Practical Channel-adaptive Video Streaming with Fountain Codes

Shakeel Ahmad*, Raouf Hamzaoui†, Marwan Al-Akaidi†
* Department of Computer and Information Science, University of Konstanz, Germany.
Email:shakeel.ahmad@uni-konstanz.de
† School of Engineering and Technology, De Montfort University, Leicester, UK.
Email:{rhamzaoui,mma}@dmu.ac.uk

***Abstract*— Video streaming is sensitive to packet loss, which can severely damage the quality of the received video. Video communication systems that rely on application-layer forward error correction (FEC) to combat packet loss are particularly suitable for pervasive computing because they can be used on top of any existing network architecture. However, since in heterogeneous environments network conditions are unpredictable, determining the right amount of redundancy introduced by the channel encoder is not obvious. This paper presents a practical implementation of a unicast video streaming system that solves this problem by using a rateless code and receiver feedback. In real simulations over the Internet our solution outperformed a standard approach based on fixed-rate forward error correction. For an Internet connection Konstanz-Beijing-Konstanz and the standard Foreman sequence compressed with the H.264 video coder, the gain in average peak signal to noise ratio exceeded 3.5 decibels at 90 kilobits per second.**

I. Introduction

An increasing number of video streaming systems are exploiting application-layer forward error correction (FEC) to combat packet loss [1], [2], [3], [4]. However, these work use fixed-rate FEC, where for a given source block the channel code rate is fixed or updated according to a prediction based on past observations of the packet loss rate. Unfortunately, the packet loss rate in the Internet and packet-based wireless networks is hard to predict and can rapidly change over time. Thus, the performance of fixed-rate FEC schemes may be poor because of the unavoidable mismatch between the actual packet loss rate and the predicted one.

In [5], we proposed a new approach that addresses this problem by using rateless codes [6], [7] instead of fixed-rate codes. With rateless codes, the code rate does not have to be fixed in advance as the encoder can generate on the fly a potentially infinite sequence of encoded symbols. Another advantage is that both the encoding and decoding times are much lower than those of standard fixed-rate erasure codes (e.g., Reed-Solomon codes). The basic idea of our approach is that for every source block, the sender keeps on sending the encoded symbols until an acknowledgement is received from the receiver or the transmission time for the source block elapses. But as the acknowledgment needs time to reach the sender, the sender may transmit redundant encoded symbols. We showed how to construct transmission strategies that minimize this overhead, while ensuring successful reconstruction of the video stream subject to an upper bound on the packet loss rate. However, our results were given for a hypothetical rateless code that assumes perfect recovery of the k original source symbols if and only if at least $k(1+\epsilon)$ encoded symbols are received. Here ϵ is a small real number that gives the trade-off between the error recovery property of the code and the amount of redundancy it introduces. This assumption is only an approximation as rateless codes are probabilistic. In [5], we also assumed for simplicity that the feedback channel is reliable and that the transmitted packets consist of a single symbol. Another limitation was that the performance of the system was evaluated analytically only using the expected overhead and an expected outage rate that corresponds to the expected number of times the source block cannot be recovered. Finally, the work in [5] assumed that a probability mass function of the channel packet loss rates is available before the transmission begins.

In this paper, we extend the results of [5] by addressing all these limitations. The goal of the work is to show the practical relevance of our approach and its actual benefits for streaming video data. We provide an implementation using a real Fountain code. We use real packet transmissions over an Internet connection. In contrast to [5], packets have more than one symbol and packet losses and delays also occur in the backward channel. Also we show how our system can be used when the packet loss rate histogram is not available. We compare our system to two traditional schemes. The first one uses fixed-rate FEC. The second one also uses fixed-rate FEC, but the code rate is updated regularly according to the channel conditions.

The remainder of the paper is organized as follows. In Section II, we describe the video streaming system introduced in [5]. In Section III, we derive analytical expressions for the expected used bandwidth of the studied schemes and compare them with simulation results that use an LT code [6]. Section IV compares the performance of the studied schemes for streaming H.264 [8] compressed video over an Internet connection Konstanz-Beijing-Konstanz.

II. Previous work

A. Video streaming system

Figure 1 shows the system introduced in [5]. The raw video stream produced by a camera from time $t = 0$

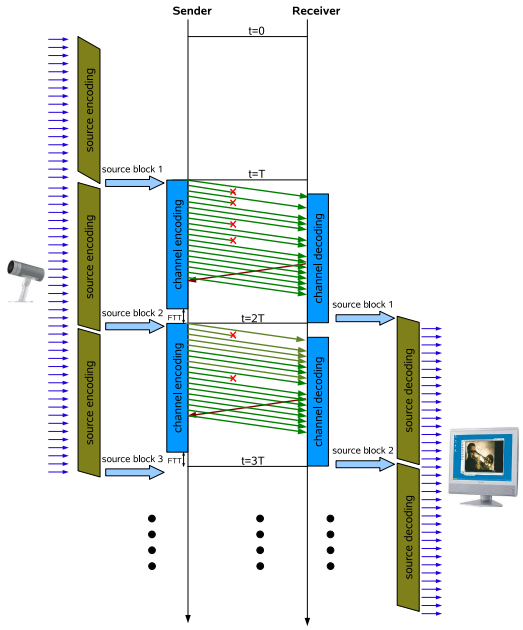

Fig. 1. Video streaming system.

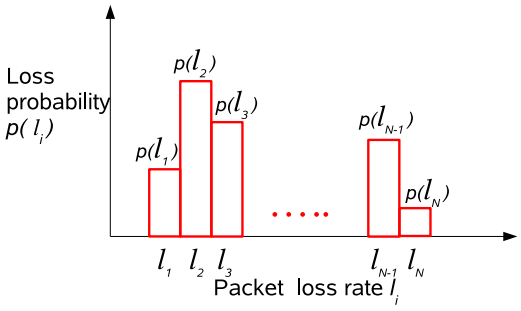

Fig. 2. Probability mass function of observed packet loss rate. A packet is considered to be lost if it is not available at the receiver within the transmission interval.

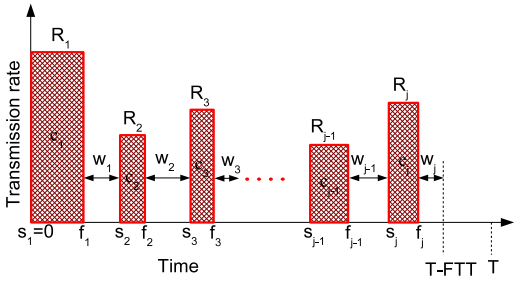

Fig. 3. Proposed transmission strategy. The encoded symbols are transmitted at rate R_i from s_i to f_i, followed by a waiting time of w_i, $i = 1, \ldots, j$.

to $t = T$ is fed into the source encoder to produce the first source block. For simplicity, we ignore the video encoding time, which is usually very small and depends on the particular implementation of the source encoder. A source block b consists of k symbols of size s each. At $t = T$, the k source symbols are encoded by applying a rateless code to produce a potentially infinite sequence of encoded symbols, each of size s. These encoded symbols are transmitted over the channel after encapsulating them in channel packets. A channel packet may contain one or more encoded symbols. Some of the channel packets are lost or arrive at the receiver too late to be useful. The receiver tries to recover the source block. If it succeeds, then an acknowledgement is sent to the transmitter and the source block is fed into the source decoder at $t = 2T$. Source decoding can be done with almost no delay providing the first byte of decoded video stream for playback at $t = 2T$, which ensures a maximum playback latency of $2T$. The same process is repeated for the next source blocks. The source blocks are encoded independently, which can be achieved, e.g., by starting each one with an intra (I) frame.

B. Transmission strategy

Since the acknowledgement needs time to reach the transmitter, the transmission strategy introduces an *overhead* equal to the number of unnecessary encoded symbols sent to the receiver (these are the encoded symbols transmitted after the encoded symbol that allowed the receiver to recover the source block). The transmission strategy is also characterized by an *outage rate* equal to 0 if the source block is successfully decoded and 1, otherwise. An optimal transmission strategy should minimize the expected overhead subject to a given expected outage rate. This problem was addressed in [5]. In the following, we describe the solution proposed there. For simplicity, the description assumes that a channel packet contains only one encoded symbol.

We assume that the channel is described by N packet loss rates $l_1 < \ldots < l_N$ with probabilities $p(l_1), \ldots, p(l_N)$ (Figure 2). We also assume that the rateless code is such that the receiver can recover source block b correctly if and only if at least $k \times (1 + \epsilon)$ encoded symbols for this block are received on time. For each $j \in \{1, \ldots, N\}$, we build a class of transmission strategies as follows.

Let $R_{\max}$ denote the channel bandwidth [9] (or capacity limit). From $t = s_1 = 0$ to $t = f_1$ we transmit at a rate R_1 such that $R_1 \times (f_1 - s_1) = c_1$. Here $c_1 = (k \times (1 + \epsilon))/(1 - l_1)$ is the number of encoded symbols that have to be transmitted to guarantee successful decoding if the packet loss rate is l_1. If we denote by RTT the round trip time, an acknowledgement is expected to arrive at time $a_1 = f_1 + RTT$. Since any symbol transmitted from f_1 to a_1 may contribute to the overhead, we wait some time w_1 until $s_2 = f_1 + w_1$ before transmitting again at a rate R_2.

Similarly, we transmit at rate R_2 from s_2 to f_2 such that $R_2 \times (f_2 - s_2) = c_2 = (k \times (1 + \epsilon))/(1 - l_2) - c_1$. The same procedure is repeated, giving transmission rates $R_1, \ldots, R_j$ $(0 < R_i \leq R_{\max},\ i = 1, \ldots, j)$ and waiting times $w_1, \ldots, w_j$ $(0 \leq w_i \leq RTT,\ i = 1, \ldots, j)$, where each transmission rate R_i, $1 \leq i \leq j$, starts at s_i and finishes at f_i (Figure 3) with

$$c_i = (k \times (1+\epsilon))/(1-l_i) - \sum_{m=0}^{i-1} c_m \quad (1)$$

$$R_i \times (f_i - s_i) = c_i \quad (2)$$

$$s_i = f_{i-1} + w_{i-1} \quad (3)$$

where $c_0 = f_0 = w_0 = 0$. Finally, we add the condition

$$f_j \leq T - FTT, \quad (4)$$

where FTT is the forward trip time. This condition states that all encoded symbols are sent within the available time budget.

Equation (1) ensures successful decoding if the packet loss rate l is smaller than or equal to l_j. It therefore guarantees that the expected outage rate is equal to $1-\sum_{i=1}^{j} p(l_i)$. In [5], we provide an algorithm that selects among each class j a transmission strategy that minimizes the expected overhead.

III. Analytical results

In this section, we derive expressions for the expected used bandwidth and expected outage rate for two transmission schemes. The first one, which we call *Algorithm*, uses the transmission strategy described in the Section II-B. The second one, which we call *Static*, follows a standard approach. It keeps on sending the encoded symbols at a fixed transmission rate until an acknowledgment is received. The transmission rate is fixed to $\mathbf{R}_j = \mathbf{C}_j/(T - FTT)$, where $\mathbf{C}_j = k \times (1+\epsilon)/(1-l_j)$ for $j \in \{1, \cdots, N\}$.

These analytical results extend those of our previous work [5], which considered only the expected overhead.

We assume again that the channel is described by N packet loss rates $l_1 < \ldots < l_N$ with probabilities $p(l_1), \ldots, p(l_N)$ (Figure 2). We also assume a hypothetical rateless code with parameter ϵ.

For $j \in \{1, 2, \cdots, N\}$, the expected bandwidth used by *Algorithm* is

$$\begin{aligned} B_j = E_j + k \times (1+\epsilon) \times \sum_{i=1}^{j-1} p(l_i)/(1-l_i) \\ + k \times (1+\epsilon)/(1-l_j) \times \sum_{i=j}^{N} p(l_i) \end{aligned} \quad (5)$$

where E_j is the associated expected overhead as obtained in [5].

On the other hand, the expected bandwidth used by *Static* is

$$\begin{aligned} B_j = \mathbf{R}_j \times \sum_{i=1}^{j-1} p(l_i) \times \min((T - FTT), (\mathbf{C}_i/\mathbf{R}_j + RTT)) \\ + \mathbf{C}_j \times \sum_{i=j}^{N} p(l_i) \end{aligned} \quad (6)$$

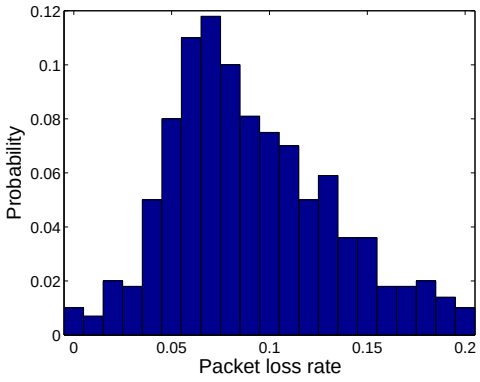

Fig. 4. Packet loss rate histogram.

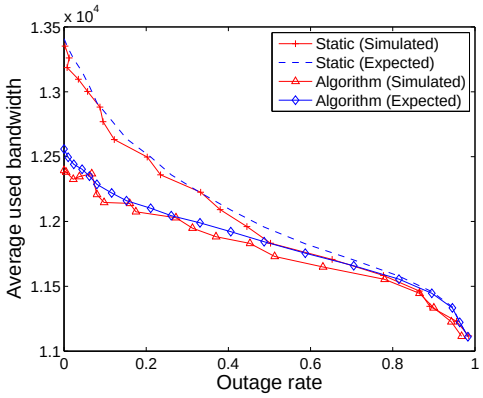

Fig. 5. Used bandwidth vs. outage rate for *Algorithm* and *Static*. For the simulations, a real LT code was used.

For both *Algorithm* and *Static*, the expected outage rate is $1 - \sum_{i=1}^{j} p(l_i)$.

To validate these analytical results, we run simulations with ML designer [10] using the following channel model. The packet loss histogram consisted of $N = 21$ packet loss rates (Figure 4). Both FTT and BTT were set to 0.05 seconds. There were no packet losses in the feedback channel. The maximum transmission rate $R_{\max}$ was equal to $20,000$ symbols/second. Each source block consisted of $k = 10000$ source symbols and was transmitted 500 times. The length of the transmission interval T was set to 1 s. A real LT code was used in the simulations. The parameter ϵ was set to 0.1.

Figure 5 shows that the expected performance closely matches the performance obtained with simulations using a real LT code. It also confirms the superiority of the proposed transmission scheme over the standard approach.

IV. Experimental results

The results provided in [5] and in the previous section were for general data. We now test the performance of our system for H.264 compressed video data.

We used a server in Konstanz to send ICMP [11] packets of size 200 symbols (one byte per symbol) to a machine in Beijing, which sent back the packets to the server in Konstanz. This set-up allowed us to

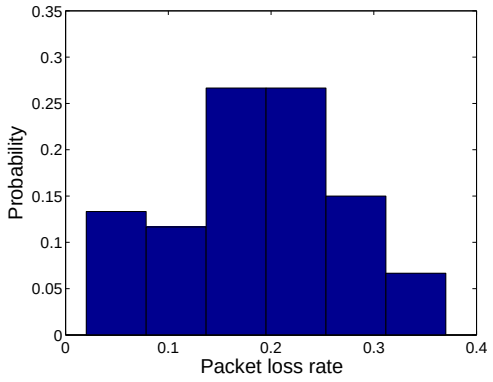

Fig. 6. Packet loss rate histogram for the link Konstanz-Beijing-Konstanz.

measure the channel characteristics without requiring any deployment on the machine in Beijing. The maximum available channel bandwidth $R_{\max}$ was estimated as the maximum sending rate at which the average forward trip time ($\overline{FTT}$) did not increase. This gave $R_{\max} = 30000$ symbols per second and $\overline{FTT} = 270$ ms. More sophisticated techniques can also be used (see [12] for an overview and [13] for a very fast method).

To measure the packet loss rates, we set T to 2 s and sent ICMP packets of size 200 symbols from time $t = 0$ to $t = T - \overline{FTT}$. The sending rate was set according to the estimated value of $R_{\max}$. The packet loss rates were obtained by observing the number of transmitted packets that were received by the client before $t = T$. We repeated the same procedure continuously for a period of 40 mn. Figure 6 shows the resulting packet loss rate histogram. Alternatively, one can start with an arbitrary packet loss rate histogram and update it progressively.

We continued the measurement for another 40 mn and recorded the FTT for all packets. If a packet was not received, its FTT was set to a negative value, indicating that it was lost. We also collected the trace of the backward trip time BTT by sending ICMP packets at a much lower rate because the feedback traffic from the client to the server is very small. The average observed value of BTT, which we denote by $\overline{BTT}$, was 256 ms. By using the collected traces of FTT and BTT, we were able to simulate a real channel.

The test sequence was the 400 frames Foreman sequence in QCIF format of 16 s playback length at a rate of 25 frames per second. The sequence was partitioned into eight source blocks such that each source block had 50 frames to be played back in 2 s. We encoded each source block separately at 64 kbps using the Nokia H.264 encoder. Each group of pictures (GOP) consisted of one I frame followed by 49 P frames.

The encoded symbols were generated on the fly from the source symbols using an LT code. We sent 200 encoded symbols per UDP/IP packet. The first two bytes of the UDP packet payload corresponded to the source block number. The next four bytes indicated the sequence number of the first encoded symbol contained in this packet. Knowing the sequence number of the first encoded symbol is enough to determine the sequence number of all encoded symbols in the same packet.

We transmitted the whole sequence of 400 frames 200 times. This corresponds to the transmission of a 53-minute movie clip. At the client side, freeze-frame error concealment was used to recover missing frames. Fig. 7 shows the average peak signal to noise ratio (PSNR) as a function of the average used bit rate for the following schemes.

1) *Algorithm*. This is the transmission strategy introduced in [5] and described in Section II. The algorithm was run with $\epsilon = 0.1$. Each point on the curve corresponds to the output of the algorithm for a different $j \in \{1, \cdots, 6\}$. We provide results for the packet loss rate histogram of Fig. 6 (*Algorithm-1*) and for an adaptive histogram initialized with a packet loss rate of 0 with probability 1 (*Algorithm-2*) and updated during the transmission.
2) *Static*. This scheme sends packets with a fixed transmission rate until an acknowledgment is received. The transmission rate is fixed to $\mathbf{R}_j = \mathbf{C}_j/(T - \overline{FTT})$, where $\mathbf{C}_j = k \times (1+\epsilon)/(1-l_j)$. Each point on the curve corresponds to a different $j \in \{1, \cdots, 6\}$. Here $\epsilon = 0.1$.
3) *Adaptive*. This scheme keeps on sending the packets at a fixed transmission rate until an acknowledgment is received. However, the code rate is chosen according to the packet loss rate measured during the last transmission interval. For example, if the packet loss rate observed during the transmission of source block b is l_i, then the transmission rate for source block $b+1$ is $\mathbf{R}_i = \mathbf{C}_i/(T - \overline{FTT})$, where $\mathbf{C}_i = k \times (1+\epsilon)/(1-l_i)$. Again $\epsilon = 0.1$.
4) *Without FEC*. This scheme does not use FEC. The used bitrate was increased by increasing the source rate.

Both *Algorithm-1* and *Algorithm-2* significantly outperformed the standard schemes. For example, at 89.86 kilobits per second (kbs), *Algorithm-1* provided an average PSNR of 32.02 dB, while *Static* reached an average PSNR of 28.36 dB at 90.04 kbs. *Algorithm-2* had a slightly worse performance than *Algorithm-1* because it did not exploit prior information about the channel.

Adaptive performed worse than *Static*. This is because the packet loss rate was rapidly changing, making it hard to predict from the packet loss rate observed during the transmission of the previous source block.

The very poor performance of the scheme that does not use FEC is mainly due to the fact that H.264 is highly syntax oriented, and any loss of syntax or control information seriously damages the reconstruction of the bitstream.

V. Conclusion

The goal of the paper was to address the limitations of the system introduced in [5] and to show its practical

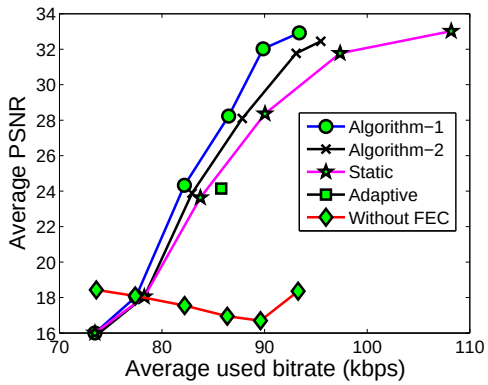

Fig. 7. Average PSNR vs. average used bandwidth for streaming the H.264 encoded Foreman sequence over the link Konstanz-Beijing-Konstanz. *Algorithm-1* uses the algorithm of [5] to compute the transmission strategy. *Algorithm-2* uses the same approach, but the packet loss rate histogram is computed in real-time. *Static* uses a static fixed transmission rate. *Adaptive* uses a fixed transmission rate that is updated according to the packet loss rate observed during the transmission of the previous source block. *Without FEC* does not use FEC.

relevance for streaming video data. Instead of hypothetical rateless codes, we used a real LT code. Instead of expected overhead and expected analytical outage rates (expected rate of source blocks that cannot be recovered), we provided average bandwidth usage and average PSNR values obtained by decoding H.264 compressed video data after real transmissions.

In [5], the packet loss rate histogram used to optimize the transmission strategy was determined over a long period of time before the transmission of the requested data starts. This implies that the client is known to the server in advance. We removed this constraint by showing that our approach is also successful when there is no a priori knowledge about the channel packet loss rates. The idea is to start with an arbitrary histogram that is updated in real-time.

Finally, we compared our system not only to a scheme based on static fixed-rate forward error correction, but also to an adaptive scheme that updates the code rate of the channel code based on observed loss rates. The results showed that our method can significantly outperform both approaches for a wide range of bit rates.

ACKNOWLEDGEMENT

Shakeel Ahmad was supported by the DFG Research Training Group GK-1042.

REFERENCES

[1] U. Horn, K. Stuhlmuller, M. Link, and B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," Signal Processing: Image Commun., vol. 15, pp. 77–94, 1999.

[2] R. Puri, K.-W. Lee, K. Ramchandran, and V. Bharghavan, "An integrated source transcoding and congestion control paradigm for video streaming in the Internet," IEEE Trans. Multimedia, vol. 3, pp. 18–32, March 2001.

[3] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the Internet: Approaches and directions", IEEE Trans. Circuits Syst. Video Technol. vol. 11, pp. 282–300, March 2001.

[4] M. Hayasaka, L. Loyola, and T. Miki, "Packet/cell loss recovery using variable FEC matrix for real time transport services over best effort networks," in Proc. 9th Asia-Pacific Conference on Communications, vol. 3, pp. 1119–1123, Sept. 2003.

[5] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Robust live unicast video streaming with rateless codes", in Proc. 16th Int. Packet Video Workshop, Lausanne, Nov. 2007.

[6] M. Luby, "LT codes", in Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.

[7] A. Shokrollahi, "Raptor codes", IEEE Trans. Inf. Theory, vol. 52, pp. 2551–2567, June 2006.

[8] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard", IEEE Trans. Circuits Syst. Video Technol. vol. 13, pp. 560–576, July 2007.

[9] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools," IEEE Network, vol. 17, pp. 27–35, Nov.-Dec. 2003.

[10] G. Schorcht et al., "System-level simulation modeling with MLDesigner", 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003.

[11] J. Postel, "Internet Control Message Protocol", Request for Comments, Internet Engineering Task Force, no. 792, Sep. 1981.

[12] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools", in Proc. 3rd ACM SIGCOMM conference on Internet measurement, pp. 39–44, 2003.

[13] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques", IEEE Journal on Selected Areas in Communications, vol. 21, pp. 879–894, Aug. 2003.