# An RTP to HTTP Video Gateway

Mathias Johanson
Framkom Research Corporation
Sallarängsbacken 2, S-431 37 Mölndal, Sweden
mathias@framkom.se

## Abstract
Multicast audio and video conferences are today commonplace in certain parts of the Internet. The vast majority of Internet users, however, are not able to participate in these events because they either lack multicast network connectivity, are located behind firewalls, have insufficient network resources available or don't have access to the proper software tools. In many cases all of these restrictions apply. This paper presents an effort to extend the scope of multicast video conferencing by the development of an Internet video gateway that interconnects multicast networks with the World Wide Web. The overall design of the gateway software is outlined and a novel algorithm for rate control of the multicast video flows is described. Some performance tests that show the efficacy of the system in terms of resource utilisation and scalability are presented.

## Keywords
Multimedia gateways, Multicast, Video, RTP, HTTP

## 1. INTRODUCTION
The explosive growth of the Internet has so far mostly been related to its success in supporting asynchronous applications like WWW-browsing and file transfers. Within the research community, however, the Internet has for many years been successfully utilised in supporting synchronous multimedia conference sessions, most notably within the Mbone initiative [1]. The Mbone is a virtual network implemented on top of the Internet that enables multicast packet delivery; a technology crucial for implementing scalable multipoint communication systems. Nevertheless, the vast majority of Internet hosts are not connected to multicast-enabled networks, so inter-operation with Mbone-type services need some sort of gateway function or tunnelling mechanism that can forward IP multicast datagrams in a controlled manner over unicast network connections. Several software tools have been designed for this purpose, including mrouted [6] and mTunnel [3], but there are still other difficulties that need to be overcome to make audio and video conferencing ubiquitous on the Internet. One difficulty is that the bandwidth available on many dialup links is too low to sustain the potentially broadband traffic of audio and video sessions. A solution to this problem is to employ media transcoding gateways that convert the transmitted media to a lower bandwidth format suitable for transmission over low-bandwidth links. One such approach is presented in [2]. Yet another obstacle is the fact that many Internet hosts are located behind firewalls. In the general case firewalls don't allow UDP-based real time traffic to pass through and in many cases they also employ techniques like network address translation that complicate end-to-end real-time communication. Moreover, the rather sophisticated applications required for real-time audiovisual communication might not be available on every computing platform and troublesome installation and configuration procedures will in any case restrain the applicability of the services in question.

This paper presents a novel software tool that has been developed to partially circumvent the aforementioned impediments to extend the range of synchronous multimedia communication.

## 2. BACKGROUND AND MOTIVATION
Synchronous collaboration tools like audio and video conferencing applications are becoming increasingly more popular on the Internet. Simple synchronous communication tools like ICQ [4] and IRC [5] have rapidly reached a large number of users due to their applicability virtually anywhere on the Internet. This is due to the fact that they rely only on the core protocols of the Internet (TCP/IP) and require very little network resources to be useful. Sophisticated multimedia collaboration software on the other hand require substantially more bandwidth and build largely on protocols that are not supported everywhere on the Internet (IP multicast [7], RTP/RTCP [11], UDP [13]). Although these technologies are expected to reach an increasingly more widespread deployment, there will always be heterogeneity in terms of network resources and services. In an effort to extend the scope of multicast video conferences we have developed an RTP to HTTP gateway software that makes it possible for an Internet user to take part of multicast video streams, albeit at potentially high latency and low frame-rate, with the only prerequisite being access to the WWW through a standard browser. Figure 1 shows an example configuration of a network that connects WWW users to a multicast network.

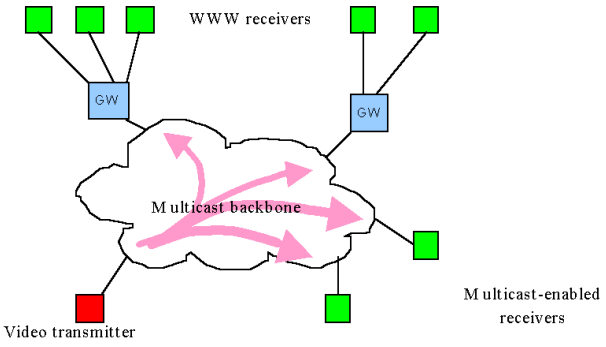


**Figure 1. Typical network configuration using gateways**


*WWW10,* May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

Note that the video gateway presented in this paper only enables users to receive video streams of multicast conference sessions. It doesn't provide any support for transmitting video to conference sessions.

## 2.1 Multicast Conferencing Tools

A suite of tools generally referred to as "the Mbone tools" have been used for some time on the global experimental multicast network known as the Mbone. The Mbone tools include real-time audio and video conferencing applications, shared whiteboards, text chat tools and more. These tools communicate using IP multicast group addresses and encapsulate real-time data in IP datagrams as specified by the *Real-time Transport Protocol* (RTP) [11], and the associated *RTP-profiles* for various media encodings. Basic session management and control as well as miscellaneous status report functions are handled by the *Real Time Control Protocol*, RTCP [11]. In addition, the *Session Announcement Protocol (SAP)* [14] and the *Session Description Protocol (SDP)* [15] are used to announce the lifetime of multicast sessions and describe what media format will be used for each session.

## 2.2 Video on the WWW

Except for experimental systems within the research community, the first large-scale use of live video on the WWW was so-called web-cameras. A web-camera is a device that is attached to a web-server that transmits live video images to a WWW-browser using HTTP. Although HTTP was originally designed for strictly asynchronous applications, extensions have been developed to enable web-servers to send continuous media streams to the client browser. This is known as "push"-technologies or HTTP streaming. Another class of applications that has emerged on the WWW is media on demand servers that transmit pre-recorded media clips to the client browser using HTTP-streaming or some other streaming protocol.

## 2.3 Packet Video Gateways

The concept of active media processing within multicast networks as a solution to the network heterogeneity problem was pioneered by Turletti and Bolot in [16] and by Pasquale et al. in [17]. Amir et al. elaborate on these ideas in [2] with the presentation of an application level video gateway that performs transcoding between JPEG and H.261 RTP streams. A classification of active networking applications is given in [18], wherein a distinction is made between *transport gateways* that bridge networks with different characteristics and *applications services* that perform active processing of the transmitted data, such as transcoding of video streams between different encodings. In [19] Ooi et al. present an architecture for a programmable media gateway that can be remotely configured to perform user-defined processing of media streams.

# 3. WEBSMILE: OVERALL ARCHITECTURE

WebSmile is a software component that is installed on an ordinary web-server that is connected to a multicast capable network. The software gives users access to multicast RTP video streams through the web-server using HTTP streaming.

## 3.1 Client Side

Two different techniques are used to enable the client browser to display the video that is streamed over HTTP; an experimental MIME-extension [20] for displaying moving images and a Java applet. The MIME extension, known as *multipart/x-mixed-replace*, makes it possible to display sequences of JPEG or GIF images in an HTML page. Since it is not supported in all browsers, this technique is complemented with a Java video player applet that is downloaded from the WebSmile server.

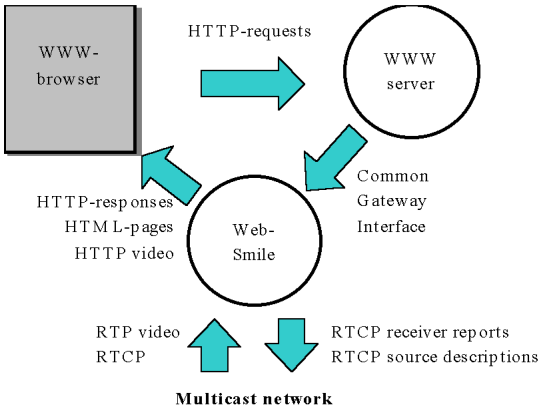


**Figure 2: Conceptual model of the WebSmile server architecture**

## 3.2 Server Side

The WebSmile gateway is implemented as a server program executed on a web server through the common gateway interface (CGI) [21]. The program performs three separate functions depending on the parameters with which it is invoked:

- Monitor a multicast session and report back information about the video sources that are identified.
- Join a session and return an HTML-page with video displays.
- Start forwarding video over HTTP.

The first function is performed by joining the multicast address and port specified and listening to RTCP source description (SDES) advertisements. The members of the session are identified by a canonical name in the format *user@host.domain* and optionally by more verbose information like a real name, address, phone number, etc. This information is reported back to the browser that originated the CGI-request as an HTML-form with a checkbutton associated with each identified session member. The user then indicates which video sources are to be monitored by checking the appropriate checkbuttons and posting the form back to the server. This invokes WebSmile in the second mode as described above to join the session and return the video display HTML page. This page contains a Java applet to display the video in case the browser has been identified (through CGI environment variables) as non-capable of displaying multipart/x-mixed-replace content. The third mode of WebSmile is invoked when the references in the video HTML-page to the HTTP-streamed video

are resolved. This is either an image hyperlink looking something like

<IMG SRC="http://server:port/cgi-bin/websmile?-s+1234+-a+224.2.2.2+-p+5566">

(where 1234 is the source id of the video to be monitored, 224.2.2.2 is the multicast address and 5566 is the UDP port number) or an applet connecting explicitly to the web server with the same CGI parameters. In both cases the video streamed over HTTP conforms to the multipart MIME specification with a content type of *image/jpeg* for each multipart entity.

## 3.3 Transcoding

In case the multicast video is not JPEG over RTP as specified by RFC2435 [12] the gateway needs to transcode the video into JPEG. Currently no transcoding support is implemented in WebSmile so only JPEG-compressed video will be forwarded. However, specialised transcoding gateways are available, including [2], that can be used in combination with WebSmile to support other formats.

# 4. RATE CONTROL

Since the bandwidth available for users connected through HTTP is, in most situations, expected to be less than the bandwidth used for the multicast sessions, rate control must be applied to the video traffic forwarded by WebSmile. This is performed by adapting the frame rate of the outbound video to the available bandwidth of each HTTP connection. The WebSmile gateway accomplishes this by writing video image data on the TCP socket of each HTTP connection until the socket buffer is filled. Images arriving on the multicast network while a socket is blocked (due to a full buffer) will not be sent to the corresponding client. When the socket is unblocked, forwarding of images is resumed. This modus operandi is simple to implement and will result in each client receiving video at a frame rate determined by TCP's flow control.

Considering the fact that the frame rate sustainable over the HTTP connections might be substantially less than the frame rate of the video being multicast, it would be desirable if the gateway could control the multicast video flow being received so that it conforms to the target bandwidth of the rate-controlled video. Otherwise network resources will be wasted on the multicast data path between the sender and the gateway, since many of the video frames simply will be dropped by the gateway.

## 4.1 Layered Multicast

An elegant solution to multicast flow control is to subdivide the data stream into a hierarchy of cumulative layers each of which is transmitted to a unique multicast address. Thus, each individual receiver can control the bandwidth of the data stream being received by subscribing to an appropriate number of multicast groups. The quality of the reconstructed data depends on how many layers are available in the decoding process. The flow control problem is thereby reduced to finding a way for the receivers to determine the optimal number of layers to subscribe to. Unfortunately, this is not so easy to do in the general case. Several approaches have been suggested [8, 9, 10]. In the present case, however, given our assumption that the bandwidth bottlenecks are the HTTP connections rather than the multicast backbone, we can use information about the bandwidth constraints of the HTTP connections as input to the multicast flow control algorithm. Since HTTP is transported over TCP we can actually let the flow control algorithm of TCP drive the decision algorithm for subscribing to multicast layers. What we need is a way to measure the bandwidth that TCP allocates for the HTTP connections. We also need a layered representation of the video signals to be transmitted. The easiest way to achieve a layered video encoding is to distribute the individual video frames temporally over the group of layers. Thus, subscribing to an increased number of layers will result in a higher frame rate of the decoded video. The temporal layering is simplified if only intra-frame compression is used, as is the case with the JPEG encoding used in WebSmile.

## 4.2 The TCP-Driven Multicast Flow Control Algorithm

Since one WebSmile gateway can support many HTTP-connected clients with video from the same session, the client with the fastest connection determines how many multicast layers must be subscribed to, in order to support the desired frame rate for each client. That is, if a gateway is serving $n$ clients with TCP connections of bandwidth $B_i$, $i$=1..$n$, respectively, with video distributed uniformly across $L$ distinct layers with an aggregate bandwidth of $B_{tot}$, then the number of multicast layers the gateway should subscribe to, $L_{GW}$, is given by

$$L_{GW} = \left\lceil \frac{\max(B_i)_{i=1}^{n}}{B_{tot}} \cdot L \right\rceil \quad (1)$$

Note that the value we get must be rounded up since only integral layers can be received. To determine the effective bandwidth of the HTTP connections WebSmile measures the time each socket write operation consumes and calculates the mean sending time for each transmitted image. Since a blocking socket interface is used, the sending time for an individual image will sometimes be very short (in case of an empty output socket buffer) and sometimes disproportionately long, but on the average a good estimation of the actual throughput is achieved.

If the expression in (1) was to be used directly by WebSmile in the multicast flow control algorithm, the total bandwidth of the video stream ($B_{tot}$) must be known. However, this parameter may change during the session, so it would be better if an equivalent expression not including $B_{tot}$ could be derived. Furthermore, since the parameter being measured is the average socket send time for an image, it would be easier if that parameter could be used directly instead of calculating the bandwidth.

Now, if we let $t$ denote the average time to send an image on the HTTP socket connected to receiver $k$, where $B_k$=max($B_i$), then the average frame size of the video, $J$, will be given by

$$J = B_k t.$$

Observing that the average frame size can also be written as

$$J = \frac{B_{tot}}{f},$$

where $f$ is the frame rate of the video, we note that the fraction in (1) can be written as

$$\frac{\max\left(B_i\right)_{i=1}^{n}}{B_{tot}} = \frac{B_k}{B_{tot}} = \frac{J}{Jtf} = \frac{1}{tf} \quad (2)$$

Substituting (2) in (1) gives the simple formula

$$L_{GW} = \left\lceil \frac{L}{tf} \right\rceil, \quad (3)$$

where $L$ and $f$ are constants. Thus the optimal number of layers to subscribe to can be determined by measuring only the transmission time for the video frames, providing we have an a priori knowledge of the number of layers used and the frame rate of the video. (Strictly speaking, the frame rate could be experimentally learned by receiving one layer and multiplying the observed frame rate with the total number of layers, $L$.)

The algorithm is continually monitoring the average image transmission time to compute the optimal subscription level and thus dynamically adapts to bandwidth fluctuations on the HTTP connections in response to TCP's flow control.

Note that the parameter $t$ in (3) was defined to be the average transmission time for an image on the TCP socket with the fastest connection. This implies that the gateway must keep track of which TCP connection has the lowest average sending time (highest throughput) at any time and use that value as input to the flow control algorithm. However, in the actual implementation of WebSmile, each HTTP-connected user is served by a separate process. Running the flow control algorithm independently in all processes using (3), with $t$ being the average image sending time for the process' own TCP connection, will in effect lead to an allocation of multicast addresses where the set of addresses allocated by the process with the fastest TCP connection will be a superset of the sets of addresses allocated by the other processes. The total allocation of multicast addresses on the gateway is hence determined by the process with the fastest TCP connection. Thus, the desired behaviour is achieved without the processes having to synchronize their operation (or even be aware of the other processes' existence.)

Finally, note that the bandwidths $B_i$ and $B_{tot}$ used in (1) in the deduction of (3) represent the actual throughput of image data, excluding transport protocol overhead. Thus, the difference in protocol overhead between HTTP/TCP and RTP/UDP doesn't impact the flow control algorithm, although it affects the overall bandwidth consumption. The transport protocol overheads are estimated in section 5.1.

## 5. Performance

To measure how well the flow control algorithm allocates bandwidth on the multicast network in relation to the throughput on the HTTP/TCP-connection, a test environment was set up with the configuration shown in Figure 3.

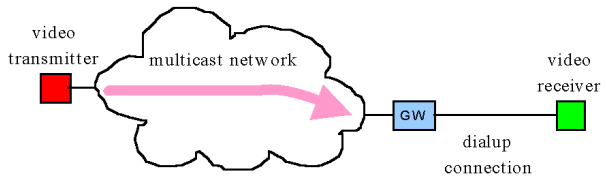


**Figure 3: Network configuration used for performance tests**

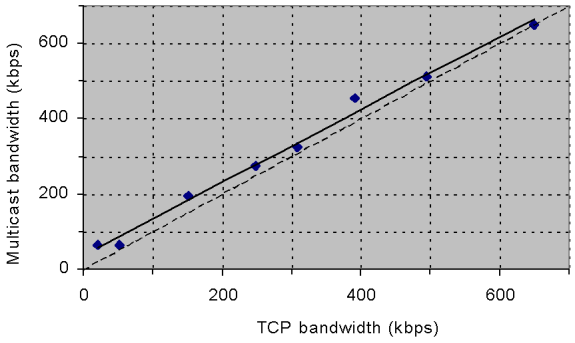


**Figure 4: Multicast bandwidth allocation in relation to HTTP/TCP bandwidth**

The line speed of the dialup connection was configurable so that different network access technologies could be emulated (in terms of bandwidth). The connection was configured at a number of different speeds ranging from 30 kbps to 2 Mbps and the resultant bandwidths allocated by WebSmile on the HTTP/TCP-connection and on the multicast connection were measured. The video was transmitted at 25 frames per second in 10 distinct temporal layers. The image resolution was 192 by 144 pixels, which after JPEG compression resulted in a total bitrate of about 650 kbps, or about 65 kbps per layer. In Figure 4 the multicast bandwidth is plotted against the HTTP/TCP bandwidth.

It is clear that the bandwidth allocated on the multicast network depends linearly on the bandwidth available to the TCP connection, as expected. It can also be noted that on the average a slightly higher bandwidth is allocated on the multicast network compared to the TCP bandwidth. (The dotted line in Figure 4 delineates an identical allocation of bandwidth.) This is due to the fact that bandwidth is allocated at a much coarser scale on the multicast network, the granularity being the bandwidth of one layer compared to TCP's byte-level congestion window adjustments. On average the over-allocation of bandwidth on the multicast network is one half of the layer bitrate, which, in the present case, is about 30 kbps.

### 5.1 Transport Protocol Overhead

The bandwidth measurements presented in Figure 4 include the overhead imposed by the transport protocols. To investigate what influence the difference in protocol overhead between HTTP/TCP and RTP/UDP transport has on the bandwidth allocation we roughly estimate the overheads.

For the HTTP/TCP transport the overhead for each packet is 20 bytes for the IP-header and 20 bytes for the TCP header. Furthermore, each image is encapsulated by an application-specific MIME multipart boundary identifier. Also a *content-type* and *content-length* MIME-field is added for each image. The WebSmile implementation adds 65 bytes of MIME-information for each image. The total overhead depends on the data segment size chosen by the TCP implementation, the fragmentation occurring on the end-to-end network connection and the average size of the images transmitted. Assuming a packet size of 576 octets including IP and TCP header (the default packet size in TCP), no additional fragmentation and an average image size of

3.5 Kb, a total overhead of 8.76% is obtained. Note that this estimation requires that the TCP sender always has enough data in the output socket buffer to transmit a full-sized packet.

The RTP/UDP overhead consists of the 20 byte IP-header, 8 bytes for the UDP header, 12 bytes for the RTP header and 8 bytes for the JPEG/RTP profile header, giving a total of 48 bytes per packet. The same packet size and fragmentation situation as in the TCP case gives an overhead of 8.33%. However, on average, the last datagram of an image will be only half of the maximum datagram size. With a 3.5 Kb average image size this increases the overhead to 9.31%.

Note that in the estimations above the overhead of retransmissions in the TCP protocol isn't included and neither is the overhead due to periodic RTCP packet transmissions in the RTP case. Nevertheless, this rough estimation indicates that the overhead is approximately the same for both transports and accounts for about 9 percent of the total bandwidth, both for HTTP/TCP and the RTP/UDP.

## 6. Future Work

The applet used for displaying live video in the client WWW-browser will be extended with functionality to playback audio as well, so that the system can be used as both an audio and video gateway. Furthermore, the integration of media transcoding support into the WebSmile system will be studied in more detail.

## 7. Summary

In this paper the development of a novel Internet video gateway has been presented. The system, known as WebSmile, enables Internet users that normally would be unable to participate in multicast video conferences to partake using only a standard web browser. The need for a system like this is motivated by the fact that many Internet users will continually be unable to utilise many of the advanced technologies needed for multicast conferencing due to resource unavailability, security concerns and other shortcomings. The design and implementation of WebSmile as an application level gateway co-located with a WWW server was discussed in section 3.

In section 4 a novel TCP-driven flow control algorithm for layered multicast video was introduced. The algorithm implemented in the video gateway works by adapting the rate of the multicast video flows to the bandwidth allocated by the HTTP/TCP connections to the receiving clients. A layered video encoding transmitted to a set of multicast addresses was suggested to enable the receiver-oriented multicast flow control. The performance of the flow control algorithm was measured in a test network configuration, and the results show that the multicast bandwidth allocated by the gateway closely match the TCP connection bandwidth. The transport protocol overheads for JPEG-video over HTTP and RTP, respectively, were estimated and found to be approximately the same.

## References


1. H. Eriksson, "Mbone: The multicast backbone," Communications of the ACM 37, 1994.
2. E. Amir, S. McCanne, H. Zhang, "An application level video gateway," ACM Multimedia '95, November 1995.
3. P. Parnes, K. Synnes, D. Schefström, "Lightweight application level multicast tunneling using mTunnel," Journal of Computer Communication, 1998.
4. http://www.icq.com - "The ICQ Internet chat service".
5. J. Oikarinen, D. Reed, "Internet Relay Chat (IRC) protocol," RFC1459, May 1993.
6. B. Fenner. "The multicast router daemon - mrouted," ftp://ftp.parc.xerox.com/pub/net-research/ipmulti.
7. S. E. Deering, "Multicast routing in a datagram internetwork," PhD thesis, Stanford University, December 1991.
8. S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven layered multicast," Proceedings of ACM SIGCOMM '96, October 1996.
9. L. Vicisano, L. Rizzo, J. Crowcroft, "TCP-like congestion control for layered multicast data transfer," Proceedings of IEEE Infocom '98, San Francisco, CA, March 1998.
10. L. Wu, R. Sharma, and B. Smith. "ThinStreams: An architecture for multicasting layered video," Proceedings of NOSSDAV '97, May 1997.
11. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A transport protocol for real-time applications," RFC1889, January 1996.
12. L. Berc, W. Fenner, R. Frederick, S. McCanne, P. Stewart, "RTP payload format for JPEG-compressed video," RFC2435, October 1998.
13. J. Postel, "User Datagram Protocol (UDP)," RFC 768, August 1980.
14. M. Handley, "SAP: Session Announcement Protocol," Internet draft, IETF Multiparty Multimedia Session Control Working Group, 1997.
15. M. Handley, V. Jacobsen, "SDP: Session Description Protocol," RFC2327, April 1998.
16. T. Turletti, J. Bolot, "Issues with multicast video distribution in heterogeneous packet networks," Proceedings of Packet Video Workshop, Portland Oregon, September 1994.
17. J. Pasquale, G. Polyzos, E. Anderson, V. Kompella, "Filter propagation in dissemination trees: Trading off bandwidth and processing in continuous media networks," Proceedings of NOSSDAV '98, pp. 269-278, October 1993.
18. D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, G. Minden, "A survey of active network research," IEEE Communications Magazine, pp. 80-86, January 1997.
19. W. Ooi, R. van Renesse, B. Smith, "Design and implementation of programmable media gateways," Proceedings of NOSSDAV 2000, June 2000.
20. N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part one: Mechanisms for specifying and describing the format of Internet message bodies," RFC1521, September 1993.
21. K. Coar, D. Robinson, "The WWW Common Gateway Interface version 1.1," Internet draft, June 1999.