Filter Propagation in Dissemination Trees: Trading Off Bandwidth and Processing in Continuous Media Networks

Joseph C. Pasquale, George C. Polyzos
Eric W. Anderson, Vachaspathi P. Kompella

Computer Systems Laboratory
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114

{pasquale, polyzos, ewa, kompella}@cs.ucsd.edu

Abstract

We describe the concept of the *relocatable continuous media filter*. The novelty of these filters is how they can propagate over a dissemination tree in a network. We describe the filter propagation protocol to achieve this. Execution of filters inside a network allows the network to be viewed in a novel way, as a "processor" with its "instruction set" being the various types of available filters. Since filters generally modify the data rate of the continuous media stream, usually (but not necessarily) reducing it, filters allow the trading off of bandwidth and processing in a network.

1. Motivation

High performance workstations and high capacity networks have become enabling technologies for distributed continuous media (CM) applications. For example, videoconferencing and remote video services will be popular for business and home users, while visualization of animations of remotely-located scientific data sets will be popular for scientific users.

While the networks of the near future will offer gigabit per second and higher bandwidths, we believe there will always be a need for more bandwidth given the current trends toward videoconferencing, hundreds of channels offered by cable television providers, as well as the desire for user-selectable on-demand "pay-per-view" home movies. These video applications are highly I/O-intensive; for example, the data rate for a single

MPEG-compressed high definition television (HDTV) video-audio stream is 20-40 Mb/s [1]. The problem is certain to be magnified once interactive virtual reality applications gain widespread popularity.

In fact, network bandwidth will frequently be a scarce resource because, ultimately, it is a shared resource whose consumption depends on all users. Workstations, on the other hand, are private resources, whose load depends only on its own user. Workstations are easily upgraded to ones that are faster, in turn making greater I/O-intensive distributed applications possible and increasing the network load correspondingly. Networks, on the other hand, are not easily upgraded.

Multicasting and filtering are tools for economizing bandwidth. There are various arguments that can be made in favor of bandwidth conservation. First of all, users will have to (eventually, if not now) pay for bandwidth usage. With ATM networks built and run by the bandwidth provider, this is certainly going to be the case. This can actually happen earlier than expected; even the Internet is preparing for usage-based charges. Therefore, it makes sense to conserve bandwidth even when it is possible to accommodate the information flows at the full bandwidth to all destinations.

Furthermore, even though the goal of the networking community is to achieve ubiquitous connectivity at "full" bandwidth, it is not clear if and how soon this goal will be achieved. The difficulties in achieving "the last mile" for fiber to the home is a good example of problems in this area. Therefore, it is reasonable to assume that for the foreseeable future, connectivity between some endpoints will be provided through multiple networks interconnected through gateways, i.e. internetworking will still be important in the future. In many such situations, bandwidth and other resources might be relatively scarce in some of the networks or the interconnection points.

Finally, even if it is technically feasible to distribute the full signal to all participants, economics might introduce a significant difference among destinations. Consider for example a mostly local videoconference with a few remote participants. Having the capability of incorporating the remote participants at reasonable cost without changing the quality or character of the local part of the conference is a definite advantage. There could be natural points where it might be helpful to throttle the bandwidth, e.g. at gateways between different networks.

Fortunately, there exist opportunities for more efficiently managing network bandwidth given distributed CM applications:

- **Dissemination Trees**: Many of these distributed CM applications have dissemination-based communication models [2], where there is some source generating data for a large number of receivers, e.g. cable television, home movies, etc. Multicast routing based on source-rooted *dissemination trees* limits bandwidth consumption by avoiding whenever possible the duplication of data streams.

- **Variable Resolution**: CM such as audio or video has the characteristic that its resolution can be varied to produce different levels of quality, many of which may be acceptable to the application. Lowering resolution will lower the data rate of the continuous media stream; therefore, given a mechanism which can dynamically control resolution, the data rate can be adjusted to meet special needs of receivers or to meet network constraints, or even in response to special conditions of the network.

To elaborate on these points, we consider some examples. Usually, receivers will be

heterogeneous in their ability to consume data based on their local equipment. A CM stream disseminated from a single point may be generated as a high-quality data stream, particularly if the source is a commercial provider. HDTV with CD-quality sound might be appropriate for a feature presentation. But some (possibly many) of the receivers will lack the hardware required to take advantage of such rich data. Some receivers will only be able to display black-and-white, or gray-scale, or 8-bit color, rather than 24-bit "true color." Not only may these receivers be unable to present the CM in its full detail, but they may have difficulty merely receiving the full data stream and selecting an appropriate portion for presentation. In such cases, there is no need to send full resolution video to these receivers, and therefore, there is an opportunity to lower the required data rate.

On the other hand, there could be situations where it would be beneficial to have bandwidth traded off for processing. For example, consider a set of similar workstations lacking decompression hardware attached to a broadcast local area network (LAN) being part of a dissemination tree. It is possible then to have the router attached to the LAN and, being upstream in the dissemination tree, to use a filter to implement the decompression function and thus provide a single common final version of the channel flow, ready to be presented by the workstations.

Finally, consider that sending CM at a high resolution sometimes yields only marginal gain. For an audio stream of a single person speaking, the difference between so-called "voice quality" audio, at 8 KBps, and "CD quality" audio, in stereo, at just over 160 KBps, is largely aesthetic. These two data rates, however, differ by a factor of 20. Users who are faced with paying for their network bandwidth may be highly inclined to trade a slight service reduction for a factor of 20 cost savings. Similar flexibility is possible with video, where the frame size, frame rate, and color depth can all be varied, each by over an order of magnitude. It is highly desirable if this can be done dynamically, at the discretion of the receiver and without disturbing the source.

In this paper, we describe a concept called the *relocatable continuous media filter* ("filter" for short), used in networks to take advantage of these opportunities for manipulating the required bandwidths of distributed CM applications. These filters can relocate themselves in a dissemination tree by propagating from a receiver upstream towards the source such that system resources are most efficiently and effectively utilized. While the idea of filtering is not new, the relocation of filters is novel in that filters may now be viewed as an abstraction for network processing power. Filters provide the ability to achieve various trade-offs between bandwidth, the traditional resource offered by the network, and processing, a non-traditional network resource.

2. Communication Model

We first describe a model for dissemination-based communication based on that described in [3]. The basic objects in this model are segments, streams and multi-streams, channels and dissemination trees, sources and receivers, and filters. A *segment* is an application-level unit of data comprised of a time-stamped sample or set of samples of continuous media, such as a block of audio samples or a video frame. A *stream* is a typed sequence of segments. A *multi-stream* is a set of related streams; typically, the relationship is based on time-correlation, where segments of different streams are synchronized at playback time.

A *channel* is the object which distributes a multi-stream from a source to multiple receivers: a source transmits a multi-stream onto a channel; a receiver receives a multi-stream by "tuning in" to a channel. Generally, many receivers can tune-in to a channel;

however, only one source can transmit onto a channel. A *dissemination tree* is a directed source-rooted tree which is a sub-graph of the network connecting a single source and multiple receivers; this tree defines the paths over which a channel delivers a multi-stream. To construct such a tree for continuous media applications, both bandwidth and delay must be taken into account (see Figure 1). A routing algorithm which effectively minimizes bandwidth while satisfying source-destination contraints (e.g. bounding delay) is described in [4]. A channel's dissemination tree can change over time (however, we ignore this complication in this paper).

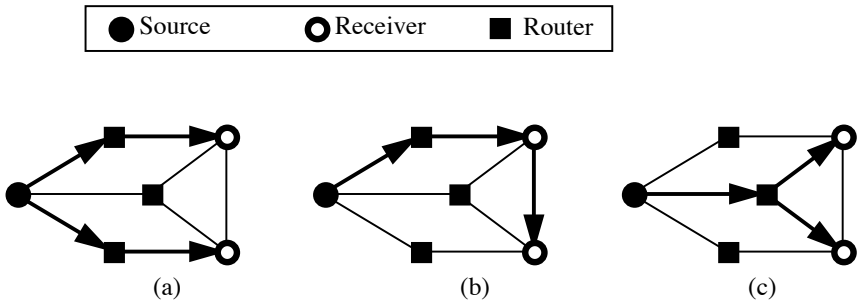

Figure 1: Three dissemination trees. In (a), delay from source to either receiver is 2 hops, but total bandwidth usage is 4 links. In (b), total bandwidth usage is 3 links, but maximum delay is 3 hops. In (c), both bandwidth (3 links) and delay (2 hops) are minimized.

3. Filters

A *filter* is a transformer of one or more input streams of a multi-stream into an output stream; the output stream replaces the input streams in the multi-stream. A receiver which has tuned in to a channel may place a filter on the channel,[1] identifying which input stream is to be filtered (see Figure 2).

There are a variety of functions which filters may implement. The simplest is a *selective* filter, which can select which segments get forwarded and which do not (i.e. which get dropped). For example, a selective filter may drop all segments on a stream. A receiver which could only display lower frequency components of a hierarchically coded video multi-stream would filter out the higher component streams. A more sophisticated selective filter would select 1 out of n segments, as would be used by a receiver which could not display video at the full frame rate offered by the source.

A yet more powerful filter is a *transforming* filter, which would actually carry out computations on a segment to produce a new segment. For example, many personal computers do not have the capability of displaying 24-bit color, but are able to display 8-bit color. A transforming filter may be applied to a video stream to reduce its resolution from 24-bit color pixels to 8-bit color using some form of dithering. Compression is a good

1. Actually, a filter is placed on a *port*, which is the receiver's access point to a channel [3].

example of a function carried out by a transforming filter (see Figure 2).

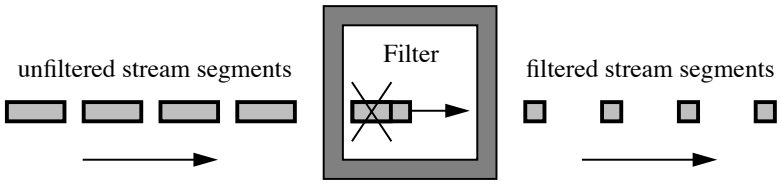

Figure 2: Example of a filter which reduces resolution of a continuous media stream by dropping information, resulting in a stream requiring a lower data rate.

Finally, filters need not be limited to operation on a single input stream. A *mixing* filter can accept multiple input streams, carry out some computation on them, and produce a new stream. For example, a multi-stream may include separate audio streams for stereo reproduction; a mixing filter can combine these audio streams to produce monaural audio as would be appropriate if the receiver had only a single loudspeaker.

4. Filter Propagation

When a receiver places a filter on a channel, where should it execute? An obvious location would be at the network's edge nearest the receiver. In this case, the channel must provide enough bandwidth to transport the input stream, even though the output stream may require (perhaps significantly) less bandwidth. Note that the closer the filter's location is to the source, the less total bandwidth is required (see Figure 3).[2]

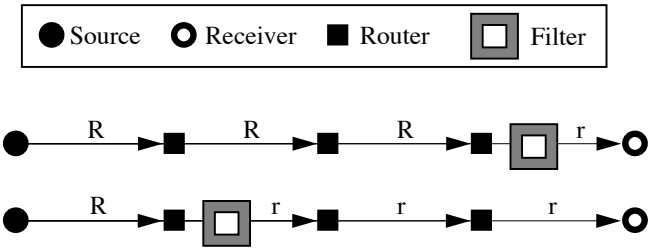

Figure 3: Let R and r be data rates, with R > r. The total utilized bandwidth of the links in (a) is greater than that of (b), as a result of propagating a data rate-reducing filter toward the source.

To take advantage of this simple observation, filters are propagated upstream towards the source in the dissemination tree, and stop whenever they reach a node with more than one output link. At this node, when a segment arrives, a copy is filtered and then transmitted over the link over which the filter propagated, while unfiltered copies of the segment are forwarded on the other outgoing links. In this manner, bandwidth consumption is minimized.

---

2. This assumes that the filter is bandwidth-reducing, otherwise, it may actually be disadvantageous to propagate the filter. Thus, a non-bandwidth reducing filter will not be propagated unless there is some other reason to do so, e.g. economizing on processing time.

A special situation arises when there exist filters of the same type on all the outgoing links of an internal node in the dissemination tree. In this case, the filters can be combined into a single filter, which if bandwidth reducing, may now propagate further upstream. Figure 4 illustrates these concepts.

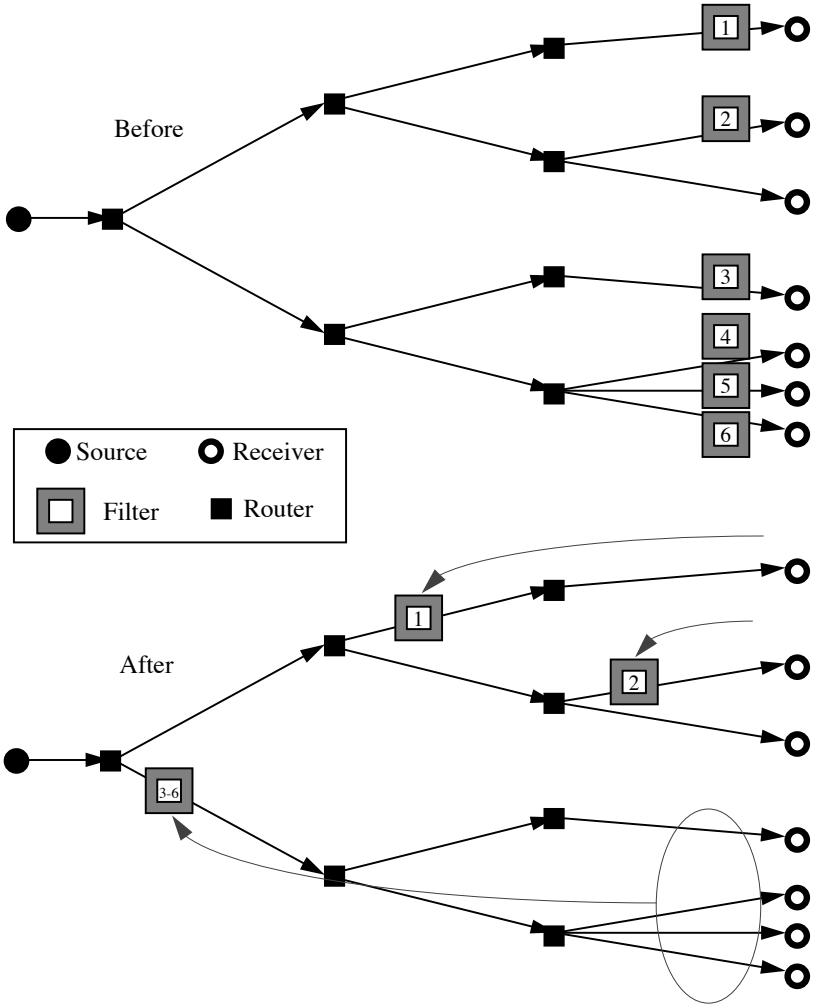

Figure 4: Example of filter propagation. All the filters are of the same type, and so they may possibly combine. Filters 1 and 2 propagate upstream but cannot combine because there is a receiver in that part of the dissemination tree which requires an unfiltered stream. Filter 3 propagates, and filters 4, 5, and 6 propagate, combine, and continue to propagate until they combine with filter 3, with the resulting filter (denoted by filter "3-6") finally propagating to the router closest to the source.

For the network to support the execution and propagation of filters, the executable code for filters must be installed in routers and filter names must be standardized. Any network router may provide an environment for the local execution of a filter. The required network infrastructure is similar to that required for supporting remote procedure calls.

5. The Filter Propagation Protocol

Filter propagation is established through a *filter propagation protocol*. The protocol is straightforward, and implements the scheme described above. Whenever a receiver issues a filter to be placed on a channel, some or all of the routers comprising the nodes of the channel's dissemination tree take part in the propagation protocol.[3] The receiver's machine first contacts the upstream router[4] to determine whether propagation is possible, notifying the router of the particular filter and the stream to which it is being applied. The router accepts the filter if (a) it recognizes the filter's name and therefore has the filter's executable code installed, and (b) it has the resources (e.g. processing time, memory buffers) to execute the filter. The latter decision is one of local policy established by the router's manager; how such policies are formulated is outside the scope of this paper. If the router accepts the filter, then it may apply the filter on a stream segment just before it is forwarded.

A router may be required to forward a stream segment over multiple outgoing links, i.e. a one-to-many forwarding operation. Thus, it is important to note that a filter's *location* is associated with a particular outgoing link of a router (or more generally, with a subset of all the outgoing links relative to the given dissemination tree) since filters are initiated by receivers in the dissemination tree, and not the sender [3]. If an outgoing link's downstream receiver initiated the filter propagation, the filter is located on that link (see Figure 5).

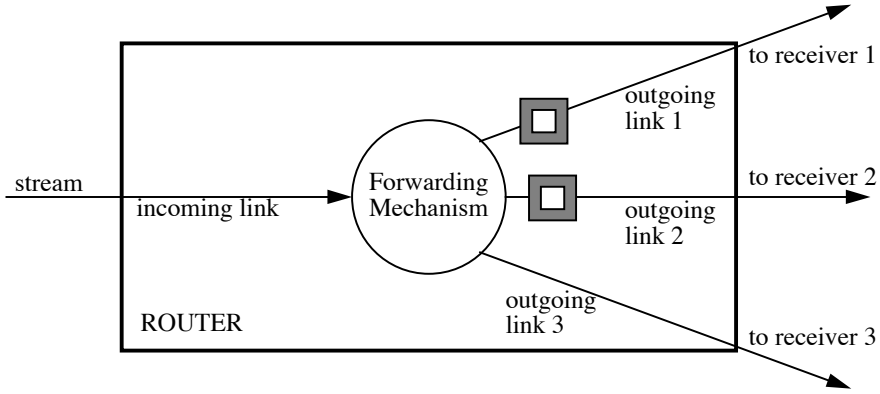

Figure 5: Filter location is associated with a particular outgoing link of a router. Receivers 1 and 2 each placed the same type of filter on the same channel; both filters have propagated to this router, and are located at outgoing links 1 and 2. A segment is (physically) filtered only once; the same filtered segment is then transmitted multiple times, over outgoing links 1 and 2. Since receiver 3 did not place a filter on the channel, the original unfiltered version of the segment is transmitted over outgoing link 3.

While the same filter may be located at multiple outgoing links, only one physical copy of the filter exists. And, if the same filter is to be applied to the same segment multi-

3. This assumes that it is advantageous to propagate the filter upstream, e.g. resulting in bandwidth reduction. This is specified as part of the filter definition.

4. There is exactly one upstream router relative to any node in a dissemination tree.

ple times, only a single execution of the filter on the segment occurs; the filtered segment is then forwarded multiple times, each time over a different outgoing link.

In addition to deciding whether to accept a filter, the router determines whether it should try to propagate the filter to the next router upstream. If the same filter is located on all the outgoing links, then the filter is propagated. In the trivial case of a single outgoing link, the filter is always propagated. The protocol is recursive in that, at this point, the same steps described above are repeated with the router and the next upstream router as the communicants. When the "next upstream router" is actually the channel's source, the protocol terminates after carrying out the above steps.

6. Operating System Issues

The filters presented here closely resemble the "stream modules" described in [5]. While some of the concepts and even the interface are in principle similar, the issues of relocatability and automatic propagation cause special concerns. In particular, propagated filters will execute on machines (e.g. routers) which are shared resources, and can have a significant impact on their performance; these effects may be felt by other users of the network.

Given the desired capabilities of selection, transformation, and mixing, a filter requires varying degrees of processing power. In a network where processing power may be limited (e.g. ATM networks), only selective filters may be supported. However, even selective filters provide substantial flexibility, especially when the source can break down continuous media data into many basic component streams which are amenable to selection.

Transforming and mixing filters require actual computation. If the computation is substantial, it is useful for the filter's execution to be preemptive in case higher priority work needs to be done. Consequently, a transforming or mixing filter may be encapsulated by a preemptable process whose execution is triggered by the arrival of a segment. Using processes allows the standard resource management techniques for the scheduling of the CPU and memory to be applicable.

Ultimately, these controls on resource usage are based on the following principle: the network must always have discretion over how much work it is willing to carry out on behalf of a receiver. Otherwise, the network is left open to abuse or, more simply, may perform poorly any time a router becomes overloaded with work. Consequently, we believe that filter execution inside the network should always be optional and therefore must be considered unreliable by the receiver.

To resolve this problem, if a filter propagates into the network an additional copy of the filter can remain located at the receiver. When a segment arrives at the receiver, the receiver's filter determines whether the segment is already filtered; if not, the filter operates on the segment. Thus, it must be possible to inspect a segment and determine whether a filter has been applied to it or not. This is achieved by including bits in the segment to be used as tags indicating filtered or unfiltered, with each bit dynamically assigned to the different filters placed on the channel.

We are currently considering two strategies for locating filters at multiple nodes of a dissemination tree:

- Locate a copy of the filter at the most upstream router (as determined by the filter propagation protocol) as well as at the receiver. If the most upstream router decides not to apply the filter to the stream, the filter located at the receiver must do so.

- Locate copies of the filter at each router along the path between the most upstream router and the receiver. As a segment is forwarded by these routers, any of them may apply the filter. Once a router applies the filter, the others will refrain from doing so. This provides a form of load balancing by giving the routers the flexibility to off-load work in response to local overloads.

7. The Network as a Processor

By executing filters "inside the network," the network may be thought of as a "processor." All processors have instruction sets: the set of all types of filters may be thought of as the network processor's set of instructions. The creation of this network processor instruction set is based on the creation and adoption of new filters to be supported on routers.

We see the process of building this instruction set as an evolutionary one. Anyone may design a new filter and propose it for adoption. The filter's designer would provide a description of the filter's function, the source code, and a unique name (some mechanism is required to resolve naming conflicts). The filter is then submitted for acceptance, being distributed to the various organizations which own the network's routers.

Thus, decisions to accept a filter as part of the instruction set are made in a decentralized and incremental fashion. The manager of a router makes such a decision for that router. If accepted, the manager installs the filter on the local router by arranging for the filter source code to be compiled for local execution and to be recognized by name when propagation occurs. Such filters can then execute on that router.

If the filter is not accepted (or for any reason not installed on a router), it must execute elsewhere. By default, the filter can always execute at the receiver, so there is never a problem of not finding a place to execute the filter.

We expect that over a period of time, a collection of useful filters will be developed which can be expected to be supported by a large set of routers.

8. Conclusions

We have described how relocatable continuous media filters can be used to trade off bandwidth and processing in networks transporting continuous media. Continuous media, such as video, have high bandwidth requirements, and so efficient allocation and management of bandwidth is important. Distributed applications with continuous media I/O components have several characteristics which can be used to advantage: many are dissemination-based, and they can accept continuous media of varying levels of resolution. Filters can be used to lower resolution, either statically when a receiver's resolution response is limited, or even dynamically based on network overloading. By propagating filters upstream, and by combining like filters at fork nodes in a dissemination tree, bandwidth utilization is effectively reduced.

9. References

[1] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *CACM*, vol. 34, no. 4, April 1991.

[2] D. Cheriton, "Dissemination-oriented Communication Systems," Distinguished Lecturer Series, Department of Computer Science and Engineering, University of California, San Diego, April 1992.

[3] J. Pasquale, G. Polyzos, E. Anderson, and V. Kompella, "The Multimedia Multicast Channel," *Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, November 1992 (also in *Lecture Notes in Computer Science #712*, pp. 197-208, Springer-Verlag).

[4] V. Kompella, J. Pasquale, and G. Polyzos, "Multicast routing for multimedia communication," *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, pp. 286-292, June 1993.

[5] D. M. Ritchie, "A stream input-output system," *AT&T Bell Laboratories Technical Journal*, vol. 62, no. 8, pp. 1897-1910, October 1984.