Kommunikation in Verteilten Systemen 1995 (KiVS)', 22.-24.02 1995, TU Chemnitz-Zwickau, Germany

Filtering Multimedia Data in Reservation-Based Internetworks

Lars C. Wolf, Ralf Guido Herrtwich, Luca Delgrossi

IBM European Networking Center
Vangerowstraße 18 • 69115 Heidelberg • Germany

Phone: +49-6221-59-3000 • Fax: +49-6221-59-3400

{lwolf, rgh, luca} @ vnet.ibm.com

Abstract: Multimedia applications with a large number of recipients of a single multimedia stream require support of different quality-of-service levels for different receivers. For hierarchically encoded streams, the filtering of substreams in routers has been proposed as a mechanism to achieve this goal. This paper discusses the effects and mechanisms of filtering in the context of Internet reservation protocols such as ST-II and RSVP. It introduces a new flow specification for hierarchically encoded streams and describes its processing in endsystems and routers.

1 Introduction

Several distributed multimedia applications such as video conferencing or video lectures must support multiple receivers. For applications with many participating receivers and for applications which transmit their data across a wide geographical range, there exists a need to support receivers and intermediate transmission paths with different capabilities.

An approach to support heterogeneous receivers in multimedia applications is to use *filter* mechanisms where only a subset of the full information is presented to the end user on the receiving side. Data which are not presented are stripped off from the original data stream at some intermediate agents, for example routers. Thus, the source always emits a full stream, but the stream is possibly scaled to a stream with lower quality.

In [10, 11] Pasquale has introduced filters as a general concept. His filters would allow a system to perform arbitrary operations on multimedia data in any part of the network. They can, for example, be used to transform one encoding format to another. Although the generality of the model is appealing, it can lead to several problems: long processing times may increase communication delays, security aspects may prohibit users from down-loading code for arbitrary filters into routers, and not all intermediate nodes, for example ATM switches, may be suited to provide the required processing capabilities.

In this paper, we consider the use of filters for the purpose of packet discarding only, i.e., we understand filtering in the true sense of the word. We assume that this filtering takes place in the network layer since only on this layer can all intermediate nodes in a communication path be known. Other "filtering" operations such as mixing audio streams can be accomplished in higher layers using mechanisms such as RTP's bridges [12]. We present a method to add filtering to internetwork reservation protocols such as ST-II [13] and RSVP [16]. It should also be possible to map the described approach to other multimedia setup protocols.

The outline of the paper is as follows: Section 2 introduces our filtering approach. Section 3 summarizes the changes that need to be made to the flow specification of a stream to accommodate filtering. Section 4 explains how the changes affect the processing of flow specifications and multimedia streams.

2 Filtering a Multimedia Stream

Deciding which parts of a multimedia stream to forward and which to filter out can only be made with respect to the data encoding scheme used. We distinguish between two classes of encoding formats:

- In *independently* coded streams, higher-quality parts are substitutions for lower-quality parts. For example, one substream S_1 may contain complete images of the size a*b and another substream S_2 may contain complete images of the size 2a*2b. To choose a different quality means to choose a different substream.
- In *hierarchically* encoded multimedia streams, higher-quality parts are additions to the lower-quality parts. For example, one substream S_1 may contain images of the size a*b and another substream S_2 may contain all *additional* pixels that extend the format to 2a*2b. To present data in the highest quality, all substreams must be presented.

Only in the hierarchical case we believe filtering to be an appropriate technique. To allow the system to better take into account the relation between the substreams, we find it appropriate to associate a single connection with all substreams. For instance, having *one* connection means automatically that all substreams are transmitted using the same route through the network, while the use of several connections can lead to different routes introducing resynchronization problems.

In the case of independently coded substreams, different streams are sent to the various targets. Since the streams are independent, synchronization problems do not occur if there is one connection per substream. We have described such a technique in [1]; in this paper we concentrate on the former case.

Hierarchically encoded streams will play an important role in the future of multimedia systems. New data formats such as MPEG-II [8] use hierarchical encoding to achieve different levels of presentation quality. These levels result from scaling the original video data in several dimensions. Gonzales and Viscito describe the following techniques for this purpose [4]:

- *Spatial scaling:* a multiplicity of spatial resolutions.
- *Rate scaling:* a multiplicity of picture rates. This is already part of MPEG-I [6, 7] via the division into *I* (intra-frame coded), *P* (predicted coded), and *B* (bidirectional predicted coded) frames. This kind of scaling is often also referred to as *temporal* scaling.
- *Amplitude scaling:* multiple versions of a picture with varying fidelity at the same spatial and temporal resolution. This is also specified as *frequency* scaling, either via data partitioning, in which the discrete cosine transformation (DCT) coefficients are separated into several regions and the regions are handled differently, or via signal-to-noise-ratio (SNR) scaling, in which the least significant bits of the DCT coefficients are separated from the most significant bits and separately handled.

For any of these scaling methods, all substreams together yield an image of the best quality. There is a well-defined order to filter out substreams, should it become necessary.

2.1 Substream Identification

Routers need information on data packet contents in order to be able to drop portions of a stream. We opted for a simple scheme where each data packet is *tagged* by the source by assigning an appropriate value to a field of the packet’s header. If data segmentation is possible, segments should be tagged in the same way as the original PDU.

In the case of ST-II, such a field is already available, so that, in this specific case, no change to the protocol is required. The data priority[1] field can be used, which consists of 3 bits, allowing for 8 different levels. We feel that 3 bits provide the support we need to implement the filtering functions we have in mind and that 8 levels are sufficient to experiment with this technique. In the future, we expect that perhaps more bits might be necessary. By convention, we say that the most important substream shall be tagged with the lowest value (0) and the least important with the highest value (7). Routers will use this information during the resource reservation (see Section 4 below).

An alternative to packet tagging is to let the application specify a *pattern* which can be identified by the router. The router analyses each packet to check if it matches with the given pattern. The advantage of this approach is that any kind of pattern can be specified and thus a very large number of applications can be supported. For instance, it would be possible to filter packets generated by old applications which have no notion of filtering, such as telnet. On the other hand, this approach has also some inconveniences, because of the longer time required by the source to specify the different patterns and by the routers to match them.

Implementing this pattern matching technique in ST-II would require modifications to the header of the data PDU to include additional information. The current header fields of an ST data packet (if the data priority field is not used as in *tagging*) do not allow for pattern matching because they would be set to the same values for the whole stream.

The RSVP setup protocol does not define the format of the data PDUs to be transferred. Thus both tagging and pattern matching can be implemented as long as the chosen data format permits it.

2.2 An Example

As an example, let us consider in the following a stream S, coded with a hierarchical scheme. The full stream consists of 3 substreams, the base substream S_0, and the additional substreams S_1 and S_2:

1. Substream S_0: (rate = 0.4 Mbit/second, substream id = 0)
2. Substream S_1: (rate = 0.6 Mbit/second, substream id = 1)
3. Substream S_2: (rate = 1.0 Mbit/second, substream id = 2)

Substream S_0 delivers the base information and is, therefore, the most important. Thus it is assigned the lowest value (substream id = 0). Substreams S_1 and S_2 get values 1 and 2 respectively. By combining substreams S_0, S_1, S_2, it is possible to build three different quality streams to be presented:

1. Stream Q_0: (low quality, contains substream S_0 only, requires 0.4 Mbit/second).
2. Stream Q_1: (medium quality, contains substream S_0 & S_1, requires 1.0 Mbit/second).
3. Stream Q_2: (high quality, contains substream S_0, S_1 & S_2, requires 2.0 Mbit/second).

Figure 1 illustrates a simple scenario with several hosts participating in the transmission of the streams. The data flows from the source H_0 to the three destinations located at H_2, H_3, and H_5 through routers H_1 and H_4. In our example, each destination decides to receive a stream of different quality, corresponding to different portions of the data. While the target at host H_2 would like to receive the full quality stream Q_2 (thus, all available substreams), targets H_3 and H_5 need not so high a quality, and therefore they use substreams S_0 & S_1 (yielding Q_1), and only S_0 (resulting in Q_0), respectively.

If the stream of this example was, for instance, an MPEG stream, intra-coded images (I-frames) could be carried by substream $S_{0,}$ providing the base information and thus the low-quality stream. Predictive frames (P-frames) and bidirectional predictive frames (B-frames) could be carried by substreams S_1 and S_2, respectively, and the medium-quality and high-quality streams could be obtained by adding S_1, or both S_1 and S_2, to S_0.

1. The word "priority" seems to imply that high-priority data shall be processed first with respect to low-priority data, and therefore it is misleading. We prefer to call this field "substream identifier".

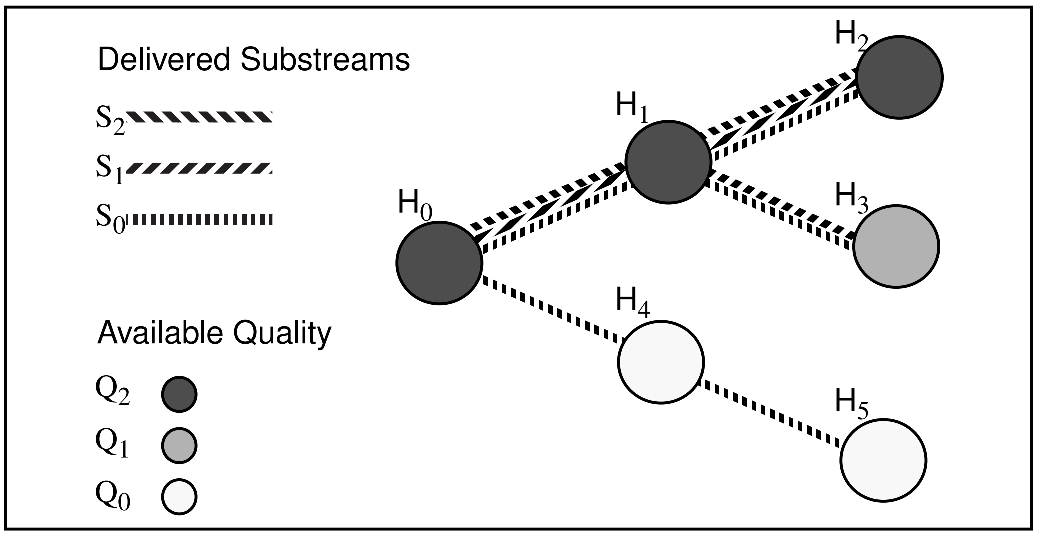

Figure 1: Different qualities created from three substreams

3 Specification of Substream Characteristics

Protocols that make use of resource reservation, such as ST-II and RSVP, need to be provided with information about the quality-of-service (QoS) requirements of a stream. With filtering, they also need to be informed about the amount of bandwidth required by each substream. This information needs to be carried to all the routers and targets, so that they know which lower-quality substreams can be derived from the full stream.

In reservation protocols, the natural way of distributing information to all the routers and targets is to include it into the *flow specification* (*FlowSpec*). The difficulty consists of finding an efficient way of describing the various substreams. When the number of possible substreams is limited to 8, as assumed above, providing a list of the substreams is still acceptable. For larger numbers, more efficient schemes should perhaps be elaborated, possibly by using regular expressions. However, we doubt that arbitrary quality levels will be needed in practice.

3.1 The ENC FlowSpec

Before we enter the discussion, we briefly introduce the FlowSpec used by our ST-II implementation. The intent here is to show how the single substreams can be described. Any other FlowSpec proposed in the literature (e.g., [9]) could have been chosen for the same purpose. The ENC (European Networking Center) FlowSpec contains the following QoS parameters:

- *diligence* (either guaranteed or statistical QoS)
- *maximum end-to-end delay* (in microseconds)
- *maximum message size* (in bytes)
- *message rate* (in messages/second)
- *workahead* (in messages/second)
- *reliability class* (determines error handling strategy)

The *diligence* parameter indicates whether “guaranteed” or “statistical” service is desired by the application. The “guaranteed” service provides the best guarantees, and can support applications with very strict requirements. It implies a conservative reservation scheme which may lead to poor resource utilization; therefore, our system offers also the “statistic” service where more optimistic assumptions are made, and small violations of the guarantees are allowed.

The *maximum end-to-end delay*, *maximum message size*, and *message rate* define the maximum transmission delay of a packet belonging to the stream, its maximum size, and the highest frequency at which an application may send packets without violating the QoS, respectively. Short-term violations of the rate are allowed; they are bounded by the *workahead* parameter. Finally, the *reliability class* parameter specifies whether transmission errors should be detected and how, if at all, they should be corrected.

3.2 Substream Description

There is no need to entirely replicate the FlowSpec for each single substream. In the following, we consider each parameter of the FlowSpec separately to verify whether replicated information for every substream is required.

Diligence: It is desirable that the diligence is specified on a substream basis. For instance, an application may require guaranteed QoS provision for the base substream or a small set of substreams only. In a video conference, it would be possible to request guaranteed service for a black-and-white quality substream while statistical service could be sufficient for the better quality in colors.

Maximum End-to-end Delay: A common specification for all substreams is sufficient for the maximum end-to-end delay parameter. The end-to-end delay of the full stream is shared by all substreams since a larger delay for one substream would increase the delay of the full stream.

Maximum Message Size and **Message Rate:** These two parameters define the total amount of bandwidth required by each substream. They need to be specified on a substream basis.

Workahead: For workahead, it has to be considered that the full stream S consists of a repetition of substream ($S_0 \ldots S_{n-1}$) packets such as

$$S = S_0S_1\ldots S_{n-1}S_0S_1\ldots S_{n-1} \ldots S_0S_1\ldots S_{n-1} = (S_0S_1\ldots S_{n-1})^*$$

or as another example

$$S = (S_0S_1S_2S_2S_1S_2S_2)^*$$

Thus, the workahead W_i of a substream S_i with a rate R_i is a simple function of a base workahead $\underline{W}$ and does not need to be specified separately. It can be specified as $W_i = (\underline{W} * R_i) / N$, where N is a 'normalization' factor with a value of 1000. Rates will always be smaller than that value, thus, the workahead per substream W_i can be specified with fine granularity.

Reliability: Applications may need to assign different reliability classes to different substreams. The reliability for substreams may be different because of the different importance of a substream with respect to the full stream. While an application may be interested in error indication or correction for the base stream, it will cause less problems if data from lower priority substreams is lost. In this case, the lowest reliability class may be used. In an MPEG stream which might be scaled in the temporal dimension, reliable transfer should probably be chosen for I-frames, because when an I-frame is lost the following P- and B-frames cannot be calculated. The same argument applies to a stream scalable in the SNR dimension; here, the MSB substream has to be transmitted with a high reliability class while for the LSB part a lower reliability class can be used.

3.3 A FlowSpec for Substreams

For each substream, a *SubFlowSpec* including the following parameters is needed:

- diligence,
- maximum message size,
- message rate,
- reliability class.

In addition to these parameters, the number of substreams that exist has to be specified.

7	*15*	*23*	*31*
f_pcode	f_pbytes	f_version	f_pad
f_delay (maximum regular transit time)			
f_accum_delay (minimum transit time)			
f_workahead		f_substreams	f_filter
f_msg_size_first			
f_rate_first		f_dil_first	f_rel_first
...			
f_msg_size_last			
f_rate_last		f_dil_last	f_rel_last

Figure 2: Flow Specification including substream descriptions

The resulting FlowSpec for a stream and all its substreams is shown in Figure 2. The meaning of the fields are:

f_pcode:	Identifies this parameter as a FlowSpec.
f_pbytes:	FlowSpec length.
f_version:	Version number; for this FlowSpec version number 5 is used.
f_pad:	Padding.
f_delay:	Stream maximum end-to-end delay (microseconds).
f_accum_delay:	Delay accumulated between origin and previous host (microseconds).
f_workahead:	Stream workahead W (messages).
f_substreams:	Number of substreams (integer in the range 2–8).
f_filter:	Filter (see below).

The last fields contain substream information:

f_msg_size_first:	Maximum message size for substream S_0 (bytes).
f_rate_first:	Rate for substream S_0 (messages/second)
f_dil_first:	Diligence for substream S_0 (guaranteed or statistical).
f_rel_first:	Reliability for substream S_0 (reliability class identifier).
f_msg_size_last:	Maximum message size for substream S_{n-1}.
f_rate_last:	Rate for substream S_{n-1}.
f_dil_last:	Diligence for substream S_{n-1}.
f_rel_last:	Reliability for substream S_{n-1}.

Substream identifiers are assigned in the order of appearance in the FlowSpec. The first substream has identifier 0. At least two substreams have to exist when using this FlowSpec. A maximum of 8 substreams is allowed.

The f_filter field indicates which substreams should be forwarded by a router. A value of 2, for instance, indicates that the two most important substreams (S_0 and S_1 in the example above) should be forwarded. Data belonging to other substreams is dropped.[2]

4 Use of the FlowSpec

It is important to specify how a FlowSpec should be handled and interpreted when it is defined. Not all mechanisms that are useful for a complete FlowSpec are automatically appropriate for

2. One could consider allowing the selection of non-contiguous substreams, choosing substreams S_0 and S_2 and emitting S_1, for example. However, we believe that the gain in flexibility does not outweigh the introduced complexity.

SubFlowSpecs. In particular, SubFlowSpec negotiation cannot be as general as FlowSpec negotiation since the dependencies of the different SubFlowSpec must be considered.

4.1 Stream Establishment

A useful method to shorten the FlowSpec negotiation when a stream is established is to provide a range of acceptable values and not just a single target value for each FlowSpec entry. This leads to different QoS levels, for example, by providing different frame rates in a video stream.

We feel that such a mechanism would be too complex with the SubFlowSpecs. Instead, we exploit the fact that streams are hierarchically encoded and that it is possible to derive different quality substreams from them. This leads to discrete quality levels rather than a continuous range and yet is appropriate for many multimedia applications. The following algorithm is easy to implement in any router.

Let a stream consist of 3 substreams, as in Figure 1. A router receiving a CONNECT message extracts the FlowSpec and gives the information about the characteristics of the specified streams to its resource management system (RMS) [15]. The RMS attempts to reserve the resources for the streams from highest quality to lowest quality and returns information about the substreams which can be handled by the router. Thus, it tries to make a reservation for the full stream first. If this succeeds, the router will deliver the full stream to its next-hop. If the reservation fails because not enough bandwidth is available, it will try to reserve for the medium-quality stream only. If this succeeds the router will forward packets belonging to S_0 and S_1 and drop packets belonging to S_2. If the reservation fails again, it will try to reserve for the low-quality stream only. Upon success, the router will forward packets belonging to S_0 and drop packets belonging to S_1 and S_2. If even the low-quality stream cannot be established, the router disconnects the correspondent targets.

4.2 Stream Acceptance

In ST-II, if the target side of an application decides to accept a new stream,[3] it replies with an ACCEPT message containing an updated FlowSpec. If the application is not willing to receive the full stream, it needs to specify which substreams should be received. This can be done by updating the f_filter field of the FlowSpec. For example, should the application be interested in receiving substreams S_0 and S_1 only, it would indicate this by setting f_filter to 2. Also, the message rate field relative to substream S_2 is set to 0 to indicate that substream S_2 should not be received.[4]

A router which transfers an ACCEPT message to the origin of the stream stores the FlowSpec information (f_filter) given in the ACCEPT messages, together with the original FlowSpec transmitted in the CONNECT message.[5] While the information from the modified FlowSpec is needed to prepare the specified filter, the original FlowSpec is needed if later a new target wants to join the existing stream and would like to get a better quality than the established targets and if the router supports the receiver-oriented connection establishment as described in [2].[6]

3. The characteristics of the stream are specified by the FlowSpec received as part of the CONNECT message.

4. First, we considered also that distinct targets might be willing to receive a substream with a different rate, thus, changing the rate field to the desired value. The router would drop packets which are not required with respect to the 'subrate.' While this introduces flexibility, we think it adds too much complexity in the routers.

5. The only information from the ACCEPT message which has to be stored is the f_filter value since it is the only field indicating the requested stream information. Of course, for links without directly attached targets a router stores only a combined f_filter describing the requested substreams to be forwarded via that link instead of the information per target.

For RSVP a similar amount of information has to be stored. The PATH message includes in the FlowSpec the information about the stream characteristics, and the RESERVATION message contains the reserver information (for the dynamic filtering style).

4.3 Filter Placement

From the information a router has about its targets, two kinds of filters can be computed:

- an *up-stream filter* which specifies which information has to be received by this router and
- a *down-stream filter* which specifies for each target which information has to be forwarded.

The up-stream filter must be established not at the router itself, but at the system up-stream from the router, in order to reduce the amount of data transferred via the connecting link. Thus, the information about the required streams has to be transmitted to the up-stream system. The down-stream filter is located at the router itself and discards packets if no target reached via this link has indicated interest. Only if a multicast connection is used, a filter may be placed on the down-stream system.

In the multicast case, the router receives from the targets (connected to the shared media network) the ACCEPT messages through the same link, they specify requests for different numbers of substreams. Thus this situation does not differ (for the router) from other scenarios and no specific mechanism is needed on a router. Systems reached via a multicast must, however, create a filter. Thus the question is raised how these systems learn about the "filter creation requirement". Since the use of multicast in networks such as Token Ring or Ethernet requires the establishment of a multicast group (for instance, using a scheme as described in [14]), the knowledge is already available on all systems participating in the multicast transmission. For a "sanity check" on a received stream, the simple filter on a down-stream system may always be set; it merely drops received packets with a higher substream number than requested. (See also the processing steps shown in Figure 6.)

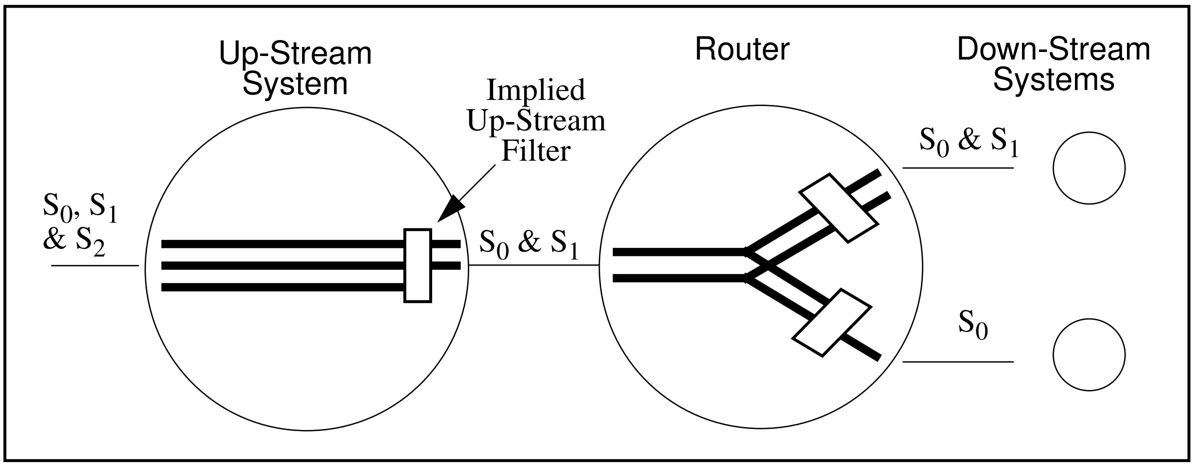

Figure 3: Filter placement

A scenario is shown in Figure 3 where the source on the left offers three substreams S_0, S_1, and S_2. The down-stream targets on the right side request S_0 and S_0 with S_1. This means that the router must receive the two substreams S_0 and S_1. On the router, down-stream filters for the three down-stream systems are established. S_2 is filtered out at the up-stream system.

The FlowSpecs occurring in the scenario above are illustrated in Figure 4. The original FlowSpec describes three substreams. All substreams have a rate of 10 messages/second and should be transmitted with the lowest reliability class. On the way back from the targets to the origin, the FlowSpecs shown in the middle of the figure are received by the router. The first down-stream system indicates interest for two substreams. For the second down-stream sys-

6. Eventually, the router has to inform its up-stream system that a different filter has to be used.

tem, only one substream should be transmitted. The result from these desired substreams is that the FlowSpec which is forwarded from the router to the up-stream system requests two substreams. The third substream can be completely filtered already at an up-stream system.

Original FlowSpec

f_pcode = 2	f_pbytes = 40	f_version = 5	f_pad
f_delay (maximum regular transit time)			
f_accum_delay (minimum transit time)			
f_workahead		f_substreams = 3	f_filter = 3
f_msgsize_first = 40960			
f_msgrate_first = 10		f_dil_first = G	f_rel_first = 0
f_msgsize = 61440			
f_msgrate = 10		f_dil = B	f_rel = 0
f_msgsize_last = 102400			
f_msgrate_last = 10		f_dil_last = B	f_rel_last = 0

FlowSpecs Received at Router from Down-Stream Systems

f_pcode	f_pbytes	f_version	f_pad
f_delay			
f_accum_delay			
f_workahead		3	2
40960			
10		G	0
61440			
10		B	0
102440			
0		B	0

f_pcode	f_pbytes	f_version	f_pad
f_delay			
f_accum_delay			
f_workahead		3	1
40960			
10		G	0
61440			
0		B	0
102440			
0		B	0

FlowSpec forwarded from Router to Up-Stream System

f_pcode = 2	f_pbytes = 40	f_version = 5	f_pad
f_delay (maximum regular transit time)			
f_accum_delay (minimum transit time)			
f_workahead		f_substreams = 3	f_filter = 2
f_msgsize_first = 40960			
f_msgrate_first = 10		f_dil_first = G	f_rel_first = 0
f_msgsize = 61440			
f_msgrate = 10		f_dil = B	f_rel = 0
f_msgsize_last = 102400			
f_msgrate_last = 0		f_dil_last = B	f_rel_last = 0

Figure 4: Exchanged FlowSpecs

4.4 Filter Administration

Which data of a stream must be received by a router? It is the set of all substreams of this stream for which some down-stream system h has indicated interest, and since that specification (via f_filter) is ordered, we can simply use $F = max(\text{f\_filter}_h)$. Other packets do not need to be transmitted from the up-stream system to this router.

The steps necessary (beyond the usual control protocol processing) are shown in Figure 5. As part of the connect and accept message processing, the information from the FlowSpecs must be stored and updated to reflect the requests of the down-stream targets.

Stream Set-Up Processing

```
switch (SCMP packet type):
[…]
case CONNECT:      store FlowSpec for this stream as original FlowSpec;
                   forward CONNECT message;
                   break;

case ACCEPT:       store FlowSpec for this stream as modified FlowSpec for this target;
                   create/update up-stream filter:
                           create/update F, the maximum of f_filter specified by all
                              targets for this stream;
                   create/update down-stream filter for this target;
                   forward ACCEPT message with a FlowSpec which reflects F;
                   break;

case DISCONNECT:           /* disconnect desired by origin */
case REFUSE:               /* disconnect desired by target */
                   remove per target FlowSpec for this stream;
                   update up-stream filter information:
                           update F, the maximum of f_filter specified by all remaining
                              targets for this stream;
                   remove down-stream filter for this target;
                   forward DISCONNECT message with a FlowSpec which reflects F;
                   break;
[…]
```

Figure 5: Reservation protocol processing

As shown in Figure 6, during the data protocol processing it is necessary to check whether a filter exists for the stream the packet belongs to, and whether this filter has to be applied to the packet. Further processing is done as in a system with no filter mechanism.

4.5 Dynamic Changes

Our algorithms take into account most cases of dynamic changes to streams. If an additional target wants to receive the stream and the origin sends out a CONNECT message, the necessary functions are already in place: while forwarding the CONNECT, a router finds the stored FlowSpec; when processing the ACCEPT, the selected substream information F is updated. The ACCEPT message reflecting this value is forwarded in the up-stream direction, adding new substreams only if the new target required these.

For receiver-oriented stream joining as described in [2], the target contacts a router which sends the CONNECT directly without involvement of the origin. If the new target would like to receive a better quality than the other targets, additional substreams are required. The router informs up-stream systems by sending a NOTIFY message with a FlowSpec reflecting the new set of substreams. Then the first system which receives this NOTIFY, and which forwards the desired substreams already via a different link, simply changes its filter and drops the NOTIFY message.

Stream Processing

```
/* data received and should be forwarded to down-stream system */
if (a filter is set for this down-stream system) {
            if (packet substream id > F, the maximum of f_filter specified by a target) {
              discard packet;
            }
} else {
            forward_packet;
}
```

If the default value of *F* is set to the number of possible substreams (currently 8) this can even generally be written as:

```
if (packet substream id > F) {
            discard packet;
} else {
            forward_packet;
}
```

Figure 6: Data transfer protocol processing

If a target disconnects from a stream, it sends a REFUSE message up-stream. If it was the only target receiving a particular substream, the next router detects (while processing the REFUSE) that the new value for *F* is smaller than the old value and forwards a REFUSE containing the new *F*. The up-stream router can therefore change its filter to a coarser one.

If the origin wants to disconnect a specific target (without tearing down the whole multiple target connection), it sends a DISCONNECT down-stream. If a router detects that the remaining targets require fewer substreams than before (the target to be disconnected was the only one receiving a particular substream), it can inform its up-stream router about the changed filter using the above described NOTIFY message.

Changes to the FlowSpec of the stream can be done as usual, for example in ST-II via CHANGE or CHANGE_REQUEST messages. For instance, if a target decides after some time that it wants to receive a different set of substreams (and the required resources are available in case more substreams are selected), it transmits a CHANGE_REQUEST message.

5 Conclusion

We have shown how to implement a simplified version of Pasquale’s filter concept in a reservation-based internetwork running ST-II or RSVP. The filtering functions we use are sufficient for many multimedia applications because they:

- are easy to implement and fast to execute,
- do not depend on knowledge about encoding formats in routers,
- do not require users to download filter code into routers, and
- work nicely with common encoding schemes for digital video.

Just like the scaling function at the transport level described in [1], the network-layer filtering is yet another mechanism to make our multimedia communication system HeiTS (Heidelberg Transport System) [3, 5] an even more useful tool for exchanging digital audio and video across internetworks.

We intend to propose the mechanisms contained in this paper to the IETF working group which is specifying a new version of the ST-II protocol.

References

[1] L. Delgrossi, C. Halstrick, D. Hehmann, R.G. Herrtwich, O. Krone, J. Sandvoss, C. Vogt: *Media Scaling with HeiTS*, ACM Multimedia 93, Anaheim, August 1993.

[2] L. Delgrossi, R.G. Herrtwich, F.O. Hoffmann, S. Schaller: *Receiver-Initiated Communication with ST-II*, to appear in Multimedia Systems Journal, ACM/Springer, also as Technical Report 43.9314, IBM European Networking Center, Heidelberg, 1993.

[3] L. Delgrossi, R.G. Herrtwich: *Real-Time Multimedia Communication with the Heidelberg Transport System*, submitted for publication.

[4] C. Gonzales, E. Viscito: *Flexibly Scalable Digital Video Coding*, Signal Processing: Image Communication, Vol. 5, No. 1-2, February 1993, pp. 5-20.

[5] R.G. Herrtwich: *The HeiProjects – Support for Distributed Multimedia Applications,* Technical Report 43.9206, IBM European Networking Center, Heidelberg, 1992.

[6] D. LeGall: *MPEG: A Video Compression Standard for Multimedia Applications*, CACM, Vol. 34, No. 4, April 1991, pp. 46-58.

[7] MPEG, ISO IEC JTC 1: *Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5Mbit/s*, International Standard ISO/IEC IS 11172, 1993.

[8] MPEG-II, ISO IEC JTC 1: *Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media, Test Model4*, Draft, MPEG 93/255b, February 1993.

[9] C. Partridge: *A Proposed Flow Specification*, Internet RFC 1363, September 1992.

[10] J. Pasquale, G. Polyzos, E. Anderson, V. Kompella: *The Multimedia Multicast Channel,* Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, November 1992.

[11] J. Pasquale: *Filter Propagation in Dissemination Trees: Trading off Bandwidth for Processing in Continuous Media Networks*, Fourth International Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster University, November 1993.

[12] H. Schulzrinne: *RTP: A Transport Protocol for Real-Time Applications.* Internet Working Draft, 1993.

[13] C. Topolcic: *Experimental Internet Stream Protocol, Version 2 (ST-II)*, Internet RFC 1190, October 1990.

[14] B. Twachtmann, R.G. Herrtwich: *Multicast in the Heidelberg Transport System*, Technical Report 43.9306, IBM European Networking Center, Heidelberg, 1993.

[15] C. Vogt, R.G. Herrtwich, R. Nagarajan: *HeiRAT: The Heidelberg Resource Administration Technique - Design Philosophy and Goals*, Kommunikation in verteilten Systemen, Munich, March 1993.

[16] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala: *RSVP: A New Resource ReSerVation Protocol*, IEEE Network, September 1993, pp. 8-18.