Telecommunication Systems 28:2, 133–150, 2005

A Novel End-to-End Architecture for H.264 Video Streaming over the Internet *

ANTONIOS ARGYRIOU anargyr@ece.gatech.edu
School of Electrical & Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

Abstract. In this paper we propose a novel end-to-end architecture for H.264 unicast video streaming over the Internet. The proposed video streaming architecture is based primarily on a new transport layer protocol, the stream control transmission protocol (SCTP). We show that the network-friendly specification of H.264, and the novel technical characteristics of SCTP, when coupled together are able to provide a highly adaptive and flexible system for unicast video streaming. More specifically, we develop algorithms that handle at the transport layer the following functions concerning video packet transmission: retransmission policy, packet prioritization, implicit receiver feedback. We combine the above algorithms with an R-D optimization strategy at the encoder, that provides more options when adapting to bandwidth variations. This combined optimization strategy leads to more options concerning the streaming parameters. Finally, with simulation results we prove that our system is capable of maintaining good perceptual quality and TCP-friendliness under various loss conditions.

Keywords: video streaming, H.264, SCTP, R-D optimization

1. Introduction

Currently employed communication systems for video streaming over IP networks, employ UDP for transport layer services. However, as it is widely known, the absence of congestion control from UDP can cause performance deterioration for TCP-based applications if wide-scale deployment takes place in the near future [Sun and Reibman, 20]. Furthermore, it has been shown that the absence of congestion control leads to an increased number of losses for a UDP-based video streaming application [Hsiao et al., 5; Floyd and Fall, 3]. When the network is congested, the UDP sender continues to send packets at a constant rate resulting in a number of unavoidable losses. As a result, these packet losses between the video sender and the receiver significantly degrade the quality of the received signal.

A number of techniques have been developed in order to cope with this problem [Rey et al., 13; Feamster and Balakrishnan, 2; Wu et al., 25]. A number of them have been developed by IETF, and they are primarily operate at the application layer. For example, RFC 2354 [Perkins et al., 11] describes several techniques for alleviating the above problems including forward error correction (FEC) or application layer re-transmissions with RTP [Schulzrinne et al., 14]. Furthermore, methods like ARQ

* An early version of this paper appeared in CCNC 2004.

[Wang et al., 22], or error resilient adaptive coding based on back-channel messages are also proposed as methods that increase the robustness of the system under packet loss [Sun and Reibman, 20]. Nevertheless, the above techniques operate without considering congestion control and neglect the fact that the streaming application might be the one that contributed to congestion. As a result, packet losses may still be high resulting in performance deterioration. A number of approaches have been reported for equation-based video streaming in order to achieve TCP-friendliness [Floyd et al., 4; Li et al., 8]. The basic principle behind this mechanism is the approximation of the TCP sending rate behavior through measurements of RTT and packet loss rate. So the actual number of UDP packets committed to the network is less or at most equal to the approximated rate. Another similar approach, that also takes into consideration congestion control, is proposed at [Feamster and Balakrishnan, 2]. The authors propose a video streaming system that is also based on RTP-level retransmissions. Their crucial difference is that they suggest the use of a congestion manager which operates under the transport layer and makes sure that all the outgoing traffic is TCP-friendly regardless of the transport layer protocol. However, this approach is limited since it requires substantial modification of the protocol stack.

Other methods for achieving network friendliness, can be realized at the video encoder. For example, an interesting approach is rate-distortion (R-D) optimization [Liang and Girod, 9]. The advantage of this approach is that it can accommodate rate fluctuations, but there is significant processing overhead. Further efforts use R-D optimization in order to meet multiple deadlines by using back-channel messages [Kalman et al., 7]. Layered coding is also used for achieving better error resilience under packet losses [Huang et al., 6], and there are numerous variations of this method [Zhang et al., 26]. Architectures for H.264 video streaming are still primitive and are based on the RTP/UDP/IP model [1; Stockhammer et al., 17], with adapted versions of currently developed error resilience tools [Sun and Reibman, 20].

Our proposed architecture is different from the above approaches, since the transport layer protocol is the one that handles actually the video streaming process in an end-to-end fashion. In our case, this protocol is Media-SCTP, a modified version of the stream control transmission protocol (SCTP) [Stewart et al., 16]. Based on our modified protocol, we propose a new architecture for streaming RTP packetized H.264 video over the Internet. The key contributions of this paper are the following:

1. Video streaming based on a connection oriented protocol (SCTP), which controls possible retransmissions and prioritizes video packets of particular importance.
2. An algorithm for better mapping of the solutions to the R-D problem, on the fluctuating network bandwidth.
3. Exploitation of transport layer receiver feedback for pro-active frame dropping.
4. Intelligent mapping and multiplexing of H.264 structures over the proposed protocol.

The rest of this paper is organized as follows. Section 2 gives an overview of both SCTP and H.264. Section 3 presents in detail the architecture of our video streaming

system. Section 4 presents the frame dropping mechanism based on receiver feedback, while section 5 provides details of the use of R-D optimization within our protocol. In section 6 we present our experimental results, and finally, in section 7 we conclude this paper.

2. SCTP and H.264 overview

In this section we provide a brief overview of both H.264 and SCTP since they were just recently standardized.

2.1. The stream control transmission protocol

SCTP was developed by IETF, and is basically a reliable transport layer protocol that was initially designed for signaling transport. However, it soon become obvious that it has general applicability as a transport layer protocol that can operate on top of connectionless packet networks such as the Internet. One of the most exciting new features that SCTP introduced, was that of explicit support for multi-homed hosts. This means that a single SCTP association (session) is capable to use alternatively anyone of the available IP addresses of a host without disrupting an ongoing session. However, this feature is currently used by SCTP only as a backup mechanism that helps recovering from link failures. SCTP can identify these failures because it maintains a state for each remote IP address (path) by sending heartbeat messages. However, in this paper, we are especially interested in the message oriented nature of SCTP, which is substantially different from the byte-stream oriented TCP. This feature allows SCTP to decouple reliable delivery from message ordering by introducing the idea of streams. A stream, is an abstraction that allows applications to preserve in-order reliable delivery within a stream, but unordered delivery across streams. In this way, head-of-line (HOL) blocking is avoided at the receiver in case multiple independent data streams are flowing in the same SCTP session. Moreover, SCTP allows the user, to explicitly set the number of retransmissions for a specific stream to a desired value. Despite all these differences from TCP, the congestion control algorithm was defined similar to TCP, primarily for achieving TCP friendliness [Stewart et al., 16].

A recent extension to the SCTP protocol called partial reliability for SCTP (PR-SCTP [Stewart et al., 15]), is of particular use for the applications we are concerned with. The authors describe an extension to the SCTP that allows an endpoint to inform its peer that it should move the cumulative ack point (CumTSNack) forward.[1] In case both endpoints of the SCTP association support this extension, it can be used by an implementation to provide partially reliable data transmission service to an upper layer protocol. The authors present the protocol extensions which can be summed up in a new parameter for the initial session setup messages (INIT and INIT ACK messages),

[1] Cumulative TSN ack point: equivalent to TCP's largest sequence number received.

and the definition of a new FORWARD TSN message type, that provides explicit control over the receiver's CumTSNack.

2.2. The H.264 video coding standard

The H.264 video coding standard functionality is separated into a video coding layer (VCL) and a network adaptation layer (NAL). The VCL performs the traditional video encoding tasks by exploiting temporal and spatial redundancy in a video sequence [Wang et al., 23]. The NAL layer allows for the efficient encapsulation of media packets in a variety of formats that are well suited for the underlying communication channel. Details of the VCL do not concern us here since VCL's objective is to achieve high compression efficiency. What we are interested is the NAL layer. The H.264 NAL, which is considered one of the most novel features of H.264, provides a flexible mechanism of encapsulating and describing video packets at the encoder before they are delivered to lower layer protocols like RTP. NAL accomplishes that by introducing high level syntax structures that provide an abstraction to the output of the upper VCL layer. The fundamental data structure of the VCL is at the slice level, which contains macroblocks of an encoded frame. These slices are sent to the NAL layer which later produces NAL Units (NALUs). Two types of NALUs exist: simple and compound. In the case of a simple packet [Wenger et al., 24], the RTP payload consists exactly of one NALU. The available NALU types, are defined in [1], and we later pay closer attention to them. Aggregated or compound packets are used in order to capture the completely different MTU sizes of two key target networks: wireline and wireless IP networks which have different transmission unit sizes [Stockhammer et al., 17].

In order to avoid inter-packet dependence for the video decoding process, the JVT group decided to define self-contained H.264 units that can be decoded independently [Stockhammer et al., 17]. This is possible since some video streaming parameters do not change that often during a session. These may include the video format, resolution, entropy coding method, aspect ratio, etc. These are called the parameter set, in H.264 terminology, and can be send out of bound during the setup of a session.

3. System architecture

The reasonable question for someone to ask is why use a reliable transport protocol like SCTP for video transport. SCTP possesses a number of features that make it quite different from TCP. The stream based nature of SCTP offers multiple levels of reliability and multiplexing for data provided by the application. The advantage of multiplexing a number of application layer streams into the same end-to-end connection, resides in the improvement of the application performance, and of course improves the scalability of a streaming server/client. Moreover, in the case where the application provides H.264 video encoded data, new possibilities arise concerning the retransmission policies of H.264 structures that are mapped to individual streams. Tradeoffs between video quality and end-to-end delay come into play when these ideas are applied. Our system proves,

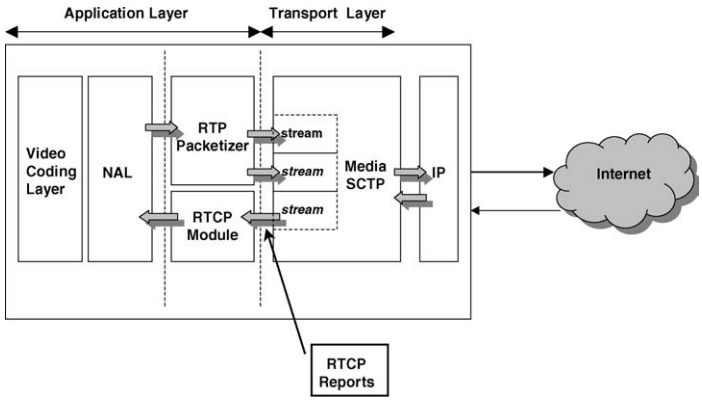

Figure 1. Streaming server architecture.

that with a careful system design, that takes into consideration the tradeoffs mentioned above, we can create an H.264 video streaming system based on a reliable transport layer protocol.

Figure 1, shows a simplified diagram of the proposed system architecture: The VCL produces slices which are sent to the NAL layer. The NALUs that are produced there, are then sent to the RTP packetizer. RTP packets are mapped to SCTP streams and their desired level of reliability is set in the way that we will explain in a while. SCTP then multiplexes DATA chunks that belong to different streams over the same SCTP session and passes them to IP. On the reverse path, the streaming server collects acknowledgments for the video packets that were sent reliably, and also collects possible RTCP reports which are sent directly to the encoder. Note that some streams of the same SCTP session may be sent reliably while others not.

3.1. Tuning reliable delivery

Video communication systems, are probably the one class of network applications that can sustain the highest packet loss rate. Reliable packet delivery it is not so critical since error concealment can "hide" the effects of packet losses. However, packet loss can be sustained until a specific threshold, after which quality deterioration is increased rapidly [Sun and Reibman, 20].

Loss recovery in our system is completely different from application layer retransmissions as we earlier described, since the transport layer takes over and makes retransmissions seamless for the RTP payloads. This is feasible, since we allow the application to map a specific class of messages to a specific SCTP stream. Each stream, will transmit the data according to application requirements. For example, in the case of H.264, a coarse mapping would be to classify NALUs that are more important (e.g., parameter set NALU) as normal reliable messages, while single slice containing NALUs could be sent completely unreliable. Various degrees of reliability can be specified in accordance with SCTP partial reliability extension [Stewart et al., 15].

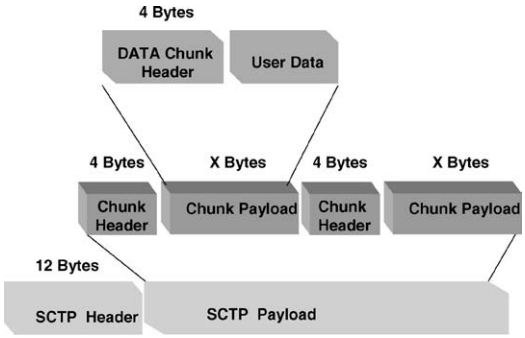

Figure 2. SCTP DATA chunk multiplexing.

Priority is enforced by the stream id. Effectively this means that the highest the stream number (0 is the highest), the higher the priority. Only completely reliable streams, the default case in SCTP, they are never dropped since the sender knows that they constitute important information. Moreover, in the case of Media-SCTP, when congestion happens, packets that reside in the output buffer are "dropped" when their priority is low. Even if this practice may lead to fluctuations in the video playout rate, it ensures that important data will be delivered, allowing thus the maintenance of a steady video quality threshold at the receiver.

3.2. Multiplexing

Currently the H.264 specification assigns one RTP packet per NALU, for both simple and compound packets. The granularity at which multiplexing is performed, depends on the application requirements. An SCTP session is able to avoid HOL blocking for data that belong to different streams. These streams could either be multiple media streams encapsulated in RTP packets (figure 2). Moreover, if we go one step further, actual H.264 video encoded data will be send in a separate stream, allowing the receiver to deliver data to the decoder without waiting for parameter set updates, which are delivered at a different stream. In this way it is possible to transmit in-band various parameters useful for the decoder. More specifically, the parameter set idea that was introduced with H.264, is an example of a control data types that might need to be transmitted in-band during an ongoing session. In this way, the regular decoding process is not interrupted since its data are received in a separate stream. Furthermore, an SCTP session can start with a number of streams that is equal to the types of NALUs. This gives another degree of freedom the the receiver/decoder set which can avoid HOL blocking between NALU types.

3.3. Levels of reliability and multiplexing

As we said, the proposed transport system is able to provide different levels of reliability and data multiplexing. We focused on the following three possibilities: (1) NALU level multiplexing according to their type [1], and (2) according to frame type (I, P, B).

NALU level. A NALU consists of one byte header followed by a bit-stream that contain macroblocks of a frame. The NALU type, specifies which type of data structure is contained in a NALU. All the NALUs that contain VCL bitstream data have id from 1 to 5, while the rest id from 6 to 12 and they contain parameters that need to be sent periodically. Clearly the large number of NALU types gives many possibilities for performing fine tuning over their delivery. In practice however, not all these types are sent during one streaming session and they are not even sent with the same frequency.

Frame type level. Temporal scalability involves separating of the video frames into a number of layers (base and enhancement) that have different temporal resolution. For example, in our case, temporal scalability is applied through the use of frame types: I, P and B. I and P frames could be considered as the base layer, while B frames constitute the enhancement layer. Our system, sends frames with a degree of reliability that corresponds to their importance. In our simulations we experimented with different re-transmission strategies having in mind that the importance order is I, P and B.

4. Frame dropping filter based on receiver feedback

Efficient bandwidth management is also performed by another subcomponent of our protocol. The proposed mechanism is a frame dropping filter, that requires close coupling in the operation of the playout buffer and the application at the receiver. Assume that the receiver has received few of the packets that carry an I frame. When the playout buffer determines that the playout deadline for this specific frame has been missed, it sends immediately a FWD-TSN [Stewart et al., 15] message to the video server. As soon as the server receives this packet, it stops all the attempts to retransmit the lost (or delayed) packet since it is useless to the receiver now. The FWD-TSN will arrive at the sender around after time RTT/2. Our goal is to avoid RTO timeouts at the sender and more important to avoid halving of the congestion window. However, if the deadline is not missed, and the receiver is about to send a SACK packet, then two cases arise: if

$$\frac{\mathrm{RTT}}{2} \leqslant playout\_time - current\_time \tag{1}$$

then the receiver must send a SACK including the missing chunks, and of course the one that is needed for the frame playout. However, if the inequality does not hold, then the receiver "tricks" the sender by not including this TSN segment as missing in the SACK packet (if it has not already send a FWD-TSN as described previously). In this way the sender thinks that the receiver got the specific TSN segment. Why we do that? In the second case, even if the sender receives the SACK and retransmits the missing TSN, it will arrive after the playout deadline, making it thus useless. Figure 3 depicts clearly this fact.

However, if a number of total packets that carry a frame have been lost, the Media-SCTP protocol can deliver the data that are available if the application can use them and

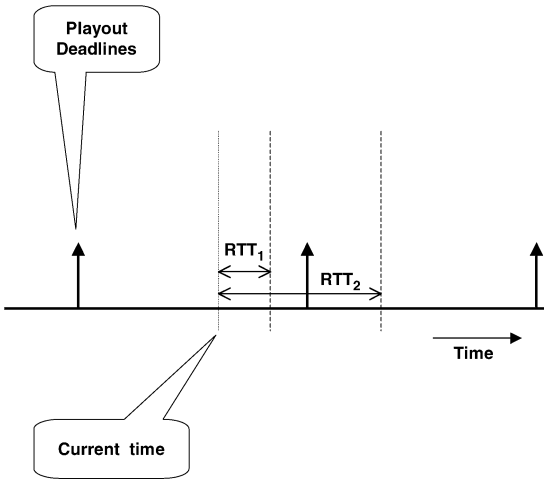

Figure 3. Link with RTT_1 will meet the deadline but not link with RTT_2.

conceal the missed slices. It is up to the application to make this decision, and we plan to realize this is the future through the SCTP socket interface.

5. R-D optimization

Rate control at the encoder, is not a part of H.264. However, guidelines have been issued for the implementation of R-D optimization algorithms as part of an H.264 compatible encoder. It has also been shown that the good performance of the H.264 reference software is largely due to the use of R-D optimization based on Lagrangian optimization [Wang et al., 23; Suehring, 19]. Objective of Lagrangian optimization, is to minimize distortion D subject to a bitrate constraint:

$$\min(D) \quad \text{subject to: } R \leqslant R_c, \tag{2}$$

where D, R and R_c are the distortion, bitrate and target bitrate, respectively. An optimal solution for this minimization problem can be obtained using Lagrangian optimization, where the cost function is

$$J = D + \lambda R. \tag{3}$$

Finding the coding parameters that minimize J has attracted considerable research work, since it also a very complex and time consuming task. It has been determined experimentally the following relation between λ and the quantization parameter (QP) [Stockhammer et al., 18]:

$$\lambda \simeq 0.85 \times 2^{(QP-12)/3}. \tag{4}$$

This equation does not depend on the actual video content such as spatial or temporal complexity. Given a QP, a near optimal parameter λ can be selected. In our case now,

the R_c bitrate constraint is translated to the actual bandwidth available in the network, which is determined as the value of the congestion window over RTT. The algorithm evaluates rate and distortion for all possible modes and types of a macroblock. Next, the type and mode that minimizes the Lagrangian cost given the current bandwidth constrain is selected. However, since bandwidth fluctuations may be abrupt, we experimented with lower expectations concerning the available bandwidth (10% less).

6. Simulation results

The purpose of our experiments is twofold. First to show that each of the mechanisms of our protocol achieves good perceptual quality at the receiver, and outperforms standard RTP/UDP based systems that use methods like ARQ and layered coding. And second, to prove that our protocol is TCP-friendly and can be practically deployed.

6.1. Simulation setup

All the simulations were performed with the use of the ns-2 network simulator [McCanne and Floyd, 10]. The network topology used throughout our experiments is shown in figure 4. In this figure, the sources are numbered as S1, S2, . . . , SN, while the destinations with the letter D. The bottleneck link between the two routers has bandwidth 1 Mbps and delay of 10 ms. The links between the nodes and the routers have bandwidth of 500 Kbps and delay 10 ms.

We essentially simulated a real-time unicast video streaming system by making the valid assumption that sender buffer does not suffer from overflow or underflow [Sun and Reibman, 20]. The TML H.264 encoder [Suehring, 19] was used in order to produce the H.264 video sequences, and the corresponding decoder was used at the receiver in order to play the sequences and calculate PSNR. We used the QCIF video sequences *container* and *foreman* as an input to the encoder [21]. The encoder output was the sequence IBPBPB. This was done because our simulations involved packet losses. We used the following approach at the receiver: When a packet is received and its checksum is checked at the IP and SCTP layers, if there is an error the packet payload (in our

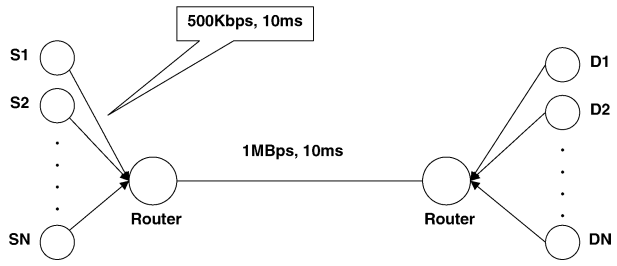

Figure 4. Simulation topology.

case RTP packets) is discarded. The quality of the video sequence at the decoder was measured using the Peak Signal to Noise Ratio (PSNR):

$$\text{PSNR} = 20 \log \frac{255}{\sqrt{\text{MSE}}}, \tag{5}$$

where the mean square error (MSE) is

$$\text{MSE} = \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (p_{ij} - \hat{p}_{ij} OB)}{MN} \tag{6}$$

with M and N equal to 176 and 144 representing the dimensions of the QCIF frame while p_{ij} and $\hat{p}_{ij}$ represent the original and predicted pixel values.

6.2. *Selective retransmission experiment*

PSNR vs. packet loss rate. It is known that the loss of I frames results in considerable quality degradation at the receiver. Moreover, in the case of data partitioning, increased I picture losses lead to picture drift effects. In this experimental setup, we used the basic H.264 version without error resilience support [1], when using both UDP or SCTP. We did that because we wanted to test only the effects of varying level of reliability versus the PSNR.

Figure 5 depicts results for various packet loss rates ranging from 0 to 10%. The performance degradation of UDP-based transport was something to be expected, especially in this case where we are not using error-resilient coding. However, in the case of Media-SCTP interesting observations can be made: In the first case reliability has been set to 1, which means that at maximum, one retransmission will take place for every SCTP packet. Since this retransmission policy covers every frame, we are not

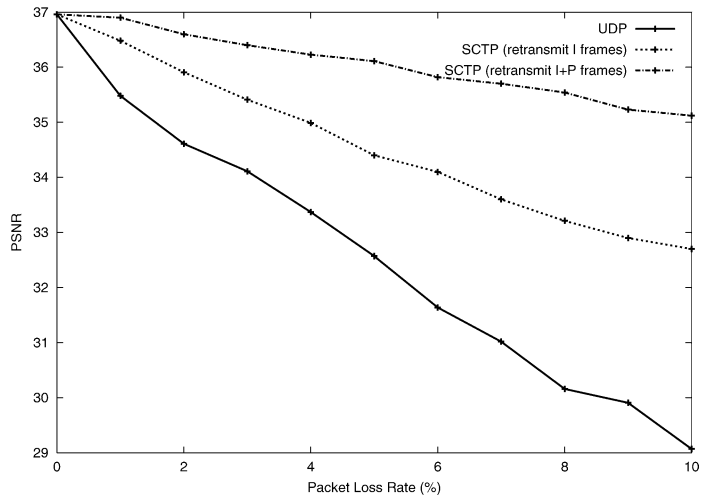

Figure 5. PSNR for various packet loss rates.

surprised that Media-SCTP delivers all the frames during the simulation resulting in a stable PSNR. Interesting tradeoffs come into play when we configured reliability for I frames to 1, while for P and B frames was set to 0, meaning that P and B frames would sent unreliably without requiring acknowledgment. A number of P and B packets are lost and that is why PSNR at the receiver is lowering as the packet loss rate is increased. Nevertheless, I frames are retransmitted when are lost, resulting in a better quality than in the case of UDP. We can also see from figure 5 that when both I and P frames are sent reliably with one retransmission, quality improvement is low making the need for B frame retransmission questionable.

The achieved frame rate is shown in figure 6. Obviously, the reliable nature of Media-SCTP does result into higher delays than UDP and so the adaptive playout algorithm reduces the playout frame rate. The standard SCTP frame rate drops sharply around packet loss 12%. This is happening because as the packet loss rate is increased SCTP experiences timeouts which result in large time before the next retransmission takes place and so the frame rate drops substantially. On the other hand, Media-SCTP preserves the desired behavior, as also UDP does, and the frame rate drops linearly. This is because we allow only one retransmission (I and P packets here) avoiding thus RTO timeout problems.

Adaptive playout buffer. Despite a careful configuration of a retransmission policy, the reliable nature of SCTP results in larger delays than UDP. This fact eventually necessitates the use of a larger playout buffer at the receiver. Moreover, delay variation (jitter), can create large fluctuations at the receiver buffer. An adaptive playout buffer is capable of accommodating delay fluctuations in the packet reception, and this is what we need here. With an adaptive playout algorithm, in the short-run the playout frame rate is kept relatively constant, while it may change significantly over time depending on network

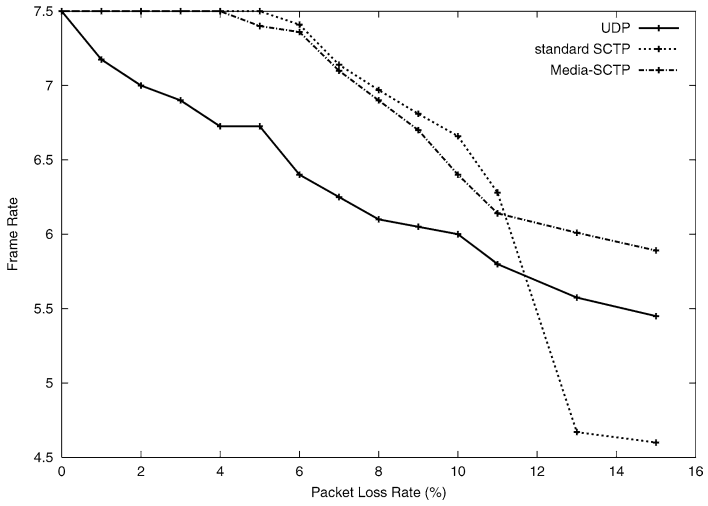

Figure 6. Achieved frame rate for various packet loss rates.

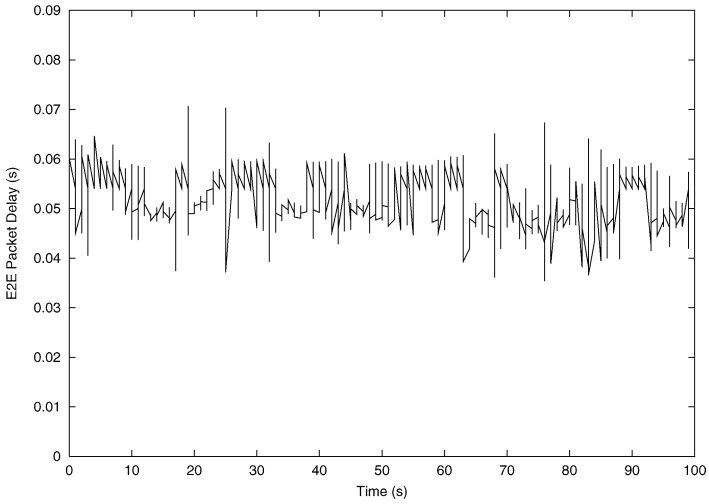

Figure 7. End-to-end packet delay for a UDP-based system.

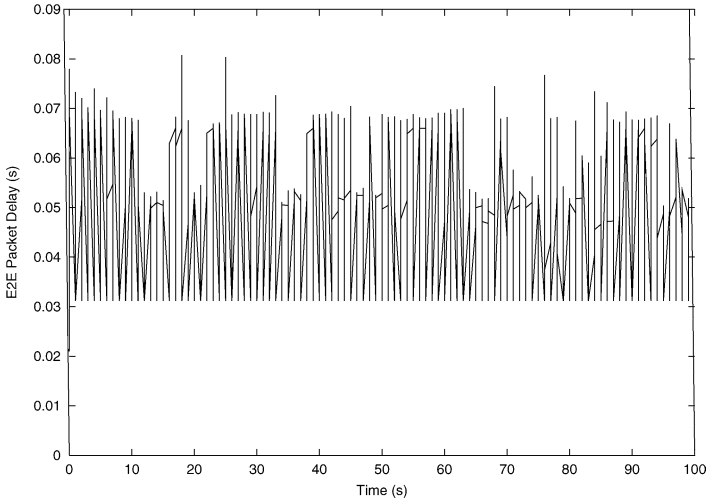

Figure 8. End-to-end packet delay for a Media-SCTP system.

conditions. This type of playout buffer algorithm is well suited for a reliable transport service, like ours, since it can conceal any delay variations. This algorithm we used was proposed by [Ramjee et al., 12].

For this experiment we added TCP-cross traffic of 2 FTP flows. Figures 7 and 8 depict end-to-end delay when UDP and Media-SCTP are used at the transport layer. We do not show results for vanilla SCTP due to the lack of space. Nevertheless, SCTP has end-to-end delay higher than both these two figures because it offers a fully reliable delivery service. However, it is also clear that the reliable nature of Media-SCTP necessitates the use of an adaptive playout algorithm.

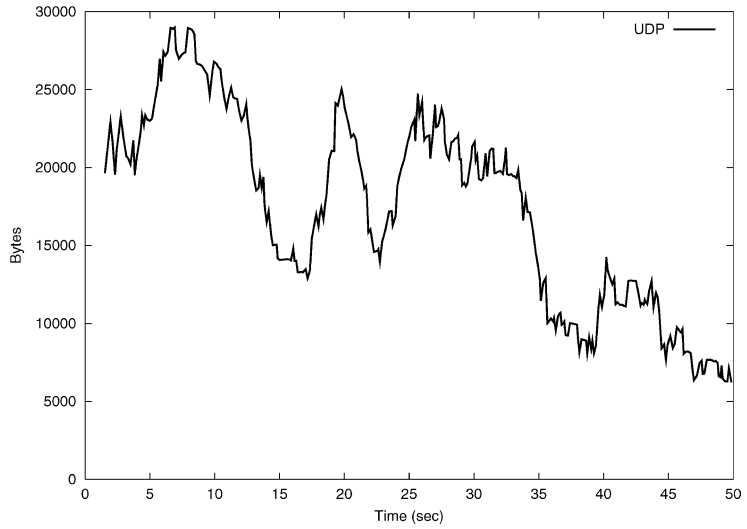

Figure 9. Buffer size evolution over time at the receiver for a UDP-based system.

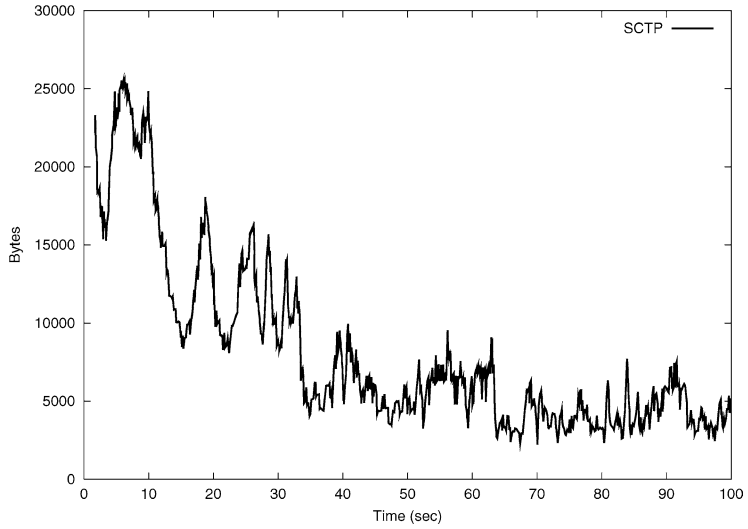

Figure 10. Buffer size evolution over time for a Media-SCTP system.

Results for the buffer size versus time for UDP are shown in figure 9. Ideally the UDP-based system would maintain a relatively fixed varying range for the buffer size. Nevertheless, the FTP/TCP cross-traffic results into congestion for some periods and this further creates UDP packet losses. That is why we observe the playout buffer drain for some periods. However, for the case of SCTP (figure 10) we can see that for the first seven seconds, they buffer occupancy is relatively high and in addition data are removed with low frequency. This means of course a low playout frame rate at the receiver. However, when congestion incident has passed (around 10 sec), the end-to-end

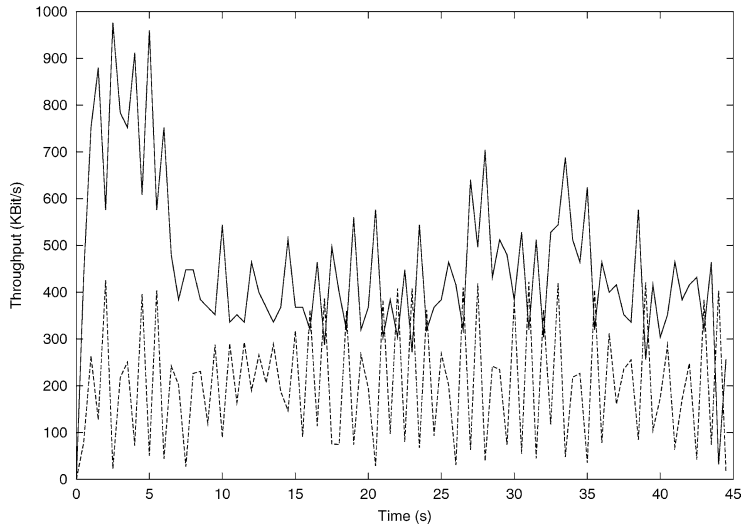

Figure 11. Instantaneous throughput for one Media-SCTP and one FTP/TCP flow.

delay is reduced and our system is capable of delivering more frames. In addition, the playout algorithm adapts to the lower delay, and provides a higher frame rate and reduced buffer requirements. This results, are very important, since they depict the importance of having a good playout buffer algorithm when a reliable video streaming system is employed.

6.3. TCP-friendliness experiment

Despite the existence of the other TCP sessions, Media-SCTP is capable of preserving good quality and relatively stable frame rate. We found that in the face of congestion Media-SCTP behaves better than UDP and incurs smaller number of losses. We turned off the timeout timers from SCTP, and required only one retransmission for every frame. This resulted into higher PNSR, but we avoided the undesirable effect of RTO timeouts for further retransmissions which would lead to large delays. In this experiment we used cross-traffic of three TCP flows competing with one Media-SCTP flow. Figure 11 shows the instantaneous throughput of only one TCP flow and the Media-SCTP flow for clarity purposes. All of them obtain a fair share of the bandwidth. After packet losses start to happen due to congestion at bottleneck router R_1, Media-SCTP tries only one more retransmission of a specific packet and then proceeds with new frames since congestion has passed. That is why we do not see large throughput variations.

6.4. R-D optimization experiment

Earlier we described the R-D optimization algorithm used by the default H.264 software implementation [Suehring, 19]. In this subsection, we used this algorithm in order to performer the simulations. Figure 12 depicts results concerning the evolution of PSNR

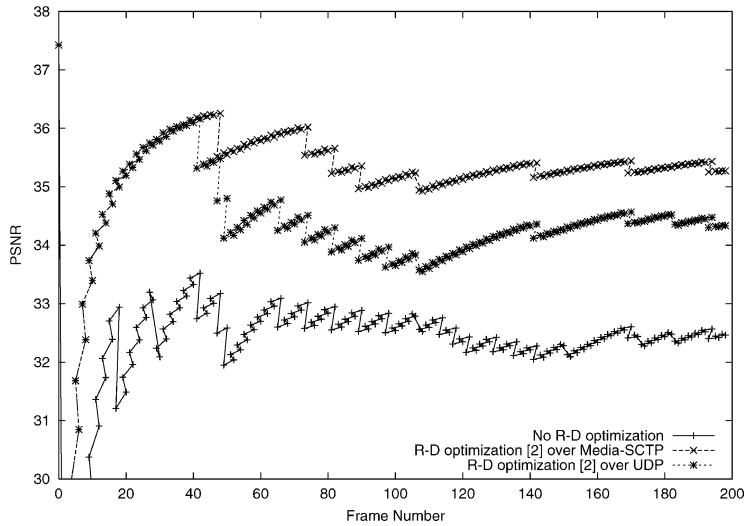

Figure 12. PSNR vs. frame number.

with respect to frame number at the receiver. In this experiment we attached 3 FTP/TCP flows at the bottleneck router R_1, in addition to the one Media-SCTP flow. In this way all the losses come from congestion events during the video streaming session. In figure 12, RTT delay is small and so the retransmission of parts of intra coded MBs help the receiver to improve PSNR. However, larger delays will have bad effects as we will later see.

We also we can observe some sudden drop in PSNR for both UDP and Media-SCTP at time 40 sec and 43 sec, respectively. Both of them suffered a burst loss during the simulation at that time. Even fast retransmissions with Media-SCTP were dropped at the router due to congestion. However at time 44 sec there is just a two packet loss for an I frame with UDP, resulting into further substantial PSNR decrease. On the other hand, Media-SCTP is using its retransmission for the intra macroblocks and additionally use a little coarser QP resulting in an improved PSNR. The achieved frame rate for these simulation was 23 fps with the fixed playout rate algorithm. So the important conclusions is that despite using a little coarser QP with Media-SCTP, the protocol does not contribute to congestion while with the UDP-based system rate control, the bitrate cannot match precisely the fluctuating bandwidth.

6.5. Intelligent frame dropping filter experiment

As we saw in the previous experiment, a lot of retransmission are wasted if we do not take into account the playout time for the corresponding frame. We tested the mechanism that we proposed in section 4 in order to check whether it could solve this problem.

In this first experiment we had six FTP flows and one Media-SCTP flow. In figure 13 the solid curve is the difference between the projected playout time and the receive time of a specific frame and in the x axis is the reception time of the frame. At the same

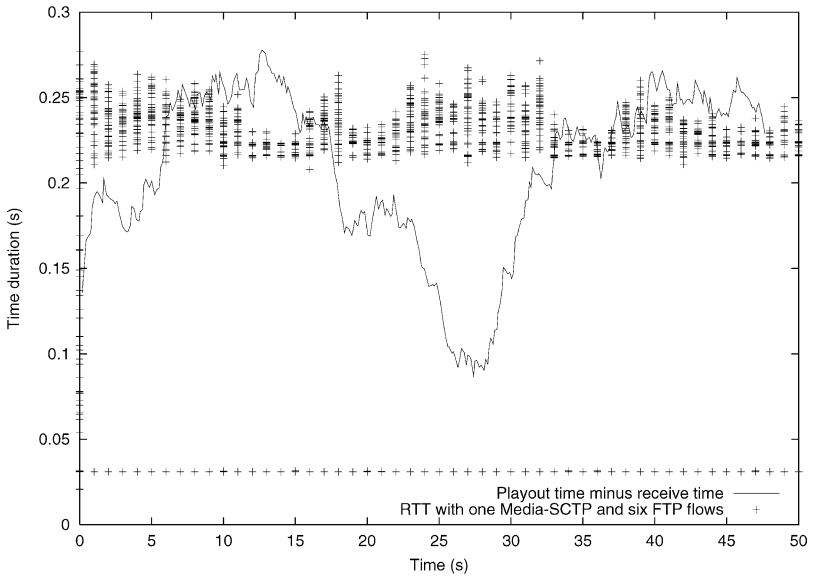

Figure 13. Projected playout time minus receive time of a specific frame over the reception time of the frame and RTT samples.

time, we see RTT samples in the same figure for the same end-to-end Media-SCTP connection. When the RTT is higher at the time a specific frame was received, this means that if a loss was discovered (through gap reports in a SACK), a retransmission from the sender would be worthless as we earlier described. This experiment supports our claim that this phenomenon does actually happen even for low number of contenting flows as it is the case in this experiment.

So our system whenever identified the above situation, the receiver sent a FWD-TSN message informing the sender to advance the CumTSNack, and to skip thus the retransmission. According to figure 13, around half of the time (25 sec) condition RTT is higher than playout time minus receive time. This means that with a loss rate of $\alpha\%$ uniformly distributed over the 50 sec time frame, and packet size of β bytes we will save bandwidth of: $\alpha \times (25/50) \times \beta$ bytes. Assuming a packet size of 1500 bytes (e.g., IEEE 802.3 LAN), and a packet loss rate of 1% the bandwidth saved is $1/200$ of the total data sent. Or we save half of the retransmissions. By saving this bandwidth we obtained better frame rate which varied between 14.5 and 17.5 when the adaptive playout algorithm was used.

Finally, figures 14(a) and (b) show two sample frames captured during the last experiment. It is clear that the retransmissions saved some macroblocks (MBs) and the receiver produced the sequence with a relatively better perceptional quality when using Media-SCTP. However, with UDP even when we used error concealment, the burst losses at the bottleneck suffered substantial degradation (even worse than the captured frame depict).

(a) (b)

Figure 14. Representative frames captured during the simulation experiments: (a) RTP/UDP; (b) RTP/ Media-SCTP.

7. Conclusions

In this paper we proposed the use of a new transport layer protocol for streaming H.264 video over the Internet. Our vehicle is SCTP which provides the capability of fine-grained retransmission policies and multiplexing at the transport layer. This features allowed the design of better algorithms concerning video packet retransmission, allocation of bandwidth after R-D optimization, and intelligent use of receiver feedback reports. We showed that even when a reliable transport protocol is used careful combination of optimization strategies can lead to good solutions that are TCP-friendly. Our future work will focus on experimenting with scenarios that involve more complicated network topologies that represent closer the current Internet.

References

[1] Advanced video coding for generic audiovisual services, ITU-T recommendation H.264 (2003).
[2] N. Feamster and H. Balakrishnan, Packet loss recovery for streaming video, in: *12th Internat. Packet Video Workshop*, Pittsburgh, PA (2002).
[3] S. Floyd and K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Transactions on Networking (1999).
[4] S. Floyd, M. Handley, J. Padhye and J. Widmer, Equation-based congestion control for unicast applications, in: *ACM SIGCOMM*, Stockholm, Sweden (2000) pp. 43–56.
[5] P.-H. Hsiao, H. Kung and K.-S. Tan, Streaming video over TCP with receiver-based delay control, IEICE Transactions on Communications, Special Issue on Internet Technology (2003).
[6] J. Huang, C. Krasic, J. Walpole and W.C. Feng, Adaptive live video streaming by priority drop, in: *IEEE Conf. on Advanced Video and Signal Based Surveillance* (2003) pp. 342–347.
[7] M. Kalman, P. Ramanathan and B. Girod, Rate-distortion optimized video streaming with multiple deadlines, in: *Internat. Conf. on Image Processing (ICIP)* (2003).
[8] Z. Li, G. Shen, S. Li and E. Delp, L-TFRC: An end-to-end congestion control mechanism for video streaming over the Internet, in: *ICME*, Vol. 2 (2003) pp. 309–312.
[9] Y. Liang and B. Girod, Prescient R-D optimized packet dependency management for low-latency video streaming, in: *Internat. Conf. on Image Processing* (2003).
[10] McCanne and S. Floyd, NS network simulator, `http://www.isi.edu/nsnam/ns/` (2004).
[11] C. Perkins et al., Options for repair of streaming media, RFC 2354 (1998).

[12] R. Ramjee, J. Kurose, D. Towsley and H. Schulzrinne, Adaptive playout mechanisms for packetized audio applications in wide-area networks, in: *IEEE INFOCOM*, Toronto, Canada (1994).
[13] J. Rey, D. Leon, A. Miyazaki, V. Varsa and R. Hakenberg, RTP retransmission payload format, `draft-ietf-avt-rtp-retransmission-07.txt` (2003).
[14] Schulzrinne, Casner, Frederick and Jacobson, RTP: A transport protocol for real-time applications, `draft-ietf-avt-rtp-new-12.txt` (2003).
[15] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen and P. Conrad, SCTP partial reliability extension, `draft-ietf-tsvwg-prsctp-00.txt` (2003).
[16] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang and V. Paxson, Stream control transmission protocol, RFC 2960 (2000).
[17] T. Stockhammer et al., H.26L/JVT coding network abstraction layer and IP-based transport, in: *Internat. Conf. on Image Processing*, Rochester, NY (2002).
[18] T. Stockhammer, M. Hannuksela and T. Wiegand, H.264/AVC in wireless environments, IEEE Transactions on Circuits and Systems for Video Technology (2003) 657–673.
[19] K. Suehring, H.264/AVC software coordination, available from `http://bs.hhi.de/~suehring/tml/` (2004).
[20] M.-T. Sun and A.R. Reibman, eds., *Compressed Video Over Networks* (Marcel Dekker, New York, 2001).
[21] TML video sequences, available from `http://www.stewe.org/vceg.org/sequences.htm` (2004).
[22] C.-H. Wang, R.-I. Chang, J.-M. Ho and S.-C. Hsu, Rate-sensitive ARQ for real-time video streaming, in: *Global Telecommunications Conf. (GLOBECOM)*, San Franscisco, CA (2003).
[23] Y. Wang, J. Ostermann and Y.-Q. Zhang, eds., *Video Processing and Communications* (Prentice-Hall, Englewood Cliffs, NJ, 2002).
[24] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund and D. Singer, RTP payload format for H.264 video, `draft-ietf-avt-rtp-h264-08.txt` (2004).
[25] D. Wu, T. Hou, W. Zhu, H.-J. Lee, T. Chiang, Y.-Q. Zhang and H.J. Chao, On end-to-end architecture for transporting MPEG-4 video over the Internet, IEEE Transactions on Circuits and Systems for Video Technology (2000).
[26] F. Zhang, M. Pickering, M. Frater and J. Arnold, Optimal QoS mapping for streaming video over differentiated services networks, in: *IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)* (2003).