An Active Router Architecture for Multicast Video Distribution

Ralph Keller, Sumi Choi, Marcel Dasen, Dan Decasper, George Fankhauser, Bernhard Plattner

[keller|dasen|gfa|plattner]@tik.ee.ethz.ch [syc1|dan]@arl.wustl.edu
Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
Applied Research Laboratory, Washington University, St. Louis MO, USA

***Abstract* -- Video distribution over the Internet poses many challenges. Due to the best-effort nature of today's public data networks, end system applications cannot rely on either bandwidth or delay guarantees. We designed and implemented a prototype of a multicast video distribution architecture involving knowledgeable active routers, a scalable video codec based on wavelet transformation, and a high-performance video scaling algorithm implemented as a router plugin. The plugin scales the video with an average overhead of only 22 µs per video datagram and is installed on-the-fly on the routers after the sender starts transmitting video for the first time. Through experiments on our test network, we show that we can dramatically improve the video quality on the receivers (up to 15 dB PSNR) by scaling the video on the routers to almost any target bandwidth. The target bandwidth is evaluated by the router solely based on monitoring of the load situation of the router's downstream links and can be adjusted within 50 ms.**

I. Introduction

Delivering real-time video over the Internet is very desirable because numerous applications like video conferencing, video-on-demand, and video broadcast require it. Generally speaking, we can distinguish between four types of video distribution schemes for the Internet: adaptive versus non-adaptive, and point-to-point (unicast) versus point-to-multipoint (multicast) schemes.

Non-adaptive schemes rely on a fixed bandwidth provided by the network usually through a reservation made ahead of time. These schemes typically use relatively simple video coders and decoders (codecs), but depend on a service from the network (the ability to reserve bandwidth and guaranteed delay bounds), which is not available in most parts of the Internet today, and may not be available in the foreseeable future.

Adaptive schemes employ codecs, which try to compensate for packet loss in the network by extensive buffering and pre-evaluation of the network service quality on the receiver, and employ *receiver feedback* to adapt the video output rate to the network load. While buffering of multiple seconds of video works well for video-on-demand (VoD) applications, it is not suited for interactive applications like teleconferencing. The main problem with receiver feedback is latency: by the time the sender receives the information about a desired rate, it might already be outdated due to the fact that the load of the network has changed.

In the point-to-point case, video streams are sent from a sender to only one receiver. In the point-to-multipoint case, packets are passed to a group of receivers which can potentially be very large (e.g., for a live broadcast of an important event). In this case, especially with a set of receivers on links with heterogeneous bandwidth, sending feedback to the sender does not work well for two additional reasons compared to the point-to-point case: First, the sender must adapt to the slowest receiver which penalizes faster receivers, and second, severe scalability problems (feedback implosion) occur if the receiver group is large. Therefore, the most popular approach for point-to-multipoint video distribution over the Internet is layered multicast [15], where the sender distributes the video over multiple layers instead of one, as in the unicast case. The individual layers are sent out on different multicast addresses and a receiver can subscribe or unsubscribe to a layer depending on whether it wants to increase or reduce the total bandwidth it receives and therefore increase or reduce the quality of the video. The receiver's feedback goes to an upstream router instead of the sender. The decision about joining or leaving multicast groups is made on the receiver by observing the link over time and performing so called "join experiments", which can take several seconds. After determining whether a layer should be subscribed or unsubscribed, an IGMP message is sent to the router, which again can take several hundred milliseconds. Thus, this scheme is suboptimal under frequent changes in link load. Unfortunately, typical Internet traffic tends to produce such patterns [21].

Ideally, the decision about whether or not to forward a video datagram for both the point-to-point as well as the point-to-multipoint case should be made by the router itself, based on the observation of its link load, without requiring any feedback from the receiver. Adaptation of multimedia traffic is called *media scaling*, and the router implementing video scaling is called *media gateway.* Media scaling does not exclude sophisticated receiver-based adaptation and loss compensations schemes – in fact, it can complement them in an incredibly powerful fashion.

We identify three important key requirements for a feasible implementation of a video gateway:

- **Router architecture:** The number of proposed video encoding and decoding schemes is already large and it is unlikely that this will change in the future. Further, changes and modifications to these schemes are frequent. For these reasons it is crucial that a scaling algorithm can be deployed automatically and on-demand to routers implementing video gateway functionality. The router must be on-the-fly extensible to perform the scaling without service interruption. Packet processing in an environment where routers are programmable is called *active networking* [26].
- **Video Codec:** The video codec must allow fine-grained scaling of the video to optimally adapt to a large range of

target bandwidths and traffic patterns. Most video scaling gateways proposed and analyzed in the literature use MPEG [18] video codecs. However, the problem with MPEG is that due to its architecture it allows for only a very limited number of adaptation levels.

- **Scaling algorithm:** The scaling algorithm must allow for a highly efficient implementation. While it would be tempting to convert video on the fly by changing from one encoding scheme to another on the gateway (*transcoding),* a simple calculation[1] demonstrates that it is not feasible to implement transcoding with today's line card processors and hardware architectures for any reasonably high data rate. At this point, all a video gateway can do is selectively drop or forward packets, and even this decision must be implemented in a very efficient way for links of megabit speed and faster.

In this paper, we describe a running prototype of an innovative video scaling architecture. It improves video quality of real-time unicast and multicast video as seen by the receiver by up to 15 dB PSNR compared to best-effort forwarding; it addresses all three key requirements as follows: First, we run our innovative *Active Router Plugins* [9] software architecture on the router. Active Router Plugins allows for on-demand downloading and installation of a kernel plugin on the router on occurrence of a plugin identifier in a video datagram. Second, we use a wavelet-based, adaptive video codec called *WaveVideo* [11], which supports up to 50 scaling levels compared to only three levels offered by MPEG. Third, we designed and implemented a video scaling algorithm which is capable of fine-tuning the video to the desired rate provided by the router based on momentary link load patterns by performing a simple table lookup, and which executes in 22 µs on our prototype PC-based hardware.

In Section II, we briefly describe the Active Router Plugins software architecture and its most important properties. In Section III, we describe how we designed and implemented our video coding and scaling algorithm on the Active Router Plugins software framework. In Section IV, we evaluate our implementation. In Section V, we present related work. In Section VI, we conclude this paper by summarizing the most important results, and briefly mention future work.

II. The Active Router Plugins architecture

As we pointed out in the previous section, we believe that it is very important to *automatically* deploy any type of application-specific processing to routers (active networking). In this section, we will briefly describe our Active Router Plugins software architecture and its application to WaveVideo scaling. The architecture was introduced in [8] and [9].

Router plugins are code modules, which implement specific datagram processing functionality like encryption, congestion control, reliable multicast or, as we show in this paper, WaveVideo scaling. We extend the regular IP forwarding loop on our router to look for special headers between the IP header and the UDP or TCP header, which reference plugins[2]. If such a header is found, the packet is passed to the referenced plugin prior to being forwarded. If the referenced plugin is not present on the system, it is downloaded from a so called *code server* over the network and automatically installed in the kernel. We call this on-demand downloading of plugins *Distributed Code Caching for Active Networks* (DAN) [9].

A WaveVideo datagram as seen by our router is depicted in Figure 1. After the IP header, we insert an Active Networking

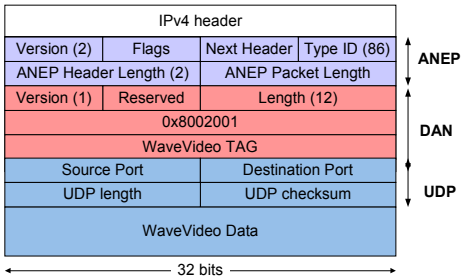

Figure 1: WaveVideo datagrams

Encapsulation Protocol Version 2 [1] header, which is the active networking community's standard header to precede active networking payload. The ANEP header can be inserted either by the end system or by a specifically configured gateway near the sender. Similarly, it is removed either by another gateway near the receiver, by the receiver's kernel, or by the video application running on the receiver. For our experiments described in Section IV, we insert the ANEP header in the sender's kernel using a special plugin and remove it in the receivers kernel, which makes active processing completely *transparent* to the end system application.

ANEP's *type ID* field indicates that a DAN header follows the ANEP header. The DAN header is specific to our Active Router Plugins environment. ANEP's *next header* field points to UDP. The DAN header essentially identifies the plugin to perform the video scaling. It carries a WaveVideo-specific tag, which is used by the scaling algorithm to decide whether or not to forward the packet based on the current load situation of the outgoing link (as explained in more detail in Section III). After the ANEP/DAN part follows the regular UDP header and the video data as UDP payload.

Before we explain how such a packet is processed on our kernel, we introduce the most important components of the Active Router Plugins kernel implemented in the NetBSD Unix system (Figure 2):

- **Device Drivers/Layer 2 processing (DD):** the DDs are standard device drivers implementing network hardware-specific send and receive functions. They pass packets to the Packet Classifier for flow classification before passing them to the IP stack.
- **Packet Classifier (PC):** The PC caches flow-specific

[1] At 155 Mbits/s OC-3 speed a router can spend only 26 µs to receive, process, and forward a packet (assuming an average size of 500 bytes) – clearly not long enough for any kind of sophisticated transcoding, except when implemented in custom hardware, which would mean loss of flexibility.

[2] Note that this scheme is compatible with non-active routers. They simply forward an active packet like any IP datagram.

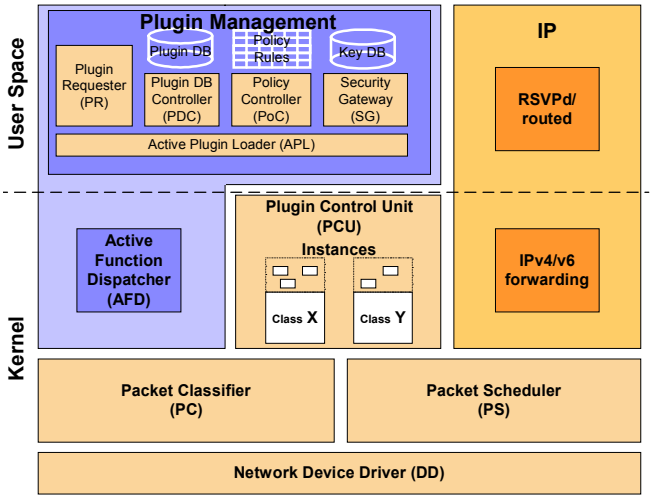

Figure 2: Active Router Plugins Systems

information (e.g., the route) and makes this information available to plugins.

- **Packet Scheduler (PS):** Our PS is an implementation of the popular Deficit Round Robin (DRR) [25] scheduler, which provides fair services to an arbitrarily large number of best-effort traffic flows.
- **Plugin Control Unit (PCU):** the PCU provides a communication mechanism for plugins to receive messages from user space as well as from other plugins.
- **Active Function Dispatcher (AFD):** the AFD scans a packet for plugin identifiers and passes it to the corresponding plugin. In case of a previously unseen plugin identifier (e.g., when the router receives our WaveVideo packet for the first time), the AFD sends a request for the new plugin to the Active Plugin Loader.
- **Active Plugin Loader (APL):** the APL is the central module of the Plugin Management (PMGNT) and acts as an interface between the kernel and the rest of the PMGNT. It forwards requests from the kernel to other plugin management modules and vice versa.
- **Policy Controller (PoC):** the PoC deals with policy rules set by the router's administrator. The APL calls the PoC for checking the validity of a request for a plugin.
- **Security Gateway (SG):** the SG maintains RSA public keys of well known code servers. It provides a method for authenticating a plugin by comparing the signature received from a code server and its MD5 hash signed by the public key of the code server.
- **Plugin Database Controller (PDC):** the PDC manages a database of plugins. It maintains local copies of plugins downloaded from code servers, so that subsequent requests from the kernel for plugins (for example after a restart of the router) can be answered using local copies instead of sending requests to code servers.
- **Plugin Requester (PR):** the PR is responsible for requesting plugins from code servers, and to process replies to such requests.

Given the short descriptions of each module, the data path for processing of a WaveVideo packet looks as follows:

As the WaveVideo packet enters the device driver, it is sent through the packet classifier, IP input processing, and ANEP header processing to the AFD. The AFD looks up the plugin identifier and passes the packet to our WaveVideo plugin. The WaveVideo plugin either forwards the packet by passing it on to the packet scheduler, or drops it as described in Section III. In case the AFD sees an unknown plugin identifier, it sends a request for the WaveVideo plugin to the APL and forwards the packet in this case without scaling the video stream. When the APL receives the plugin request, it makes a call to the PR to download the plugin from a code server. As the plugin arrives at the PR, it is passed to the APL. The APL checks with the PC for policies and with the SG for authentication. If the plugin is accepted, the PDC creates a local copy, while the APL downloads it to the PCU. As the plugin registers itself with the PCU, the AFD passes subsequent WaveVideo packets to the plugin for scaling.

Currently, the Active Router Plugins software architecture is implemented on a Pentium PC with three ATM network interfaces and an Ethernet card acting as a router, which we call Active Network Node (ANN). We are currently working on a more sophisticated version of the ANN as described in [8], which consists of an ATM switch fabric with 8 ports and active line cards on every port, which we call Active Network Processing Elements (ANPEs). The ANPEs consists of a custom 1.2 Gbit/s ATM host adapter chip, a Pentium CPU, a large FPGA that can implement active processing in hardware, and up to two gigabytes of DRAM. The Active Router Plugins software architecture running on the ANPEs is identical to that running on the prototype PC, which makes the results presented in this paper comparable to what we expect to see on the next generation ANN.

In the next section, we first take a closer look at different video encoding schemes before we present our innovative, adaptive WaveVideo codec. We will show how a video can be scaled on a knowledgeable gateway router and how we implemented scaling on our Active Network Node.

III. Multicast Video Distribution using a Network of ANNs

A. Video Encoding Schemes

The method used to encode individual pixels of the video frames into network packets is central to video scaling. In this section, we elaborate on various encoding schemes and illustrate, how scaling can be performed efficiently.

In general, scaling of video can be performed in *spatial* and *temporal* space, the later being 2D, including color. Spatial scaling means decreasing the resolution while temporal scaling means modifying the frame rate. Color scaling allows for reduction of color which can go as far as turning a colored video into a grayscale or black and white video. Only a few codecs allow *efficient* scaling. For example, coding in YC_bC_r color space makes adaptation in the color space very simple: a grey scale video can be generated by only decoding the luminance (Y) channel. However, adaptation of compressed video

is problematic, since compression reduces redundancy and a lot of computation is required to access the pixel information. This is especially true for schemes using motion vectors to compensate for motion in the temporal space. The problem is resolved by compression in different layers whereas each layer possesses well defined properties. Receivers select specific layers to receive the desired video quality.

The most popular video standards for the Internet are proposed by the Motion Picture Expert Group called MPEG, and the technically similar ITU-T derivative for video telephony H.263 [20]. MPEG consists of a family of codecs for various applications. MPEG-1 has been designed for videodisk playback and does not offer adaptation. MPEG-2 offers scalability with respect to Signal-to-Noise Ratio (SNR), spatial and temporal space. SNR scaling refers to the fact that visual quality can be adapted other than by resolution reduction. Small changes in the picture are smoothed out, which is not immediately visible to the human eye, but greatly improves the compression rate. MPEG-4 [19] is a new standard targeted at video conferencing. It has to be seen as a meta standard defining a framework for various encoding schemes. As a baseline it includes H.263 like encoding, which is a standard for low bitrate communication using multiples of 64 Kbit/s channels. Technically, H.263 belongs to the same family of codecs as MPEG. Both use discrete cosine transformation (DCT) and similar motion prediction and compensation.

However, formats like MPEG and H.263 make scaling a computation-intensive task since specific quality features (such as the DCT and motion coefficients) cannot be easily extracted from a video stream. Partial decompression and buffering of frames is required to perform video scaling. The standard stream formats were neither designed for scaling nor loss tolerance (see [10] for a discussion of a loss-tolerant H.263 version), thus lack important features. Since high-performance scaling is a crucial requirement, we favor a wavelet-based approach called WaveVideo featuring a simple difference-based temporal encoding.

B. WaveVideo Encoding

WaveVideo [7], [11] encodes single video frames by first transforming the color channels from the spatial to the frequency domain and then quantizing and compressing the decorrelated output. A two-dimensional wavelet transform (WT) is applied to the image, which is implemented and approximated using iterated discrete-time filters. This 2D-process splits an image into a low-frequency (LL) and three high-frequency subbands (HL, LH, HH) and is repeated recursively on the LL-subband at each level of the transformation. For luminance (Y) and color difference (C_bC_r) channels a tree consisting of low- and high-frequency subbands is generated (Figure 3).

Once an image has been transformed into frequency space, the correlation contained in typical natural pictures is dissolved and the actual compression step introducing loss by quantization of the high frequency coefficients is performed. Usually, transformed images have a large number of zero-coefficients and only decorrelated features are represented by nonzero values. Statistical analysis of coefficient distributions manifests that not only zero values, but small values in general dominate the transformed image.

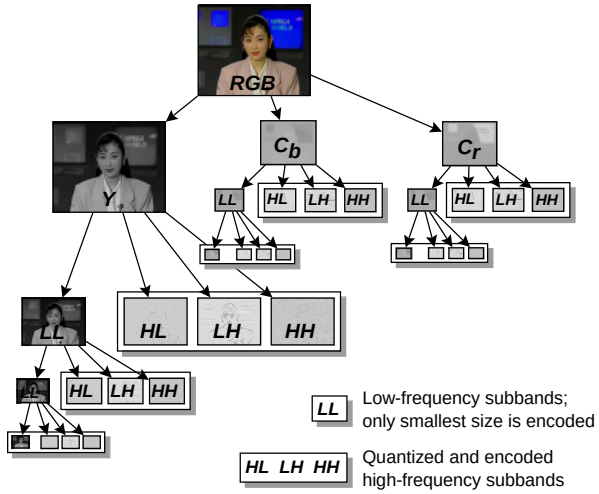

Figure 3: Hierarchical decomposition of video images

The quantized and entropy-coded (i.e., run-length encoded) leaves of the tree shown in Figure 3 are finally assembled into datagrams with a tag describing the packet's video content, an ANEP/DAN header, and the standard UDP header (see network format in Figure 1).

The applied compression and channel coding schemes make WaveVideo suitable for environments with high packet losses such as heavily congested Internet links and wireless communication [16]. Independently of the use of video gateways in the network, the WaveVideo decoder running on the receiver tries to conceal packet losses in two ways:

1. The loss of high-frequency wavelet-coded coefficients results in a smooth degradation of the whole image area. No artifacts like missing blocks or wrongly colored blocks (which can be noticed using MPEG under heavy loss) are visible.
2. If packets containing low-frequency parts are lost, a caching algorithm (LL-cache) supplies a previous version of the missing low-frequency coefficient.

More and more products are using wavelet-based codecs implementing proprietary coding formats, especially for scalable streaming over the Internet. Further, several upcoming open standards will be based on this method as well (e.g., JPEG 2000 [14], MPEG-4 [19]).

C. Video Stream Scaling

The active network node can apply two different schemes to scale the video stream to a lower transmission rate: It can either reduce the frame rate or the image quality of individual frames. Dropping complete image frames is simple to implement since the node has to examine only the sequence number of a frame. However, frame rate filtering has to take into account that filters can be cascaded and that packets have been eliminated by upstream routers. Thus, to create an equidistant spacing of video frames, the node has to buffer outgoing frames and for-

ward them at a constant rate.

The second approach reduces the image quality by eliminating high-frequency parts from the video stream. This scheme drops packets from high-frequency subbands at deep transformation levels first, since these coefficients result in only a slight degradation of the image, perceivable by a minor loss of details. Contrary, low-frequency parts are forwarded with the highest priority to ensure the overall consistency of the image.

Eliminating high-frequency parts from the video allows scaling the stream by approximately a factor of 50 (depending on the frame size of the video): for example, a small-sized Quarter Common Intermediate Format (QCIF) video frame has four wavelet transformation levels for the luminance channel (Y) and three transformation levels for the chrominance channels (C_bC_r). Each transformation level contains three high-frequency parts (HL, LH, HH). Having four transformation levels, there are potentially 12 high-frequency parts to eliminate from the Y-channel and 9 high-frequency parts from the C_b- and C_r-channels. All in all, this allows to adapt the video stream from full-quality frames consisting of 33 segments to just three low-frequency segments per frame. Using this scheme, an average sample video stream can be scaled from 2.6 Mbit/s down to a minimum of about 50 Kbit/s.

D. Implementing Video Scaling

Besides providing a high level of scalability, WaveVideo stream adaptation can be implemented efficiently by labeling each individual network packet with a tag. The tag describes the frame's content such as the corresponding frequency subband, color channel, and depth in the wavelet tree (Figure 4).

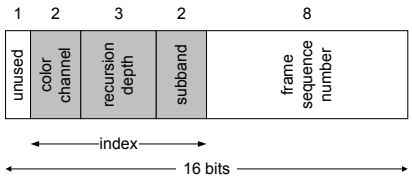

Figure 4: Bit layout of the network tag

The tag allows for reduction of the stream's bandwidth by selectively dropping packets belonging to a particular quality set. In our case, we define a quality set such that the transmitted output stream does not exceed an upper bandwidth limit. In other words, we want to prevent the packet scheduler from randomly dropping packets to ensure that datagrams containing low-frequency coefficients (which define the overall image) reach the video sink even under congestion. Our scheme continuously monitors the forwarded video stream and periodically adapts the quality profile to fit the bandwidth assigned by the packet scheduler. Thus, if the forwarded bandwidth is higher than the bandwidth that the scheduler can guarantee, we modify the quality profile such that more high-frequency parts are eliminated from the video stream. Similarly, we include more high-frequency coefficients into the profile if the filtered video stream falls below the limit set by the scheduler. With a profile update period of 50 ms, the WaveVideo plugin can quickly respond to both fluctuations in the available link bandwidth and variations in the video stream encoding.

This scheme can be implemented in a very simple and efficient way using a quality profile encoded as a boolean lookup table with 128 ($=2^7$) entries to hold all combinations of color channel, recursion depth, and subband (see Figure 4, whereas the sequence number is not used).

Thus, if a packet is transiting through the node, we use 7 bits from the tag as an index into the quality profile table and lookup the entry: if the value is set, the node forwards the packet, otherwise it drops it. Modifying the quality profile is simple: to add or remove high-frequency coefficients from the forwarded video stream, we simply set or reset the corresponding table entries, beginning with coefficients at deep recursion levels. In summary, the overhead imposed by scaling the video stream is just a table lookup. As we will show in the next section, both operations can be executed very fast.

IV. Evaluation

A. Test Network Setup

We evaluate our architecture and implementation using a test network of end systems and ANNs illustrated in Figure 5. We

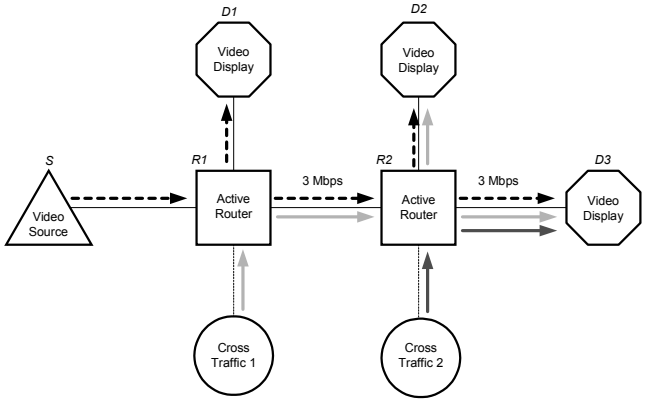

Figure 5: Test Network Setup

use two routers running our Active Router Plugins kernel. The video source (S) multicasts the video to three destinations (D1, D2, and D3). Receiver D1 is not affected by cross traffic at all, therefore displaying a disturbance free video in all test cases. D2 and D3 are exposed to cross traffic: the link between R2 and D2 is shared by the video and one cross traffic stream, whereas the link to D3 is shared by the video and two cross traffic streams. Thus, D2 is moderately congested and D3 heavily congested. For all of our experiments we use two test videos: *Akiyo*, a low-motion sequence, which requires 1.3 Mbit/s in average, and *Foreman*, with a higher degree of motion, requiring 2.6 Mbit/s. Both videos are lossless encoded with QCIF resolution at 12 frames/s. To bring the routers' downstream links into a congested state by the cross traffic, we restrict the respective ATM link rate to 3 Mbit/s using hardware pacing by the ATM card[3]. The routers run an implementation of the Deficit Round Robin (DRR) [25] packet scheduler, which assigns an equal share of the available bandwidth to each flow. Thus, when the link is shared with one

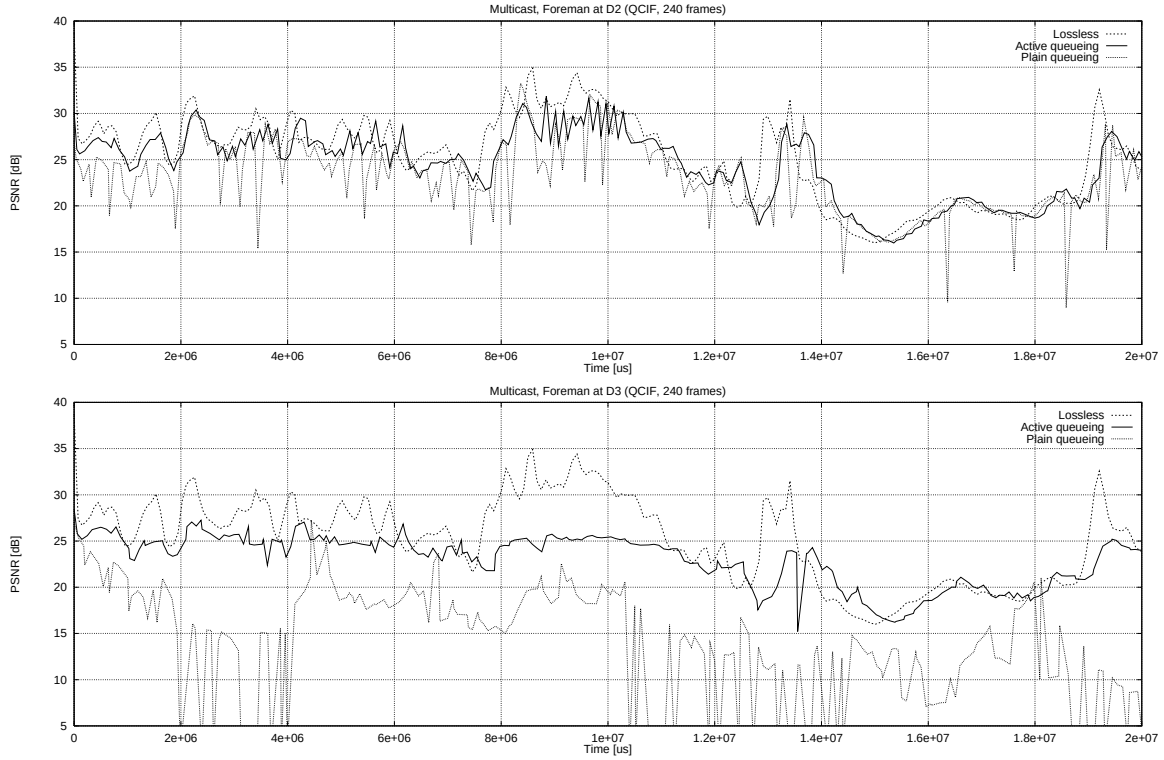

Figure 6: Video quality measurements using the Foreman test sequence on the receivers D2 (moderately congested) and D3 (heavily congested)

competing cross traffic flow, the video flow will get 1.5 Mbit/s. With two concurrently active cross traffic flows, the video gets 1 Mbit/s. If the bandwidth provided by the packet scheduler to the video flow is insufficient and no scaling is active, the packet scheduler drops the oldest packets (plain-queueing) from the video flow queue.

B. Scaling Performance

We measured the scaling performance of our WaveVideo plugin on R2, which is a Pentium II PC running at 300 MHz. We use the Pentium's processor clock register (TSC) which is incremented by every cycle and allows for very accurate measurements.

The scaling algorithm requires approximately 6700 cycles or 22 µs per video datagram. Thus, we can scale 45,000 packets per second. Given the average WaveVideo packet size of 820 Bytes as observed with our test sequences, we could scale at a rate of 360 Mbits/s[4].

C. Plugin Download Time

In [9] we showed that a 100 KBytes plugin requires approximately 500 ms to download from a code server connected one hop away from the active router including security checks and installation on the router. 100 KBytes corresponds roughly to the size of our scaling plugin. To download the plugin from a transatlantic site 22 hops away representing a worst case scenario, the router takes approximately 4 seconds for the same procedure. Since we can download the plugins almost parallel to both routers, the complete download sequence does not take longer than 4 seconds and is much faster if the code server is closer. During the download time, the video is viewable on the receivers, although in bad quality for those receivers on heavily congested links. Note that the download of the plugins is only required the very first time routers process a WaveVideo packet.

D. Video Quality Measurements

We compared the received test sequences with the original videos used for encoding and calculated the mean square error (in PSNR) for the luminance channel (the color channels follow the same behavior). The results are shown in Figure 6. Knowledgeable packet dropping in congested situations clearly demonstrates its benefits: On D2, which is only moderately congested, the video quality almost always follows the PSNR of the lossless video. Also, visual results are very good. However, plain-queueing exhibits quality losses of more than 10 dB which are perceivable by either disturbing artifacts or a general fuzziness.

On D3, which is heavily congested by two cross traffic

3. We measure at such a low rate only because the WaveVideo coder and decoder on the *end systems* are not capable of handling video streams with a peak bandwidth higher than 3 Mbits/s when running on a standard PC without custom hardware support. Note that this does not affect the relative improvement in PSNR as demonstrated in this section and that we can scale the video at much higher rates on the routers as shown in Section IV.B

4. These numbers do not take into account the time the packet spends in the rest of the kernel forwarding loop nor the time it spends on the network interface card and to cross the PCI bus.

Figure 7: Representative sample frames from the test sequences Akiyo (top row) and Foreman (bottom row) under three different conditions. Both frames are shown in the losslessly decoded version, the received video without scaling support (i.e., plain-queued packets), and the received video with scaling support (from left to right).

streams, the situation is even worse. Congestion is so high that the Foreman video sequence with a high degree of motion is no longer usable when applying plain queuing on the router. Also, the LL-cache is no longer helpful due to high losses of low frequency parts of the video. During several seconds, the video is almost unrecognizable. In contrast, using active queueing, the receiver shows almost undisturbed video playback. Subjective, visual results show some minor artifacts (little spikes due to motion blur, or small jumps on camera moves), but otherwise an intact video.

Figure 7 shows two representative video frames that were taken from the test setup[5].

Akiyo is a low-motion test sequence. It is used here to demonstrate the static image degradation incurred by packet losses on the network. As we can see, the plain-dropped video of Akiyo shows a general fuzziness, smoothed and blurred edges. Looking at the actively scaled version of the video, transmitted at the same bandwidth, a minor degradation in high-frequency details is observable but the general definition is close to the losslessly decoded video.

Looking at the Foreman test sequence, the same explanation applies. However, due to the higher degree of motion, the video without scaling support suffers not only from loss of definition, it also shows motion-blur artifacts. The same video, transmitted under the same conditions with scaling support, shows still a little motion artifact (right side of the foreman's ear) but is otherwise almost perfectly decoded.

E. Fast Reaction to Congestion

One major advantage of our video scaling architecture is the fact that nodes have local knowledge about the load situation. Because the WaveVideo plugin can directly interact with the packet scheduler, it can immediately react to congestion. To demonstrate the node's ability to quickly respond to an overload situation, we inject cross traffic bursts and measure the video quality perceived at the receiver. When cross traffic is active, the DRR scheduler assigns the cross traffic an equal share of the link bandwidth, thus limiting the available bandwidth for the video flow to 1.5 Mbit/s. Figure 8 depicts the quality of the received video streams (top), the loss in video quality due to less available bandwidth (middle), and the burst periods (bottom).

The quality of the plain-queued video stream suffers PSNR drops of 5-10 dB whenever the cross traffic is active, disturbing the video seriously. Further, the video quality does not recover until the burst is finished, making the video defective for the complete duration of the burst. On the other hand, active video scaling follows closely the video quality of the original video with only minor falls right after the cross traffic is turned on. As soon as the WaveVideo plugin discovers a decline in bandwidth (which is in the worst-case the 50 ms profile update period), it scales the video to the new available

[5] The videos analyzed in this section can be downloaded from http://www.tik.ee.ethz.ch/~keller/wvinfocom/

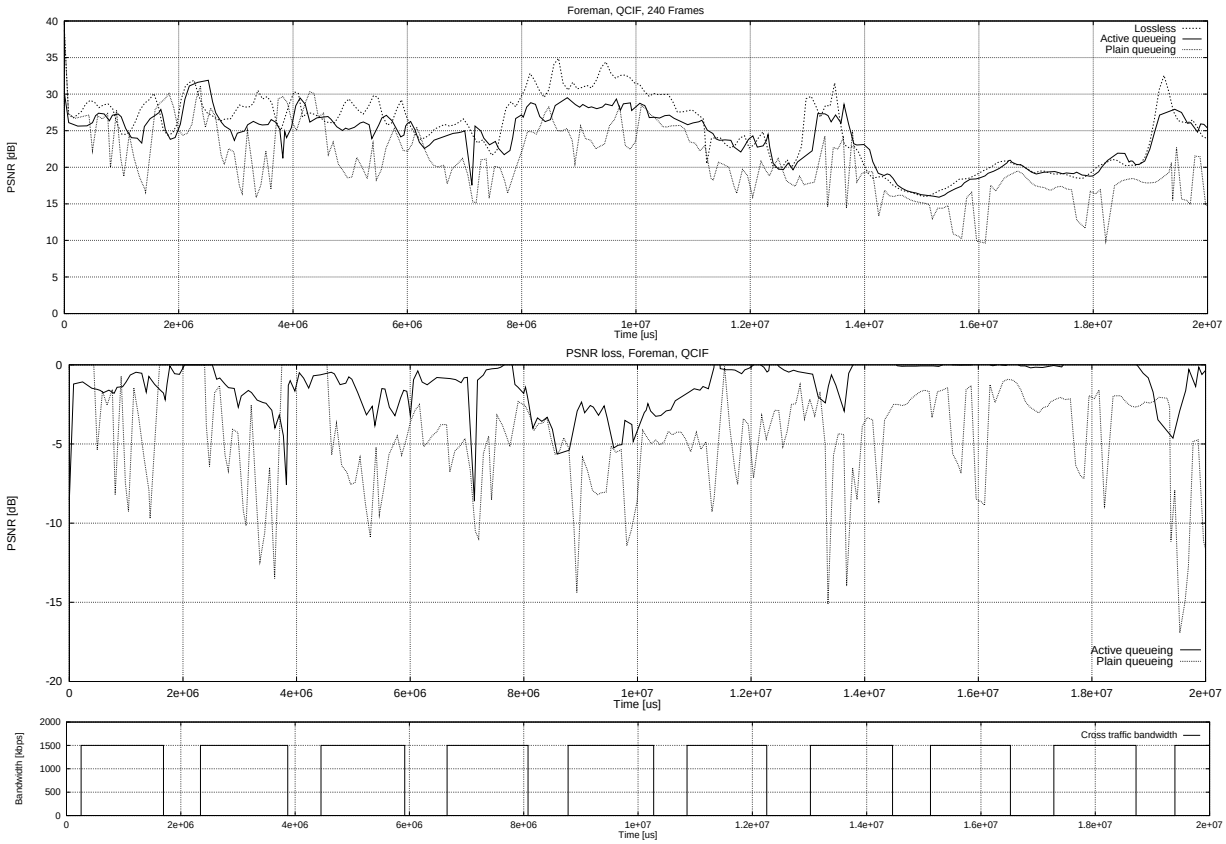

Figure 8: Impact on video quality due to periodic 1 second bursts measured in PSNR (top), PSNR difference to the original video (middle), and cross traffic activity (bottom).

bandwidth. Doing so, the video stream quality recovers rapidly to a level very close to the original video stream showing no disturbing artifacts during the bursts.

To further demonstrate that quality is indeed gained by active dropping, we analyze the received frequency subbands at the receivers. Figure 9 depicts a histogram of the subband distribution on D2, whereas routers are exposed to cross traffic bursts. Our test sequence consists of 33 subbands. The lowest-frequency subband (containing the most crucial image information) is shown at the bottom of the graph and the highest frequency subband is displayed on top. The gray level of each cell indicates how many times a specific subband was received during a period of 8 frames: if a cell is white, the subband was never received at all, and if a cell is completely black, the subband was present in all the last 8 frames. Active dropping now clearly shows its benefits: during burst activity, plain-dropping shows a random distribution of the frequency subbands, forwarding all subbands with a more or less equal probability and thus not taking into account the frequency subband. On the other hand, active dropping ensures that low-frequency subbands (which are crucial for the general image definition) are always forwarded to the receivers.

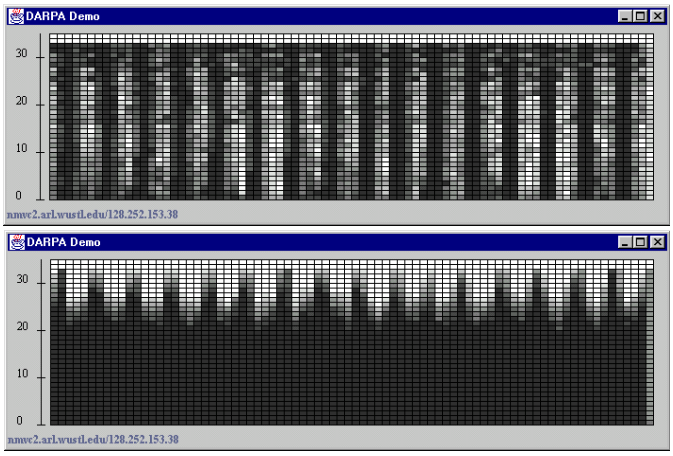

Figure 9: Distribution of received frequency subbands at D2 for plain-dropping (top) and active dropping (bottom) when exposed to bursty cross traffic.

V. Related Work

A. Programmable Router Software Platforms

Most of the better known active network research efforts like ANTS [27], SwitchWare [2], and Scout [17] would conceptually allow the implementation of a video scaling gateway. Two potential problems arise: First, the architecture must provide a way to automatically install the WaveVideo-specific code module which implements the video scaling. Second, the architec-

ture must provide a reasonably high-performance data path. ANTS allows automatic installation of code modules (it loads them from the upstream node), but its Java-based implementation running in user space of an off-the-shelf Unix operating system makes it questionable whether its provides sufficient performance. Joust [12] tightly couples a Java virtual machine to the Scout OS and delivers up to three times better performance than ANTS, but the forwarding loop is still an order of magnitude slower than a pure kernel-based implementation.

Both Scout and SwitchWare offer extensible kernels, but to the best of our knowledge both do not support the ability to download kernel code modules in-band and on-the-fly.

B. Video Coding Schemes

Compression schemes have been proposed which do not apply block-based DCT and motion compensation against blocking artifacts occurring at high compression ratios. Schemes based on wavelet transformation are inherently adaptive due to their hierarchal decomposition of the source data. Blocking artifacts are almost nonexisting with wavelet transformation, but images tend to be blurred at high compression ratios. The theoretical foundations of wavelet-based image coding are discussed in [3] and [28]. For video encoding computational efficiency is a primary concern. WaveVideo uses integer-based filtering for calculating the wavelet transform as described in [5].

Efficient channel coding of quantized coefficients is an important issue for video coding. Besides the well known algorithms using run length encoding (RLE), huffman, or arithmetic coders and combinations thereof, Shapiro [24] proposed an optimal scheme called zero trees for coding and adaptation of the coefficients. Due to the higher computational efficiency, WaveVideo applies a modified RLE-based coding of the wavelet coefficients.

C. Video Scaling

Various papers have proposed ways for scaling motion compensated DCT-based schemes. Yeadon [29] measures the computational complexity of video scaling in the network for scaling in spatial, temporal, and color space. Furthermore, he explores fine-grained rate adaptation and requantization (SNR scaling) by means of transcoding. Rowe et al. [22] use transcoding in the Continuous Media Toolkit developed at U.C. Berkeley. Transcoding requires partial or full decompression and recompression of video and is generally performed on the application layer. Since their gateway supports only DCT-based schemes, partial decompression is required. Scaling in these systems is performed in temporal space using frame dropping without first decompressing the frames. However, some state must be kept in the nodes to account for the interdependency of the I-, B-, and P-frames. Scaling is very limited by the fact that I-frames account for about 60% of the streams and dropping them would render a full group of pictures (GOP) undecodable. Requantization to scale in the spatial direction requires undoing and then redoing entropy encoding to modify DCT coefficients. The same applies to color scaling, which can be achieved by requantization of the color channel.

Scaling is particularly important in the context of multicasting to heterogeneous receivers: McCanne et al. [15] describe a scheme called *receiver-driven layered multicast* that uses IP multicast and RTP [23] to transmit different layers of video. Receivers subscribe to a quality set by performing join experiments where they monitor their network link and add or drop layers according to the load on the link. The main advantage of this scheme is that it does not require changes in the network infrastructure (assuming the network does support IP multicast). However, sending layers on different network channels has disadvantages. Fine-grained adaptation, as offered by WaveVideo, requires up to hundreds of channels. Multiplexing and synchronizing such a high number of independent channels requires significant computing power on the end system and many of today's operating systems do not even support that many open channels. Senders connected to media gateways overcome these disadvantages by sending all layers in a single channel. For efficient association of network packets with video layers, only a simple tag is needed, adding only insignificant overhead to the aggregated volume of the video flow, far less than for example an RTP header used for synchronization.

Bhattacharjee et al. [4] show that adaptations in the network for multicast video are superior to either source or receiver adaptation. Contrary to the active network approach, scaling by transmitting different layers in independent streams has also been explored. Hoffman and Speer [13] describe an approach based on a temporal hierarchy created by multiple rates of motion JPEG (MJPEG) video streams. Receivers can apply an aggressive strategy by subscribing to all layers and by dropping some afterwards. One major drawback of this approach is the lack of scalability to large receiver groups and long response time to bandwidth fluctuations.

VI. Conclusions and Future Work

In this paper we describe our implementation of an innovative active video scaling architecture that allows for on-demand deployment of executable code in the form of plugins. Plugins are retrieved from a nearby code server and installed in the router's kernel after the sender starts transmitting video for the first time. Since plugin code runs in kernel space, this scheme is highly efficient making it suitable for data path applications with link rates of up to 100 Mbits/s.

To demonstrate the applicability of our architecture for applications requiring bandwidth in the order of several Mbits per second such as multipoint video distribution, we designed and implemented a scalable video codec based on wavelet transformation, and a high-performance video scaling algorithm executing as a plugin on the router. The plugin adapts the video stream to fit the momentary network load. If an output link is congested, the video stream is lowered to the bandwidth that the packet scheduler can guarantee. The video adaptation scheme ensures that low-frequency wavelet coefficients (which are crucial for the general definition of the image) are always forwarded but drops high-frequency parts (that describe image

details) if bandwidth becomes scarce. Our experiments show that multicast receivers gain up to 15 dB in video quality compared to best-effort forwarding. In addition, the video adaptation algorithm can react within less than 50 ms to network load fluctuations. Thus, receivers see no disturbing artifacts, motion-blur, or wrongly coded colors even on networks with very bursty traffic patterns.

Our future plans include running our architecture on the switch-based Active Network Node as described in Section II. Further, we plan to investigate other applications such as sensor data mixing, congestion control for real-time audio, and application-specific reliable multicast.

ACKNOWLEDGMENTS

We would like to thank Jon Turner for contributing the idea of using histograms for visualizing packet losses and John DeHard for helping with the demonstration setup.

REFERENCES

[1] Active Network Encapsulation Protocol, RFC, http://www.cis.upenn.edu/~switchware/ANEP/docs/ANEP.txt, July 1997.

[2] D. Alexander, et al., "The SwitchWare Active Network Architecture," In IEEE Network Special Issue on Active and Programmable Networks, May/June 1998.

[3] M. Antonio, M. Barlaud, P. Mathieu, and I. Daubechies, "Image Coding using Wavelet Transform," IEEE Trans. on Image Process., Vol. 1, No. 2, 1992.

[4] S. Bhattacharjee, E. Zegura, K. L. Calvert, "Network Support for Multicast Video Distribution," Networking and Telecommunications Group, Georgia Tech, 1999.

[5] H. Chao, P. Fisher, "An Approach to Fast Integer Reversible Wavelet Transforms for Image Compression," Computer and Information Science Inc., October 1996.

[6] M. E. Crovella, A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," Proceedings of ACM SIGMCETRICS'96, May 1996.

[7] M. Dasen, G. Fankhauser, and B. Plattner, "An Error-Tolerant, Scalable Video Stream Encoding and Compression for Mobile Computing," in ACTS Mobile Summit 1996, Granada, Spain, November 1996, Vol. 2, pp. 762–771.

[8] D. Decasper, G. Parulkar, S. Choi, J. DeHart, T. Wolf, B. Plattner, "A Scalable, High Performance Active Network Node," In IEEE Network, January/February 1999.

[9] D. Decasper. "A Software Architecture for Next Generation Routers," Ph.D. Thesis, ETH Zurich, May 1999.

[10] N. Faerber, B. Girod, J. Villasenor, "Extensions of ITU-T Recommendation of H.324 for Error-Resilient Video Transmission," IEEE Communications Magazine, June, Vol. 36, No 6, pp. 120-128, 1998.

[11] G. Fankhauser, M. Dasen, N. Weiler, B. Plattner, and B. Stiller, "WaveVideo – An Integrated Approach to Adaptive Wireless Video," in ACM Monet, Special Issue on Adaptive Mobile Networking and Computing, Vol. 4, No. 4, 1999.

[12] J. Hartman, et al., "Joust: A Platform for Liquid Software," In IEEE Networks, July 1998.

[13] D. Hoffman, M. Speer, "Hierarchical Video Distribution over Internet-style Networks," IEEE International Conference on Image Processing, Lausanne, Switzerland, Vol. III, pp. 5-8, 1996.

[14] JPEG homepage, http://www.jpeg.org/public/jpeglinks.htm.

[15] S. McCanne, M. Vetterli, V. Jacobson: "Low-complexity Video Coding for Receiver-driven Layered Multicast," IEEE Journal on Selected Areas in Communications, Vol. 16, No. 6, pp. 983-1001, August 1997.

[16] J. Meierhofer, G. Fankhauser, "Error-Resilient, Tagged Video Stream Coding with Wireless Data Link Control", in Proc. of IEEE WPMC'99, Amsterdam, Netherlands, September 1999, pp. 306-311.

[17] D. Mosberger, "Scout: A Path-based Operating System," Ph.D. Dissertation, Department of Computer Science, University of Arizona, July 1997.

[18] Official MPEG web site at http://drogo.cselt.stet.it/mpeg/

[19] MPEG-4, ISO/IEC JTC1/SC29/WG11 N2725, March 1999.

[20] ITU-T Rec. H.263, Video Codec for Low Bitrate Communication, 1996.

[21] V. Paxon, "Measurements and Analysis of End-to-End Internet Dynamics," Ph.D. Thesis, University of California, Berkeley, April 1997.

[22] L. A. Rowe, "Continuous Media Applications," Multipoint Workshop, ACM Multimedia 94, San Francisco, November 1994.

[23] RTP: A Transport Protocol for Real-Time Applications, RFC 1889, January 1996.

[24] J. M. Shapiro, "Embedded Image Coding Using Zero Trees of Wavelet Coefficients," IEEE Transactions on Signal Processing, Vol. 41, No. 12, pp. 3445-3462, December 1993.

[25] M. Shreedhar, G. Varghese, "Efficient Fair Queuing using Deficit Round Robin," SIGCOMM 95 and ACM/IEEE Trans Networking, 1995.

[26] D. Tennenhouse, et al. "A Survey of Active Network Research," IEEE Communications, January 1997.

[27] D. Wetherall, et al., "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols," In Proceedings of IEEE OPENARCH'98, April 1998.

[28] M.V. Wickerhauser, "High Resolution Still Picture Compression," Dept. of Mathematics, Washington University St. Louis, April 1992.

[29] N. Yeadon, "Quality of Service Filters for Multimedia Communications," Ph.D. Thesis, Lancaster University, Lancaster, May 1996.