

### *A. Host and network heterogeneity*

Heterogeneity in WANs has been an issue for a long time: best-effort services are one result of supporting the lowest common denominator among widely different network switches and links. For continuous media we would like to offer enhanced end-to-end services, for example guaranteed delay and jitter bounds, by employing whatever mechanisms are available at each switch. Since it is questionable whether the available mechanisms will converge in the future, a more viable approach

[8] Of course, these protocols must still address the problem of how to summarize the feedback in a meaningful way.

is to *substitute* a requested service by an existing one that approximates it as closely as possible ([86]). The end-to-end service resulting from appropriate combinations of the heterogeneous services offered along a distribution path can be a much more satisfactory approximation to the required service than what would result from using a common but inferior local service everywhere. Enhanced local services may be able not only to provide local guarantees, but also to offset the shortcomings (inadequate or nonexistent guarantees) of local services in other path links ([86]).

With multicasting, heterogeneity problems are aggravated as simple paths are turned into trees and, in addition to switches, hosts within a group can also differ. Since continuous media imposes heavy demands on both networks and hosts, it is likely that not everyone will be able to receive all of a sender's traffic due to link, switch, or host limitations. This argues in favor of hierarchical-coding approaches so that receivers can choose to get only those parts of the media that they can use ([84], [78]). This approach to dealing with heterogeneity can, in many cases, be an adequate solution for the problems posed by *closed-loop* feedback-based flow and congestion control, since it matches FEC in being *open-loop*, and as such it does not require continuous feedback from the receivers, nor any extra actions on the part of the sender.

Experience has shown that several representational formats for each media type can coexist ([54]). This is already a problem with traditional data communications, but it is more of an issue with images, audio and video. Such problems are typically addressed at the presentation layer, which can provide translation services at three points: at the transmitter, at the receiver, or inside the network. In the latter case, format converters are required to be deployed inside the network. This may be an adequate approach for converting protocols or text encodings among largely independent network areas by placing the converters in the gateways, but it is not suitable when the terminals themselves can use different encodings within the same area, where translation capabilities will be required at practically every switch. Thus, it is more realistic to move the translation procedures in the hosts themselves. In unicasting, translation can be effectively done at either the sender or at the receiver.

In contrast, with multicasting, translation at the sender requires the stream to be duplicated and translated for each different type of receiver, precluding link sharing over common paths. This approach also does not scale for large heterogeneous groups since the sender's resources are limited. Last, it requires the sender to be aware of the receiver's capabilities, which is incompatible with the host-group model.[9] Translation at the receiver is the most economical and scalable approach since it fully exploits sharing and moves all responsibilities away from the sender. Note that appropriate hierarchical coding can be easily combined with, and probably facilitates, translation and reconstruction of the signal at the receivers, according to their needs and abilities.

[9] The sender may use different multicast groups for each encoding to avoid this, but the other problems remain.

### *B. Resource reservations*

Resource reservation can occur at the receivers or at the network switches. Since failure to make adequate resource reservations at the receiver, as required by the sender's traffic profile, simply leads to receiver overflow, such reservations are not a direct concern of the network service. However, in this case flow-control mechanisms may have to be employed, which are difficult to scale for multicasting, and even more so when continuous media is transmitted. We would expect then the receivers to allocate sufficient resources in advance in order to avoid or minimize flow-control problems.

On the other hand, resource reservations at the network switches will be needed if any service guarantees are to be provided. The exact nature of these reservations will differ according to the required service guarantees and the approach taken towards satisfying them ([60]), so resource reservation along transmission paths could be viewed as a subset of general switch-state establishment mechanisms ([103]). These mechanisms are employed during connection establishment along with admission control to check for admissibility, and if possible, set up a transmission path. They can also be used during transmission to alter a connection's reservations. An alternative to reserving resources for an indefinite period of time during connection establishment is to make *advance* reservations for a future connection with a given lifetime ([35]). This allows more sessions to be admitted (due to their deterministic timing) and also permits negative responses for reservation requests to be dealt with more gracefully.

We are narrowing the general problem of switch-state establishment to resource reservations here, since for the high bandwidth and low delay needs of continuous media, transmission speed, switch capacity, and buffer memory will probably all be in short supply. Adaptive schemes have been proposed to be used in place of resource reservations ([12]) to solve the congestion problems that occasionally arise from statistical multiplexing in a packet network. These schemes are very well suited to the needs of adaptive applications, such as low-cost video conferencing, but they are inherently inadequate for rigid applications that require more stringent guarantees, such as commercial video distribution. Thus, adaptive schemes should be viewed as either supplementary solutions to resource reservations, used to increase efficiency by requesting more relaxed guarantees for appropriate applications ([19]), or as interim solutions for enabling the use of adaptive applications in current networks that do not support resource reservations.

The first component of resource reservation schemes is a specification model for describing flow characteristics ([75]), that depends heavily on the model of service guarantees supported by the network. Then, an appropriate protocol is required to communicate these specifications to the receivers and reserve resources on the transmission path so that the service parameters requested can be supported ([34], [103]). Simple unicast approaches to resource reservations are generally source-based. A set-up 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 over-allocated early on in the path, so that even if switches encountered further along the path are short of

resources, the connection can still be set up. After the set-up message reaches its destination, assuming the connection can be admitted along the path, a response message is returned on the reverse path, allowing the intermediate switches to relax commitments in some cases.

Similarly, for multicasting, there must be a way for senders to notify receivers of their properties, so that appropriate reservations can be made. In a perfectly homogeneous environment, the reservations will be made once on each outgoing link of a switch, for all downstream receivers, so that resource usage can be minimized.[10] However, receiver and network heterogeneity prohibits use of this simplistic scheme, since each receiver and path may be able, or willing, to commit different amounts of resources. One approach is to allocate resources as before during the first message's trip and then have all receivers send back their relaxation (or rejection) messages ([8]). Each switch that acts as a junction will only propagate towards the source the most restrictive relaxation among all those received. However, since paths from such junctions towards receivers may have committed more resources than are now needed, additional passes will be required for convergence, or resources will be wasted. To handle dynamic groups without constant source intervention, this model can be augmented with receiver-initiated reservations that propagate towards an already established distribution tree ([8]).

An alternative approach is to abandon reservations during the sender's multicast set-up message and instead reserve resources based on the modified specifications with which the receivers respond to the initial message ([103]). Again, resource reservations will be merged on junction points, but since the (now upstream) requests are expected to be heterogeneous, each junction will reserve adequate resources for the most demanding receivers and reuse them to support the less demanding ones. Even though it is still unclear how aggregation of reservations should be performed, this approach has the potential to support both heterogeneous requests and resource conservation, possibly without over-committing resources, thus maximizing the possibility for a new session to be admitted. Since this mechanism converges in one rather than in multiple passes, the reservation state in the switches can be periodically refreshed, turning the fixed *hard* state of a static connection into adaptive *soft* state suitable for a dynamic environment. In this way this mechanism can accommodate both group membership changes and routing modifications without involving the sender.

An important issue when providing guaranteed services is how local reservations affect the end-to-end offered service and vice versa. For example, local bandwidth reservations result in an end-to-end bandwidth guarantee equal to the lowest bandwidth guarantee in the path. In contrast, with local delay guarantees it is harder to compute end-to-end delay bounds, especially if we want a tight practical bound rather than a loose theoretical result ([86], [60]).

In a source-initiated approach, a two-pass mechanism is used, with over-allocations in the first pass and relaxations in the second, so that after the second pass the end-to-end service is efficiently mapped to local reservations. During the first pass there is no way to predict what lies ahead, so as many resources as possible must be allocated to maximize the probability that the end-to-end path will be acceptable. However, over-allocated resources will be wasted and other reservation attempts may be blocked until reservations are relaxed, possibly after several passes when multicasting is used.

In the receiver-initiated approach, there is only a single pass, so only local reservations can be specified, which result in an end-to-end service whose characteristics cannot be predicted in advance. As a result, depending on local decisions, the end-to-end reservations may be either inadequate or superfluous. A solution to this problem is to use *advertising* ([86]). In this approach, source-initiated messages gather information about local links as they travel towards the receivers. The receivers can then decide how local reservations along these paths would affect end-to-end services and then use the receiver-initiated approach to reserve the appropriate per-link resources. By including in the information gathered in the first pass the exact services and resources available at each link, the receivers can compose heterogeneous local services and link specific reservations to build enhanced end-to-end paths.

The interaction of routing and resource reservations (and therefore, admission control) further complicates matters. Even in the simple case of static routing, success in building a multicast tree depends on the adequacy of resources on each switch. We would like to construct the tree using the switches that pass the admissibility tests, thus favoring the sender-initiated reservation approach. On the other hand, we do not want the construction to fail due to over-allocation, so receiver-initiated reservations are preferable because they may avoid over-committing resources and converge in one pass. Now however, the tree constructed by the routing algorithm may be inadequate to support the reservations, again rejecting a session that could in principle be set up. We will see later that such problems are even more pronounced when routing is dynamic, as in many wide-area internetworks, or when group membership changes, resulting in modified distribution trees even with static routing.

### C. Extending multicast routing

Our motivation for routing multicast traffic along trees rather than along arbitrary paths is to minimize transmission cost through link sharing. For continuous media, the volume of data transferred makes this goal even more important. However, for real-time multimedia applications we must take into account two additional factors: delay constraints, particularly for interactive applications, and media heterogeneity. As we have already discussed, separate handling of media streams is useful if we want to use the most effective coding techniques for each stream, and in order to gain maximum benefits from hierarchical coding. The question arises then whether we should use the same or separate distribution trees for each stream. Considering the load that continuous media puts on network links and the interaction among admission control and routing, it would seem better to use a separate tree for each media type. Thus, each media stream (or sub-stream) could ask for the appropriate

[10] Reserved resources can also be shared among data transmitted from multiple senders to the same group, in applications such as conferencing where the number of simultaneous senders is much smaller than their total population ([103], [44]).

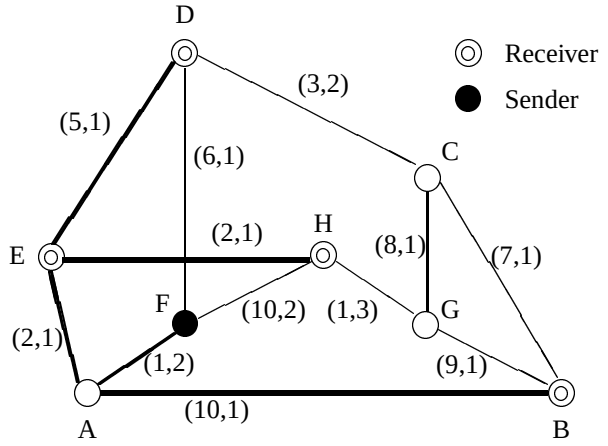


Fig. 5. A *Steiner* tree with bounded maximum delay

quality-of-service parameters and get routed accordingly, and receivers would choose to connect to any subset of the trees. On the other hand, the management overhead of multiple trees per source may be prohibitive. In addition, routing each media stream separately will cause inter-media synchronization problems if appropriate guarantees are not provided for each stream.

Turning to delay requirements, if we use delay as the link metric during routing, we can easily see that the shortest delay tree, made up from the shortest paths from sender to each receiver, is not the same as the tree of total minimal cost that maximizes link sharing at the expense of individual path delays. We have then a global tree metric (tree cost) and many individual receiver-oriented metrics (path delays) that are potentially in conflict. Since we cannot hope to optimize on all fronts, we can try to optimize cost subject to the constraint that delay is tolerable. As we have seen, interactive applications can be characterized by upper bounds on end-to-end delay and/or limits on jitter. In this sense, it is reasonable to design the tree so as to optimize total cost while keeping individual paths within their respective bounds.[11]

This new problem is essentially a version of the Steiner tree problem with additional constraints on the paths. Even though it is NP-complete, fast heuristic algorithms ([57], [58]) that are nearly optimal on the average have been developed. A similar formulation can be used when the constraint is link capacities which must not be exceeded, instead of a delay bound. Again, heuristics exist to solve this variant of the problem ([53]). Fig. 5) shows a *Steiner* tree with bounded maximum delay. The total cost is $20$, the average delay is $3.5$ and the (bounded) maximum delay is $4$. Compare this tree with the one in Fig. 3.

Group dynamics are an obstacle in maintaining optimality, whatever the method of constructing the initial trees. Since repeating all routing computations whenever members join or leave the group may be prohibitively expensive, an alternative is to prune extraneous links when a member leaves the group, and add the most economical (and admissible, if delay bounds have been specified as above) extension path towards a new member, either from a fixed or from the optimal location in the existing group ([10]). Rather than making modifications blindly, thus deteriorating tree quality up to the point of needing complete tree redesign, the most advanced algorithms store some of the state accumulated during tree construction and make only local calculations that still satisfy the requirements of the application. However, simulations have shown that even simple multicast routing ([27]) using the shortest path tree is not significantly worse in terms of total tree cost from the optimal solutions or the near-optimal heuristics. For realistic network topologies the cost of a shortest path tree is less than 50% larger than that of a near-optimal heuristic tree ([27]), while path delays for heuristic trees are 30% to 70% larger than shortest path delays ([100]). Since shortest path trees are easily built and modified using the underlying unicast routing and they never deteriorate in terms of delay, but simply vary in their inefficiency in terms of total cost, if an application is prepared to accept a moderate cost overhead, it can avoid special multicast-tree construction and maintenance methods by employing the shortest delay paths.

A similar cost versus simplicity trade-off is involved when using shared trees for all senders to a group ([6]). But in this case, both total cost and path delays will suffer. The overhead under extreme conditions has been determined theoretically ([97]), but in practice, simulations have been used to determine the average performance of such schemes in terms of cost and delay sub-optimality ([100]). For shared trees, optimality is hard to achieve and even harder to maintain, as discussed earlier, but a simple approach is to choose the center among group members in a way so that only as many trees as group members will have to be considered. For these trees, when path delay is optimized, simulations show that delays are close to 20% larger than the shortest paths, and tree cost is about 10% lower than that of shortest path trees ([100]), which as we saw above is typically within 50% of the optimum, thus striking a balance between cost and delay inefficiency. Furthermore, a single tree constructed using the underlying unicast routing mechanisms minimizes state and maintenance overhead. Unfortunately, apart from their moderate sub-optimality, shared trees also suffer from *traffic concentration*, since they route data from all senders through the same links. 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 ([100]).

For these reasons, recent proposals ([23]) try to combine shared trees and shortest paths by starting each group connection in the shared tree mode and then changing individual paths to shortest delay ones upon receiver requests.[12] The overhead versus delay trade-offs and the point at which change from one paradigm to the other should occur can be determined experimentally ([101]).

A final point regarding routing is dealing with network dynamics. Unicast routing in most internetworks is dynamic in order to guide packets around hosts and links that are down and to adapt to variations in network load. Even though this is sufficient for best effort service, it creates problems when routes are carefully planned to support quality-of-service guarantees.

[11] Normally, all receivers would be satisfied by the same limits, as these are determined by human perception properties.

[12] Actually, this proposal also supports traditional source-rooted trees for the applications that need them.

With multicasting, the problem is even more complicated as resource reservations are shared and group dynamics interact with network reconfigurations. Little is known as to how to deal with such problems, but a conservative approach would be to make the multicast routing algorithm ignore unnecessary routing changes, i.e. changes that reduce route costs but are not due to failed links or switches. The rationale is that load variations should not significantly affect streams for which resource reservations have already been made, but should instead be dealt with by rerouting best-effort traffic. This approach, called *route pinning*, preserves the quality guarantees as far as a path remains physically connected. Note that multicast algorithms using the underlying unicast-routing mechanisms need to ignore dynamic route optimizations in order to implement route pinning.

### *D. Feedback control revisited*

We have seen the problems associated with extending error, flow, and congestion control methods to the multicast paradigm, due both to the possibility of feedback implosion and to the inability of the sender to deal uniformly with conflicting feedback reports. For error control there is the additional problem of the delay associated with *automatic repeat request* (ARQ) schemes which retransmit corrupted packets according to receiver-generated feedback. There is no point for such retransmissions if the deadline for using the corresponding data has already expired. Since ARQ schemes introduce additional delay and jitter to a media stream, additional buffering would be required at the receiver to smoothen out its impact. Due to the bandwidth of continuous-media streams and the strict delay and jitter bounds of interactive applications, retransmissions will not be viable for most applications. Fortunately, many continuous-media applications can tolerate moderate error rates without significant quality degradation, with the degree of tolerance depending both on the nature of the application and the media coding/compression method used. While rigid applications may only be resilient to sporadic problems that are masked by the nature of the media, adaptive applications may also dynamically adapt to long-term quality-of-service modifications.

It is possible to avoid using feedback and retransmissions to control error rates by employing error concealment techniques, such as computing a likely value for missing pieces of the data through interpolation from neighboring values, or forward error correction (FEC) schemes. Planning in advance for error recovery may involve embedding extra information on the transmitted streams, making packets self-contained, transmitting critical packets more than once, and so on. As encoding techniques try to minimize redundancy, a balance should be sought between redundancy for error correction and compression ratios, according to the expected channel characteristics and the application's error tolerance.

As an example of the possible tradeoffs involved in error control, video frames can be coded in *intraframe* mode, where spatial redundancy is exploited only, and in *interframe* mode, where temporal redundancy is also exploited ([39]). Using both techniques increases the compression ratio, but lost packets, while causing spot problems on one frame in intraframe mode (which may be concealed through interpolation), create longer-term problems in interframe mode. To limit this problem with interframe coding, we can periodically transmit intraframe-encoded frames that provide regeneration points. The refreshment rate could be adapted according to general network conditions as perceived by the sender, with the exact rate striking the balance between compression and error tolerance. Alternatively, we can prioritize the components of a, possibly hierarchically encoded, media stream at the sender in advance and then transmit components at redundancy levels reflecting their priorities ([3]). By interleaving all components in each packet, the result of the variable priority and redundancy levels is that more important components will be recoverable using a smaller fraction of the total number of packets sent, compared to the corresponding fraction for less important ones. The net effect is added reliability for important media components with only moderate increases in total transmission bandwidth.

For flow and congestion control, we cannot simply increase redundancy to avoid reliance on feedback, since this will make problems worse. However, we can still use preventive open-loop methods based on resource reservations so that streams will be guaranteed to meet their deadlines. The control overhead from setting up resource reservations will be incurred only infrequently rather than continuously, as in feedback-based methods. It is important to note that resource reservations must be made on a per-stream basis, according to the characteristics of each medium and to the importance placed on it by the application and the receivers themselves. A communication abstraction promoting open-loop control and providing a service analogous to that of a television broadcast channel is the *Multimedia Multicast Channel* ([78]). A source transmits continuous-media streams onto the channel and receivers "tune in" to the channel to receive a selected subset of the streams. To support heterogeneity, each receiver may tailor the selected streams to meet individual needs through the use of filters, which, if compatible, can propagate upstream and combine in order to economize on resources ([77]).

Use of hierarchical coding can further adapt the streams to application priorities and network and receiver capabilities. For example, effective open-loop congestion control can be exercised in high-speed networks by selectively dropping high-resolution signal contents without source or receiver involvement and with limited signal quality degradation, which is localized to the subtree downstream from the congestion path. Alternatively, when prioritized and encoded streams are interleaved in each packet at variable redundancy levels as above, *any* packets can be dropped with smaller impact on high priority streams than on low priority ones ([3]). With this scheme the network does not have to be modified to recognize priorities, but redundancy must be added to high priority streams.

It is not yet clear whether effective guarantees can be provided in the face of dynamic routing schemes which will force abandoning reservations in paths that have been physically disconnected due to link or switch failures. In addition, the guarantees cannot be absolutely strict for all streams if statistical multiplexing is to provide any economies at all. Therefore, an application should expect at least minor and probably major service degradation during temporary congestion periods or route

changes. FEC can smoothen out the minor anomalies, but to be both effective and economical, the redundancy rate will have to adapt to current network conditions rather than being adequate for the worst case. A scalable feedback mechanism that can be used to estimate network conditions without creating implosion problems has been proposed ([12]): it first estimates the number of receivers in a group and then what the average quality of reception is (the averaging technique depends on the application), using probabilistic techniques. This method has been used in applications ([94]) for senders to detect congestion problems and adapt their output rates (to relieve congestion) and error redundancy factors (to increase the chances of error recovery). A further enhancement to scalable feedback control can be obtained by splitting the receivers in groups according to their reception status (congestion control) and capabilities (flow control), and only send them the data that each group can handle. This avoids problems created by very slow or very fast machines dragging the whole group towards one extreme ([18]).

## V. Experience with multicasting on the Internet

The Internet, although lacking support for many of the features discussed in the previous sections, has been extensively used as a testbed for algorithms and protocols supporting multimedia multicasting. The network layer service uses IP (the Internet Protocol) which provides connectionless, datagram-based, best-effort delivery. Only unicast routing is provided using various algorithms, and no quality-of-service or resource reservation provisions are made. Reliability is provided if required at the transport layer using TCP (the Transmission Control Protocol), or it may be bypassed using UDP (the User Datagram Protocol). TCP provides error-free byte streams and uses a windowing scheme for flow and congestion control, with the window size adapting to the loss rate encountered, which is taken as an indication of congestion status ([52]). By themselves, these characteristics are not sufficient to provide robust support for either real-time continuous media or multicast, and some features, such as the congestion control scheme, may even get in the way. However, experimentation with multicast protocols and multimedia applications over the Internet has been extensive since the first audiocast of an Internet Engineering Task Force (IETF) meeting ([13]).

Regarding IP support for multicasting, early work on routing algorithms ([22]) and the host-group model ([17]) initiated the current flurry of activity on the Internet, although some work has also been influenced by earlier research in broadcasting ([97]). The extensions of the IP model to support multicasting are the provision of special (class D) multicast addresses and IGMP (the Internet Group Management Protocol). IGMP supports the host-group model, with receivers explicitly joining the groups denoted by multicast addresses ([21]). Multicast-aware routers periodically multicast, on a well-known address, membership queries on their LANs and gather replies from interested hosts in order to discover which groups have members present in their area.

Routing is performed by routers that learn the shortest paths to each group member using DVMRP (Distance Vector Multicast Routing Protocol) and simply aggregate these shortest paths into a tree. DVMRP is based on the Bellman-Ford algorithm, using the underlying unicast routing tables to find the shortest routes. Initially, it only performed truncated broadcast, i.e. it forwarded all packets everywhere, only dropping them if no group members were located in the lowest level leaf subnetworks ([24]). Extended versions can prune unused parts of this tree, thus conserving resources. Since only some machines have been extended to support multicast, the multicast-aware routers that serve multicast-aware sub-networks directly exchange *encapsulated* packets among them. These are regular packets embedded in another packet that has the two multicast routers as source and destination. Thus, the encapsulated packets travel over virtual links, called *tunnels*, which are simple unicast routes that pass through non-multicast-capable routers without disturbing them. These multicast routers and the virtual links connecting them form the MBone ([31]). DVMRP and the tunneling scheme suffer from inefficiencies and inflexibility, so an alternative routing protocol has been proposed: MOSPF (Multicast Open Shortest Path First). MOSPF uses Dijkstra's algorithm to create shortest path trees which do not require pruning ([69], [70], [71]).

Since both DVMRP and MOSPF suffer from scalability problems, two new routing protocols have been proposed: CBT (Core Based Trees) builds a single tree for each group trying to conserve resources ([6]), based on predetermined fixed points. PIM (Protocol Independent Multicast) also supports single shared trees, but it allows shortest-delay paths when receivers request them, and even traditional per-source trees ([23]). Both protocols were designed with an eye on scalability, interoperability with various underlying routing schemes and possible resource reservation protocols. Technically they are advances over DVMRP and MOSPF, especially PIM which tries to combine simple management with short delays, but they are more complex than their less-flexible predecessors and thus more difficult to deploy on a large scale. One way to achieve both scalability and interoperability among different routing protocols is to use hierarchical routing, where a second-level protocol routes data among disjoint network areas ([90]). Each such area chooses its own first-level protocol, which routes data based only on local topological information.

Two proposals for Internet resource reservations exist. ST-II (Stream Protocol II) is a complete network-layer protocol that supports both multicasting and resource-oriented negotiations based on a sender-initiated approach ([91]). Actually, it is more of a framework, with specific mechanisms expected to be provided externally. Implementations have been tested ([76]) and new versions designed ([25]). A proposal closer to the ones presented here is RSVP (Resource ReSerVation Protocol) which acts as an overlay on routing protocols, supporting receiver-initiated resource reservations over any available multicast routing scheme ([103]), essentially reversing the ST-II mechanism ([67]). In addition, RSVP supports dynamic reservation modifications and network reconfigurations. Experimental implementations interoperating with other IP-multicast extensions are currently being tested.

Furthermore, a new transport protocol supporting continuous media has been developed: RTP (Real Time Protocol) ([83]) provides support for timing information, packet sequence-

numbers and option specification, without imposing any additional error control or sequencing mechanisms. An application can use this basic framework adapted to its requirements to add whatever mechanisms seem appropriate, such as error control based on loss detection using sequence numbers, or intra-media and inter-media synchronization based on timing information. A companion control protocol, RTCP (Real Time Control Protocol), can be used for gathering feedback from the receivers, again according to the application's needs. For example, an application can use RTP for transport and RTCP adapted for scalable feedback control, along with appropriate FEC and adaptation mechanisms ([94]).

Another relevant protocol is SDP (Session Description Protocol) ([47]), which provides a mechanism for applications to learn what streams are carried in the network, describing them in adequate detail so that anyone interested can launch the appropriate receiver applications. When multiple hosts and applications are communicating, as in a multi-party conference, there is a need to mediate transmission and reception of data among participants. As the specific needs of each application and conference setting may vary, one way to support multiple control policies is to use a logical *conference control channel* as a shared mechanism through which control messages are exchanged ([46]). Floor control and session management applications can then employ this channel for their needs.

## VI. Conclusion

In this paper we have presented the issues arising from real-time continuous-media support and multicasting in packet networks, emphasizing issues of interaction between the two. The heavy demands of continuous media on any network have motivated research in many areas, with multicasting being one of the most active, due to its potential for large economies in resource usage. Apart from the problems that are specific to either multimedia or multicasting, there are special issues that become more important when the two are combined. Fortunately, the nature of continuous media gives rise to some special-case solutions that can be exploited to our advantage.

Issues such as bandwidth and delay requirements, error tolerance, statistical guarantees and heterogeneity, create a complex environment that must be supported in an efficient and flexible way. Probably, the most important research issues revolve around the core theme of open-loop, preventive or anticipatory, mechanisms versus the traditional closed-loop corrective ones. The exact mix of these two modes of operation achieving the desired quality-of-service and flexibility with minimum cost is one of the most interesting topics for further research.

## References


[1] Aguilar et al. 1986 Aguilar L, Garcia-Luna-Aceves JJ, Moran D, Graighill EJ, Brungardt R (1986) Architecture for a multimedia teleconferencing system. In: Proceedings of the ACM SIGCOMM '86, pp 126-36, Association for Computing Machinery, Baltimore, Maryland

[2] Ahamad 1990 Ahamad M (ed) (1990) Protocols for high-speed networks. Technology Series, IEEE Comput Soc Press, Los Alamitos, Calif, USA

[3] Albanese et al. 1994 Albanese A, Blomer J, Edmonds J, Luby M, Sudan M (1994) Priority encoding transmission. In: Goldwasser S (ed), Proceedings of the 35th annual symposium on foundations of computer science, pp 604-12, IEEE Comput Soc Press, Los Alamitos, Calif, USA

[4] Armstrong et al. 1992 Armstrong S, Freier A, Marzullo K (1992) Multicast transport protocol. Internet Request For Comments, RFC 1301

[5] AT&T 1984 AT&T (1984) Engineering and operations in the Bell system. AT&T Bell Laboratories

[6] Ballardie et al. 1993 Ballardie A, Crowcroft J, Francis P (1993) Core-based trees (CBT) - An architecture for scalable inter-domain multicast routing. Comput Commun Rev 23(4):85-95 (Proceedings of the ACM SIGCOMM '93)

[7] Bellman 1957 Bellman RE (1957) Dynamic programming. Princeton University Press, Princeton, New Jersey, USA

[8] Bettati et al. 1995 Bettati R, Ferrari D, Gupta A, Heffner W, Howe W, Moran M, Nguyen Q, Yavatkar R (1995) Connection establishment for multi-party real-time communication. In: Proceedings of the 5th international workshop on network and operating system support for digital audio and video, pp 255-65, Springer-Verlag, Berlin, Germany

[9] Biersack 1992 Biersack EW (1992) Performance evaluation of forward error correction in ATM networks. Comput Commun Rev 22(4):248-57 (Proceedings of the ACM SIGCOMM '92)

[10] Biersack and Nonnenmacher 1995 Biersack EW, Nonnenmacher J (1995) WAVE: A new multicast routing algorithm for static and dynamic multicast groups. In: Proceedings of the 5th international workshop on network and operating system support for digital audio and video, pp 243-54, Springer-Verlag, Berlin, Germany

[11] Blakowski and Steinmetz 1996 Blakowski G, Steinmetz R (1996) A media synchronization survey: Reference model, specification, and case studies. IEEE J Select Areas Commun 14(1):5-35

[12] Bolot et al. 1994 Bolot JC, Turletti T, Wakeman I (1994) Scalable feedback control for multicast video distribution in the Internet. Comput Commun Rev 24(4):58-67 (Proceedings of the ACM SIGCOMM '94)

[13] Casner 1992 Casner S (1992) First IETF Internet audiocast. Comput Commun Rev 22(3):92-7

[14] Casner et al. 1990 Casner S, Seo K, Edmond W, Topolcic C (1990) N-way conferencing with packet video. Tech Rep ISI/RS-90-252. Information Sciences Institute, Marina Del Rey, Calif, USA (Proceedings of the 3rd international workshop on packet video)

[15] Chandran and Lin 1992 Chandran SR, Lin S (1992) Selective-repeat-ARQ schemes for broadcast links. IEEE Trans Commun 40(1):12-9

[16] Chanson et al. 1989 Chanson ST, Neufeld GW, Liang L (1989) A bibliography on multicast and group communications. Oper Syst Rev 23(4):20-5

[17] Cheriton and Deering 1985 Cheriton DR, Deering S (1985) Host groups: A multicast extension for datagram internetworks. In: Proceedings of the 9th data communications symposium, pp 172-9, IEEE Comput Soc Press, Baltimore, Maryland, USA

[18] Cheung et al. 1995 Cheung SY, Ammar MH, Li X (1995) On the use of destination set grouping to improve fairness in multicast video distribution. In: Proceedings of the IEEE INFOCOM '96, pp 553-60, IEEE Comput Soc Press, Los Alamitos, Calif, USA (Tech Rep GIT-CC-95-25. Georgia Institute of Technology, Atlanta, GA)

[19] Clark et al. 1992 Clark DD, Shenker S, Zhang L (1992) Supporting real-time applications in an integrated services packet network: Architecture and mechanism. Comput Commun Rev 22(4):14-26 (Proceedings of the ACM SIGCOMM '92)

[20] Crowcroft and Paliwoda 1988 Crowcroft J, Paliwoda K (1988) A multicast transport protocol. Comput Commun Rev 18(4):247-56 (Proceedings of the ACM SIGCOMM '88)

[21] Deering 1989 Deering S (1989) Host extensions for IP multicasting. Internet Request For Comments, RFC 1112

[22] Deering and Cheriton 1990 Deering S, Cheriton DR (1990) Multicast routing in internetworks and extended LANs. ACM Trans Comput Syst 8(2):85-110

[23] Deering et al. 1994 Deering S, Estrin D, Farinacci D, Jacobson V, Liu C, Wei L (1994) An architecture for wide-area multicast routing. Comput Commun Rev 24(4):126-35 (Proceedings of the ACM SIGCOMM '94)

[24] Deering et al. 1988 Deering S, Partridge C, Waitzman D (1988). Distance vector multicast routing protocol. Internet Request For Comments, RFC 1075

[25] Delgrossi and Berger 1995 Delgrossi L, Berger L (1995) Internet stream protocol Version 2 (ST2) protocol specification - Version ST2+. Internet Request For Comments, RFC 1819

[26] Dijkstra 1959 Dijkstra EW (1959) A note on two problems in connection with graphs. Numer Math 1:269-71

[27] Doar and Leslie 1993 Doar M, Leslie I (1993) How bad is naive multicast routing? In: Proceedings of the IEEE INFOCOM '93, pp 82-9, IEEE Comput Soc Press, Los Alamitos, Calif, USA

[28] Eckberg 1992 Eckberg AE (1992) B-ISDN/ATM traffic and congestion control. IEEE Network 6(5):28-37

[29] Ellis et al. 1991 Ellis CA, Gibbs SJ, Rein GL (1991) Groupware: Some issues and experiences. Commun ACM 34(1):39-58

[30] Eng et al. 1988 Eng KY, Hluchyj MG, Yeh YS (1988) Multicast and broadcast services in a knockout packet switch. In: Proceedings of the IEEE INFOCOM '88, pp 29-34, IEEE Comput Soc Press, Los Angeles, Calif, USA

[31] Eriksson 1994 Eriksson H (1994) MBONE: The multicast backbone. Commun ACM 37(8):54-60

[32] Ferrari 1990 Ferrari D (1990) Client requirements for real-time communication services. IEEE Commun Mag 28(11):65-72

[33] Ferrari 1992 Ferrari D (1992) Delay jitter control scheme for packet-switching internetworks. Comput Commun 15(6):367-73

[34] Ferrari and Verma 1990 Ferrari D, Verma DC (1990) A scheme for real-time channel establishment in wide-area networks. IEEE J Select Areas Commun 8(3):368-79

[35] Ferrari et al. 1997 Ferrari D, Gupta A, Ventre G (1997) Distributed advance reservation of real-time connections. Multimedia Systems 5(3):187-98 (Proceedings of the 5th international workshop on network and operating system support for digital audio and video)

[36] Floyd et al. 1995 Floyd S, Jacobson V, McCanne S, Liu CG, Zhang L (1995) A reliable multicast framework for light-weight sessions and application level framing. Comput Commun Rev 25(4):342-56 (Proceedings of the ACM SIGCOMM '95)

[37] Ford and Fulkerson 1962 Ford LR, Fulkerson DR (1962) Flows in networks. Princeton University Press, Princeton, New Jersey, USA

[38] Frank et al. 1985 Frank AJ, Wittie LD, Bernstein AJ (1985) Multicast communication on network computers. IEEE Software 2(3):49-61

[39] Le Gall 1991 Le Gall D (1991) MPEG: A video compression standard for multimedia applications. Commun ACM 34(4):46-58

[40] Garey et al. 1978 Garey MR, Graham RL, Johnson DS (1978) The complexity of computing Steiner minimal trees. SIAM J Appl Math 34:477-95

[41] Ghanbari 1989 Ghanbari M (1989) Two-layer coding of video signals for VBR networks. IEEE J Select Areas Commun 7(5):771-81

[42] Grief 1992 Grief I (ed) (1992) Computer supported co-operative work. Morgan Kaufmann, San Mateo, Calif, USA

[43] Guojun 1993 Guojun L (1993) Advances in digital image compression techniques. Comput Commun 16(4):202-14

[44] Gupta et al. 1995 Gupta A, Howe W, Moran M, Nguyen Q (1995) Resource sharing for multi-party real-time communication. In: Proceedings of the IEEE INFOCOM '95, pp 1230-7, IEEE Comput Soc Press, Los Alamitos, Calif, USA

[45] Hakimi 1971 Hakimi SL (1971) Steiner's problem in graphs and its implications. Networks 1:113-33

[46] Handley et al. 1995 Handley M, Wakeman I, Crowcroft J (1995) The Conference Control Channel Protocol (CCCP): A scalable base for building conference control applications. Comput Commun Rev 25(4):275-87 (Proceedings of the ACM SIGCOMM '95)

[47] Hardly and Jacobson 1995 Hardly M, Jacobson V (1995) SDP: Session description protocol. Internet Draft, Multipart Multicast Session Control Working Group

[48] Hehman et al. 1990 Hehman DB, Salmony MG, Stuttgen HJ (1990) Transport services for multimedia applications on broadband networks. Comput Commun 13(4):197-203

[49] Herzog et al. 1995 Herzog S, Shenker S, Estrin D (1995) Sharing the "cost" of multicast trees: an axiomatic analysis. Comput Commun Rev 25(4):315-27 (Proceedings of the ACM SIGCOMM '95)

[50] Huang and Knauer 1984 Huang A, Knauer S (1984) Starlite: a wideband digital switch. In: Proceedings of the IEEE GLOBECOM '84, pp 121-5, Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, USA

[51] Hughes 1989 Hughes L (1989) Survey of multicast address handling techniques for Ethernet communication controllers. Microprocessors Microsystems 13(9):563-8

[52] Jacobson 1988 Jacobson V (1988) Congestion avoidance and control. Comput Commun Rev 18(4):314-29 (Proceedings of the ACM SIGCOMM '88)

[53] Jiang 1992 Jiang X (1992) Routing broadband multicast streams. Comput Commun 15(1):45-51

[54] Jurgen 1992 Jurgen RK (ed) (1992) An abundance of video formats. IEEE Spectrum, March, 26-8

[55] Kabada and Jaffe 1983 Kabada BK, Jaffe JM (1983) Routing to multiple destinations in computer networks. IEEE Trans Commun 31(3):343-51

[56] Karlsson and Vetterli 1989 Karlsson G, Vetterli M (1989) Packet video and its integration into the network architecture. IEEE J Select Areas Commun 7(5):739-51

[57] Kompella et al. 1993 Kompella VP, Pasquale JC, Polyzos GC (1993) Multicast routing for multimedia communication. IEEE/ACM Trans Networking 1(3):286-92

[58] Kompella et al. 1996 Kompella VP, Pasquale JC, Polyzos GC (1996) Optimal multicast routing with quality of service constraints. J Sys Network Manag 4(2):107-31

[59] Kou et al. 1981 Kou L, Markowsky G, Berman L (1981) A fast algorithm for Steiner trees. Acta Informatica 15:141-5

[60] Kurose 1993 Kurose J (1993) Open issues and challenges in providing QoS guarantees in high-speed networks. Comput Commun Rev 23(1):6-15

[61] Lee 1988 Lee TT (1988) Nonblocking copy networks for multicast packet switching. IEEE J Select Areas Commun 6(9):1455-67

[62] Leiner 1988 Leiner B (ed) (1988) Critical issues in high-bandwidth networking. Internet Request For Comments, RFC 1077

[63] Leung et al. 1990 Leung WH, Baumgartner TJ, Hwang YH, Morgan MJ, Tu SC (1990) A software architecture for workstations supporting multimedia conferencing in packet switching networks. IEEE J Select Areas Commun 8(3):380-90

[64] Liou 1991 Liou M (1991) Overview of the $p\times64$ Kbit/s video coding standard. Commun ACM 34(4):59-63

[65] Lippman 1991 Lippman A (1991) Feature sets for interactive images. Commun ACM 34(4):92-102

[66] Mase et al. 1983 Mase K, Takenada T, Yamamoto H, Shinihara M (1983) Go-back-n ARQ schemes for point-to-multipoint satellite communications. IEEE Trans Commun 31(4):583-9

[67] Mitzel et al. 1994 Mitzel DJ, Estrin D, Shenker S, Zhang L (1994) An architectural comparison of ST-II and RSVP. In: Proceedings of the IEEE INFOCOM '94, pp 716-25, IEEE Comput Soc Press, Los Alamitos, Calif, USA

[68] Mockapetris 1983 Mockapetris PV (1983) Analysis of reliable multicast algorithms for local networks. In: Proceedings of the 8th data communications symposium, pp 150-7, IEEE Comput Soc Press, Baltimore, Maryland, USA

[69] Moy 1994a Moy J (1994a) MOSPF: Analysis and experience. Internet Request For Comments, RFC 1585

[70] Moy 1994b Moy J (1994b) Multicast extensions to OSPF. Internet Request For Comments, RFC 1584

[71] Moy 1994c Moy J (1994c) Multicast routing extensions for OSPF. Commun ACM 37(8):61-6

[72] Nomura et al. 1989 Nomura M, Fujii T, Ohta N (1989) Basic characteristics of variable rate video coding in ATM environment. IEEE J Select Areas Commun 7(5):752-60

[73] Pancha and El Zarki 1994 Pancha P, El Zarki M (1994) MPEG coding for variable bit rate video transmission. IEEE Commun Mag 32(5):54-66

[74] Partridge 1990 Partridge C (ed) (1990) Workshop report: Internet research steering group workshop on very-high-speed networks. Internet Request For Comments, RFC 1152

[75] Partridge 1992 Partridge C (1992) A proposed flow specification. Internet Request For Comments, RFC 1363

[76] Partridge and Pink 1992 Partridge C, Pink S (1992) An implementation of the revised internet stream protocol (ST-2). Internetworking: Res Exp 3(1):27-54

[77] Pasquale et al. 1994 Pasquale JC, Polyzos GC, Anderson EW, Kompella VP (1994) Filter propagation in dissemination trees:

Trading off bandwidth and processing in continuous media networks. In: D. Sheperd, G. Blair, G. Coulson, N. Davies and others (eds), Proceedings of the 4th international workshop on network and operating system support for digital audio and video, pp 259-68, Springer-Verlag, Berlin, Germany

[78] Pasquale et al. 1994 Pasquale JC, Polyzos GC, Anderson EW, Kompella VP (1994) The multimedia multicast channel. Internetworking: Res Exp 5(4):151-62

[79] Pingali et al. 1994 Pingali S, Towsley D, Kurose JF (1994) A comparison of sender-initiated and receiver-initiated reliable multicast protocols. Performance Evaluation Rev 22(1):221-30 (Proceedings of the ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems)

[80] Rayward-Smith 1983 Rayward-Smith VJ (1983) The computation of nearly minimal Steiner trees in graphs. Int J Math Educ Sci Tech 14(1):15-23

[81] Sabri and Prasada 1985 Sabri S, Prasada B (1985) Video conferencing systems. Proc IEEE 73(4):671-88

[82] Santoso and Fdida 1993 Santoso H, Fdida S (1993) Transport layer multicast: An enhancement for XTP bucket error control. IFIP Trans C (communication systems) C-14:333-49

[83] Schulzrinne et al. 1996 Schulzrinne H, Casner S, Frederick R, Jacobson V (1996) RTP: A transport protocol for real-time applications. Internet Request For Comments, RFC 1889

[84] Shacham 1992 Shacham N (1992) Multipoint communication by hierarchically encoded data. In: Proceedings of the IEEE INFOCOM '92, pp 2107-14, IEEE Comput Soc Press, Los Alamitos, Calif, USA

[85] Shacham and McKenney 1990 Shacham N, McKenney P (1990) Packet recovery in high-speed networks using coding and buffer management. In: Proceedings of the IEEE INFOCOM '90, pp 124-31, IEEE Comput Soc Press, Los Alamitos, Calif, USA

[86] Shenker and Breslau 1995 Shenker S, Breslau L (1995) Two issues in reservation establishment. Comput Commun Rev 25(4):14-26 (Proceedings of the ACM SIGCOMM '95)

[87] Sincoskie 1990 Sincoskie WD (1990) Video on demand: is it feasible? In: Proceedings of the IEEE GLOBECOM '90, pp 201-5, Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, USA

[88] Steinmetz 1996 Steinmetz R (1996) Human perception of jitter and media synchronization. IEEE J Selec Areas Commun 14(1):61-72

[89] Sutherland and Litteral 1992 Sutherland J, Litteral L (1992) Residential video services. IEEE Commun Mag 30(7):36-41

[90] Thyagarajan and Deering 1995 Thyagarajan A, Deering S (1995) Hierarchical distance-vector multicast routing for the MBone. Comput Commun Rev 25(4):60-6 (Proceedings of the ACM SIGCOMM '95)

[91] Topolcic 1990 Topolcic C (1990) Experimental Internet stream protocol, Version 2 (ST-II). Internet Request For Comments, RFC 1190

[92] Trajkovic and Golestani 1992 Trajkovic L, Golestani SJ (1992) Congestion control for multimedia services. IEEE Network 6(5):20-6

[93] Tudor 1995 Tudor PN (1995) MPEG-2 video compression. Electr Commun Eng J 7(6):257-64

[94] Turletti 1994 Turletti T (1994) The INRIA videoconferencing system (IVS). ConneXions 8(10):20-4

[95] Turner 1988 Turner JS (1988) Design of a broadcast packet switching network. IEEE Trans Commun 36(6):734-43

[96] Turner 1992 Turner JS (1992) Managing bandwidth in ATM networks with bursty traffic. IEEE Network 6(5):50-8

[97] Wall 1982 Wall DW (1982) Mechanisms for broadcast and selective broadcast. Ph.D. thesis, Computer Science Department, Stanford University, Calif, USA

[98] Wallace 1991 Wallace GK (1991) The JPEG still picture compression standard. Commun ACM 34(4):30-44

[99] Waxman 1988 Waxman BM (1988) Routing of multipoint connections. IEEE J Selec Areas Commun 6(9):1617-22

[100] Wei and Estrin 1994 Wei L, Estrin D (1994) The trade-offs of multicast trees and algorithms. Tech Rep USC-CS-93-560. University of Southern California, Los Angeles, Calif, USA (Proceedings of the 3rd International Conference on Computer Communications and Networks)

[101] Wei and Estrin 1995 Wei L, Estrin D (1995) Multicast routing in dense and sparse modes: Simulation study of tradeoffs and dynamics. In: K. Makki, N. Pissinou (eds), Proceedings of the 4th International Conference on Computer Communications and Networks, pp 150-7, IEEE Comput Soc Press, Los Alamitos, Calif, USA (Tech Rep USC-CS-95-613. University of Southern California, Los Angeles, Calif, USA)

[102] Zhang 1995 Zhang H (1995) Service disciplines for guaranteed performance service in packet switching networks. Proc IEEE 83(10):1347-96

[103] Zhang et al. 1993 Zhang L, Deering S, Estrin D, Shenker S, Zappala D (1993) RSVP: A new resource reservation protocol. IEEE Network 7(5):8-18

[104] De Zong et al. 1993 De Zong W, Onozato Y, Kaniyil J (1993) A copy network with shared buffers for large-scale multicast ATM switching. IEEE/ACM Trans Networking 1(2):157-65