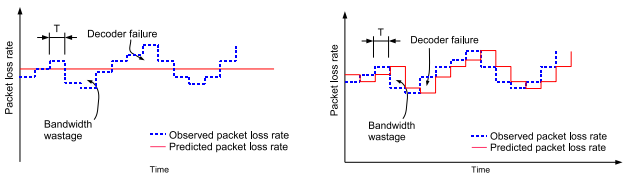
Adaptive Unicast Video Streaming with Rateless Codes and Feedback

Shakeel Ahmad, Raouf Hamzaoui, Marwan Al-Akaidi

***Abstract*— Video streaming over the Internet and packet-based wireless networks is sensitive to packet loss, which can severely damage the quality of the received video. To protect the transmitted video data against packet loss, application-layer forward error correction (FEC) is commonly used. Typically, for a given source block, the channel code rate is fixed in advance according to an estimation of the packet loss rate. However, since network conditions are difficult to predict, determining the right amount of redundancy introduced by the channel encoder is not obvious. To address this problem, we consider a general framework where the sender applies rateless erasure coding to every source block and keeps on transmitting the encoded symbols until it receives an acknowledgment from the receiver indicating that the block was decoded successfully. Within this framework, we design transmission strategies that aim at minimizing the expected bandwidth usage while ensuring successful decoding subject to an upper bound on the packet loss rate. In real simulations over the Internet, our solution outperformed standard FEC and hybrid ARQ approaches. For the QCIF Foreman sequence compressed with the H.264 video coder, the gain in average peak signal to noise ratio over the best previous scheme exceeded 3.5 decibels at 90 kilobits per second.**

I. Introduction

Video streaming over best-effort packet networks requires error resilience mechanisms against packet loss [1]. One of the most powerful such mechanisms is application-layer forward error correction (FEC) where redundant information is added to the original data to make it more robust against transmission errors. Previous works [2], [3], [4], [5], [6] use FEC, where for a given source block the channel code rate is fixed or updated (for example, by puncturing or shortening a Reed-Solomon code) according to a prediction based on past observations of the packet loss rate. Unfortunately, the packet loss rate in many networks, including the Internet and wireless networks, is hard to predict and can rapidly change over time. Thus, the performance of such FEC schemes may be poor because of the unavoidable mismatch between the actual packet loss rate and the predicted one. Indeed, overestimating the packet loss rate would waste the bandwidth and underestimating it would result in decoding failure (Fig. 1).



(a) (b)

Fig. 1. Mismatch between observed and predicted packet loss rate. The observed packet loss rate corresponds to time intervals of length T. (a) Static prediction. (b) Adaptive prediction.

To address this problem, we propose to use a rateless code [7], [8] and receiver feedback. With a rateless code, the code rate does not have to be fixed a priori as the encoder can generate on the fly a potentially infinite stream of encoded symbols. The sender keeps on sending the encoded symbols corresponding to a source block until an acknowledgment is received from the receiver or the transmission time for the source block elapses. While this approach is not new, no previous work has attempted to optimize it. The problem is that if the transmission rate is too low, the receiver may not be able to successfully decode the source block while if it is too high, the sender may transmit too many redundant symbols before it receives the acknowledgment.

Our main contribution is to design for this frame-

 This work has been presented in part at the 16th International Packet Video Workshop, Lausanne, Nov. 2007 and at The Third International Conference on Pervasive Computing and Applications, Alexandria, Oct. 2008. The authors are with the Faculty of Technology, De Montfort University, The Gateway, Leicester LE1 9BH, UK. Email: sahmad@dmu.ac.uk, rhamzaoui@dmu.ac.uk, mma@dmu.ac.uk. Tel: (Shakeel Ahmad): 44 116 207 8973, (Raouf Hamzaoui): 44 116 207 8096. (Marwan Al-Akaidi): 44 116 257 7087.

work transmission strategies that aim at minimizing this overhead (or equivalently the used bandwidth) while ensuring successful reconstruction of the video stream subject to an upper bound on the packet loss rate. Our solution typically consists of a sequence of alternating transmission bursts and waiting times. The optimization does not rely on only one packet loss rate but on a histogram of previously observed packet loss rates.

This paper synthesizes and extends preliminary results presented in [9] and [10]. Specifically, we provide an in-depth discussion of related work, give a detailed description of the proposed transmission strategy and the algorithm to optimize it, and significantly expand the experimental work. In particular, we compare the proposed scheme to more advanced streaming mechanisms and study the influence of both the packet loss rate histogram and bursty packet losses on the system performance.

The remainder of the paper is organized as follows. In Section II, we discuss related work. In Section III, we describe our video streaming system. In Section IV, we present appropriate transmission strategies for this system, derive analytical expressions for the expected used bandwidth of these strategies, and propose an efficient algorithm whose aim is to select strategies that minimize the expected used bandwidth while ensuring successful decoding subject to an upper bound on the packet loss rate. In Section IV-B, we derive analytical expressions for the expected used bandwidth of the studied schemes and compare them with simulation results that use an LT code [7]. Section V contains experimental results. We study the efficiency of our algorithm by comparing it with exhaustive search. We also compare the performance of our transmission strategy to three standard schemes for streaming H.264 [11] compressed video over a real Internet connection. The results show that our strategy provides significant gains in rate-distortion performance.

II. Related work

Rateless codes are probabilistic erasure codes which can recover k source symbols from any received $k(1+\epsilon)$ encoded symbols with high probability. 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. Both the encoding and decoding times of rateless codes are significantly lower than those of traditional fixed-rate erasure codes (e.g., Reed-Solomon codes).

The application of rateless codes to video streaming was initially proposed for Multimedia Broadcast Multicast System (MBMS) standard [12] of the third Generation Partnership Project (3GPP) [13], [14]. In these works, a Raptor code is used as a conventional FEC code with a fixed code rate that can be set according to the transmission conditions, e.g., to meet some expected worst-case receiving conditions.

Vukobratovic *et al.* [15] designed rateless codes with unequal error protection properties to multicast a scalable video stream to a set of receivers with heterogeneous conditions. The goal is to provide increasing quality of service to classes of receivers with increasingly better conditions. The framework assumes that the packet loss rate of each class of receivers is known in advance.

Wagner, Chakareski, and Frossard [16] proposed a framework for streaming video from multiple senders to a single receiver over heterogeneous packet erasure channels. Rateless codes are used to avoid coordination among the servers sending video to the same receiver. For a given total bandwidth, the received video quality is optimized by assigning the best FEC code rate to each server.

Schierl *et al.* [17] used Raptor codes for real-time streaming of scalable video over Mobile Ad Hoc Networks (MANETs). As in [16], they consider a streaming scenario from multiple sources to one receiver. They show how source and relay nodes can allocate an optimized FEC code rate for different clients.

Tan *et al.* [18] considered unicast streaming of stereoscopic video over packet erasure channels. The authors define three layers of source data and apply unequal error protection with rateless codes to these layers. The work assumes that the packet loss rate is Gaussian with known mean and variance.

Compared to this previous work, our scheme differs in several important ways:

- We exploit feedback from the receiver to avoid sending data when the receiver was able to recover the source block.
- We develop a more flexible transmission framework, where transmission strategies consist of a sequence of alternating transmission bursts of variable rates and waiting times. This frame-

work includes transmission strategies that use a fixed transmission rate as a special case.
- We provide an efficient rate scheduling algorithm that aims at optimizing the transmission strategy. The optimization does not rely on only one packet loss rate (for example the average packet loss rate over a time interval), but on a histogram of observed packet loss rates. This allows better adaptation to varying channel conditions.

III. Video Streaming System

Fig. 2 shows the proposed system in a live video streaming application. The raw video stream produced by the camera from time $t = 0$ to $t = T$ is fed into the video 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. At $t = T$, the sender applies FEC to the source block as will be explained in Section IV. The encoded symbols are then transmitted according to the transmission strategy described in Section IV-A. Some of the transmitted encoded symbols are lost or arrive at the receiver too late to be useful. The receiver tries to recover the source block. If it succeeds, it sends an acknowledgment 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 frame of decoded video stream for playback at $t = 2T$, which ensures a maximum end-to-end playback latency of $2T$. Increasing T will increase the size of the source block. This will lead to a more efficient rateless code, but also to a longer playback latency.

The same process is repeated. In this way, source block b corresponds to the video stream captured from $t = (b-1) \times T$ to $t = b \times T,\ b = 1, 2, \ldots$ The source blocks are encoded independently, which can be achieved, e.g., by starting each one with an I frame. Moreover, source block b has to be FEC encoded, transmitted, and FEC decoded from $t = b \times T$ to $t = (b+1) \times T$, so that it is available for playback at $t = (b+1) \times T$.

IV. Proposed FEC Scheme

Consider a source block b consisting of k symbols. We encode the k source symbols by applying a rateless code to produce a potentially infinite

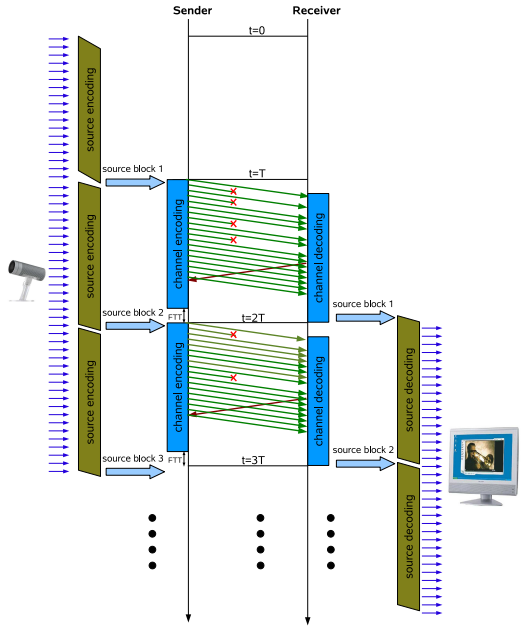


Fig. 2. Proposed video streaming system.

stream of encoded symbols. These encoded symbols are transmitted over the channel after encapsulating them in channel packets. A channel packet may contain one or more encoded symbols. For simplicity, we describe our system when a channel packet contains only one encoded symbol. The maximum transmission rate allowed by our system is denoted by $R_{\max}$. We assume that $R_{\max}$ is large enough to transmit $k(1+\epsilon)/(1-l)$ encoded symbols in a time period of length $T - FTT$. Here FTT is the forward trip time and l is the packet loss rate observed during a time interval of length T. Some of the channel packets are lost or arrive at the receiver too late to be useful. We assume that the receiver can recover source block b correctly if and only if at least $k(1+\epsilon)$ encoded symbols for this block are received before time $(b+1) \times T$. Thus, a simple transmission strategy π to guarantee successful decoding of source block b is to send at least $C = k(1+\epsilon)/(1-l)$ encoded symbols from $t = b \times T$ to $t = (b+1) \times T - FTT$. Unfortunately, the transmitter does not know beforehand the value of C because l is unpredictable and varies from block to block. Overestimating l would result in bandwidth wastage and underestimating it would lead to decoding failure. However, when a feedback channel is available, this problem can be alleviated by making the receiver send an acknowledgment to the transmitter as soon as enough encoded symbols

are received. Since the acknowledgment needs time to reach the transmitter, this approach introduces an *overhead* equal to the number of unnecessary encoded symbols sent to the receiver.

The transmission strategy π is also characterized by an *outage rate* equal to 0 if the source block is successfully decoded and 1, otherwise. Note that a source block can be decoded successfully if and only if $l \leq L(\pi)$, where $L(\pi) = 1 - k(1+\epsilon)/c_{\max}(\pi)$, and $c_{\max}(\pi)$ is the maximum number of encoded symbols that can be sent with the transmission strategy in a transmission interval of duration $T - FTT$.

Fig. 3 shows two transmission strategies. In both cases, the transmitter keeps on sending the encoded symbols until an acknowledgment is received. In Fig. 3(a), the transmission rate is fixed and equal to the maximum transmission rate $R_{\max}$. In Fig. 3(b), it is variable and yields a smaller overhead. Thus, the second transmission strategy (π_2) seems to be better than the first one (π_1). However, since $L(\pi_2) < L(\pi_1)$, the probability of successful decoding is greater with π_1 than with π_2.

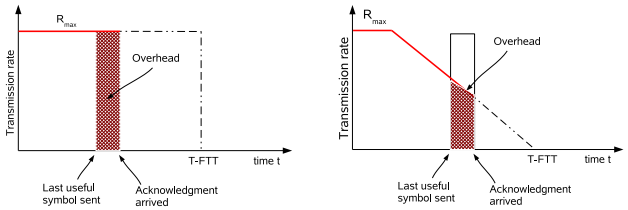


Fig. 3. (a) Transmission strategy with a fixed transmission rate. (b) Transmission strategy with a variable transmission rate. The shaded areas show the overhead.

A. Proposed transmission strategy

Our goal is to construct transmission strategies that minimize the expected overhead subject to a constraint on the expected outage rate. Our transmission strategies are built according to the following principle: make an optimistic guess, send, and wait. If no acknowledgment arrives, this indicates that the guess was wrong, so make the next optimistic guess (excluding the previous one), and repeat the procedure.

Such transmission strategies depend on the packet loss rate distribution. To simplify the problem, we approximate the probability density function of the packet loss rate by a packet loss rate histogram characterized by N packet loss rates $l_1 < \ldots < l_N$ with probabilities $p(l_1), \ldots, p(l_N)$ (Fig. 4).

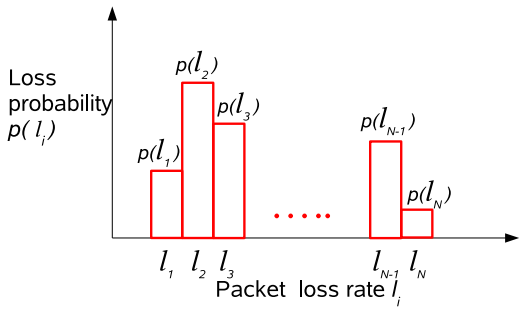


Fig. 4. Probability mass function of observed packet loss rate. A packet is considered to be lost if it is not available at the receiver within a time interval of length T.

Let $j \in \{1, \ldots, N\}$, we first show how to build a class of transmission strategies whose expected outage rate is equal to $1 - \sum_{i=1}^{j} p(l_i)$. In Section IV-C, we provide an algorithm that selects among each class a transmission strategy that minimizes the expected overhead.

We make an optimistic guess by assuming that the packet loss rate l is minimum (equal to l_1) and start transmitting at rate R_1 from $t = s_1 = 0$ to $t = f_1$. Under this assumption, $c_1 = k(1+\epsilon)/(1-l_1)$ is the number of encoded symbols that have to be transmitted to guarantee successful decoding. Thus we select R_1 to satisfy $R_1(f_1 - s_1) = c_1$. If we denote by RTT the round trip time and assume that a reliable feedback channel is available, an acknowledgment 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. An intuitive choice for w_1 would be $w_1 = RTT$. However, this choice may not be the best as it may not leave enough time to transmit the number of encoded symbols required to satisfy the target outage rate.

Similarly, we transmit at rate R_2 from s_2 to f_2 such that $R_2(f_2 - s_2) = c_2 = k(1+\epsilon)/(1-l_2) - c_1$. 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 (Fig. 5) with

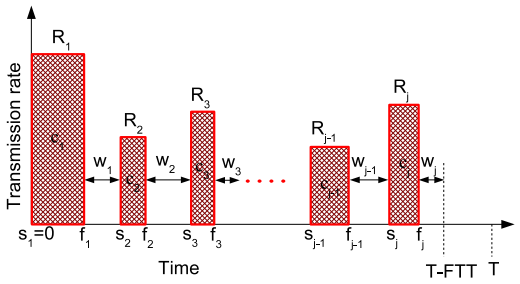


Fig. 5. Proposed transmission strategy. The encoded symbols are transmitted at rate R_i from s_i to f_i, followed by a waiting time of w_i, $i = 1, \ldots, j$.

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

$$R_i(f_i - s_i) = c_i \tag{2}$$

$$s_i = f_{i-1} + w_{i-1} \tag{3}$$

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

$$f_j \leq T - FTT, \tag{4}$$

which states that all encoded symbols are sent within the available time budget.

Note that 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 $1 - \sum_{i=1}^{j} p(l_i)$.

It is easy to see that for each class j, a transmission strategy is completely defined by the transmission rates $R_1, \ldots, R_j$ and the waiting times $w_1, \ldots, w_j$. Fig. 6 shows examples of admissible transmission strategies.

B. Expected used bandwidth and outage rate

We derive analytical expressions for the expected used bandwidth and expected outage rate for two transmission schemes. The first one, which we call *Algorithm*, uses a transmission strategy $\pi = (R_1, \ldots, R_j, w_1, \ldots, w_j)$ in the class of transmission strategies designed for $j \in \{1, \ldots, N\}$. The second one, which we call *Static*, follows a conventional 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 $\mathbf{C}_j = k(1+\epsilon)/(1-l_j)$ for $j \in \{1, \cdots, N\}$.

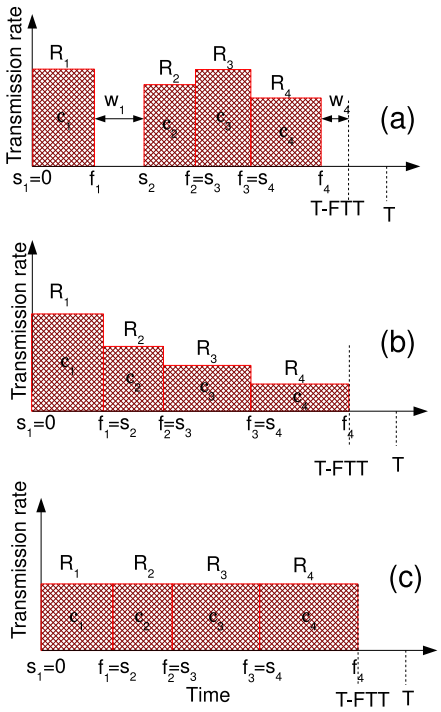


Fig. 6. Three examples of transmission strategies. In (b) and (c), all waiting times are equal to zero.

To find the expected used bandwidth, we first determine the expected overhead. Given a transmission strategy π in the class of transmission strategies designed for j, let $H_i(\pi)$ be the overhead when $l = l_i$ $(1 \leq i \leq j)$. Then the expected overhead for π is

$$E_j(\pi) = \sum_{i=1}^{j} p(l_i) H_i(\pi) \tag{5}$$

When $l = l_i$, the transmission of the smallest number of encoded symbols necessary for successful decoding of a block will be finished at time f_i, and an acknowledgment will be expected at $a_i = f_i + RTT$. For all $i < m \leq j$, the mth transmission burst may contribute to $H_i(\pi)$ if $s_m < a_i$. This contribution is equal to $R_m \Omega_{i,m}$, where

$$\Omega_{i,m} = \max((\min(f_m, a_i) - s_m), 0) \tag{6}$$

is the time for which we transmit at rate R_m before a_i.

Thus $H_i(\pi)$ is the sum of the overhead added by the transmission bursts of indices m ($i < m \leq j$) under $l = l_i$ and can be given as

$$H_i(\pi) = \begin{cases} \sum_{m=i+1}^{j} R_m \Omega_{i,m} & \text{if } i = 1, \ldots, j-1; \\ 0 & \text{if } i = j \end{cases} \tag{7}$$

Combining equations (5) and (7), we can write $E_j(\pi)$ as

$$E_j(\pi) = E_j(R_1, \cdots, R_j, w_1, \cdots, w_j) \tag{8}$$
$$= \sum_{i=1}^{j-1} p(l_i) \sum_{m=i+1}^{j} R_m \Omega_{i,m}. \tag{9}$$

On the other hand, if we use *Static* with a code rate corresponding to the maximum loss rate l_j, then the expected overhead is

$$E_j^{STATIC} = \mathbf{R}_j \sum_{i=1}^{j-1} p(l_i) \left[\min\left[(T - FTT), (\mathbf{C}_i/\mathbf{R}_j + RTT)\right] - \mathbf{C}_i/\mathbf{R}_j\right] \tag{10}$$

and the expected outage rate is $1 - \sum_{i=1}^{j} p(l_i)$.

We now derive expressions for the expected used bandwidth. For $j \in \{1, 2, \cdots, N\}$, the expected bandwidth used by *Algorithm* is

$$B_j(\pi) = E_j(\pi) + \sum_{i=1}^{j-1} p(l_i)\mathbf{C}_i + \mathbf{C}_j \sum_{i=j}^{N} p(l_i) \tag{11}$$

where $E_j(\pi)$ is the expected overhead obtained in (5).

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

$$B_j^{STATIC} = E_j^{STATIC} + \sum_{i=1}^{j-1} p(l_i)\mathbf{C}_i + \mathbf{C}_j \sum_{i=j}^{N} p(l_i). \tag{12}$$

C. Proposed algorithm

Given $j \in \{1, \ldots, N\}$ and the associated class of transmission strategies described in Section IV-A, our goal is to find the transmission rates and the waiting times that minimize the expected overhead subject to the expected outage rate $1 - \sum_{i=1}^{j} p(l_i)$.

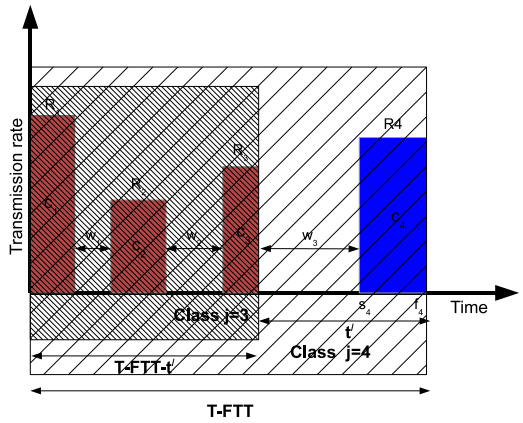


Fig. 7. Graphical description of the proposed algorithm when $j = 4$.

In this section, we provide an algorithm that aims at solving this problem. Our algorithm has linear time complexity. Experimental results show that its performance is close to optimal. Before giving a formal description of the algorithm, we explain it intuitively in the particular case where $j = 4$ (Fig. 7).

Our algorithm is inspired by dynamic programming. The idea is to break the problem into subproblems and compute the solution in a greedy way from $j = 1$ to $j = 4$. We start with $j = 4$, set the time budget to $T - FTT$, w_4 to zero and iterate over all possible values of R_4 and w_3 (both the set of transmission rates and the set of waiting times are discretized). In each iteration, the expected overhead is calculated by using the chosen values of R_4 and w_3 and a set of pre-calculated values for R_3, R_2, R_1, w_2, and w_1. These pre-calculated values are the ones selected by the algorithm for class $j = 3$ when the time budget is $T - FTT - t'$ with $t' = w_3 + c_4/R_4$. Note that the pre-calculated values are computed starting with class $j = 1$ for all time budgets between 0 and $T - FTT$. For each class j, the associated values of $R_1, \ldots, R_j$ and $w_1, \ldots, w_{j-1}$ are used to find the solution for the next higher class $j + 1$. The values of $R_1, \ldots, R_4$ and $w_1, \ldots, w_4$ associated to the iteration that gives the smallest expected overhead is marked by the algorithm. We repeat the same procedure by gradually increasing the value of w_4. The solution returned by the algorithm is the one with the smallest expected overhead.

Formally, for $j \in \{1, \ldots, N\}$ and $t \in [0, T - FTT]$, let us denote by $E_j^*(t)$ the smallest expected

overhead achievable within the time budget $[0, t]$ and providing an expected outage rate $1 - \sum_{i=1}^{j} p(l_i)$. Let $R^*_{1,j}(t), \ldots, R^*_{j,j}(t)$ be the transmission rates and $w^*_{1,j}(t), \ldots, w^*_{j,j}(t)$ be the waiting times corresponding to $E^*_j(t)$. Then the solution to the problem is given by the transmission rates $R^*_{1,j}(T - FTT), \ldots, R^*_{j,j}(T - FTT)$ and the waiting times $w^*_{1,j}(T - FTT), \ldots, w^*_{j,j}(T - FTT)$. We propose to compute these values in a greedy (nonoptimal) way, stage by stage from $i = 1$ to $i = j$. This is done as follows.

Let $\widehat{E}_i(t), \widehat{R}_{i,i}(t), \widehat{w}_{i-1,i}(t) (i = 1, \ldots, j)$ denote the approximations to $E^*_i(t), R^*_{i,i}(t), w^*_{i-1,i}(t)$ computed by our algorithm. Then for $1 \le i \le j$ and $t < \sum_{m=1}^{i} c_m / R_{\max}$, we set $\widehat{E}_i(t) = \infty, \widehat{R}_{i,i}(t) = \infty, \widehat{w}_{i-1,i}(t) = \infty$ to avoid selecting $\widehat{R}_{i,i}(t)$ and $\widehat{w}_{i-1,i}(t)$ at any stage where t is not large enough to transmit $c_1 + \cdots + c_i$ even at the highest possible transmission rate.

For $i = 1$ and $t \ge c_1 / R_{\max}$, we set $\widehat{E}_1(t) = 0, \widehat{R}_{1,1}(t) = c_1/t, \widehat{w}_{0,1}(t) = 0$. For $1 < i \le j$ and $t \ge \sum_{m=1}^{i} c_m / R_{\max}$, we set

$$(\widehat{R}_{i,i}(t), \widehat{w}_{i-1,i}(t)) = \arg \min_{\substack{0 < R_i \le R_{\max} \\ 0 \le w_{i-1} \le RTT}} R_i \sum_{m=1}^{i-1} p(l_m)\Omega_{m,i} + \widehat{E}_{i-1}(t - \frac{c_i}{R_i} - w_{i-1}) \tag{13}$$

$$\widehat{E}_i(t) = \min_{\substack{0 < R_i \le R_{\max} \\ 0 \le w_{i-1} \le RTT}} R_i \sum_{m=1}^{i-1} p(l_m)\Omega_{m,i} + \widehat{E}_{i-1}(t - \frac{c_i}{R_i} - w_{i-1}) \tag{14}$$

We see (13) and (14) as discrete optimization problems. For R_i, this is a natural choice as noninteger transmission rates are not admissible. More generally, when a packet contains more than one symbol, R_i is constrained to take on values in $M, 2M, \ldots, \lfloor R_{\max}/M \rfloor$, where M is the number of symbols per packet. For the waiting time w_{i-1}, discretization is obtained by using uniform quantization with a small step size.

Given j, $R_{\max}$, $l_i, p(l_i)$, $i = 1, \ldots, j$, k, ϵ, T, FTT, and RTT, the algorithm first determines $c_1, \ldots, c_j$ using (1). Then $\widehat{E}_i(t)$, $\widehat{R}_{i,i}(t)$, and $\widehat{w}_{i-1,i}(t)$ are computed using the above equations for $i = 1, \ldots, j$ and $t = 0, \ldots, T - FTT$. In the next step, the algorithm tries to find another transmission strategy that gives the same expected overhead and expected outage rate, but which uses less time to complete the transmission (i.e., it has a smaller f_j or, equivalently, a greater w_j). If one such solution can be found, it is selected because it gives the receiver more time for decoding.

A pseudo-code of the algorithm is given in Algorithm 1. The algorithm has $j \times RTT \times Q \times (T - FTT) \times Q \times R_{\max}$ time complexity and $3 \times j \times (T - FTT) \times Q$ space complexity, where Q is the number of steps per second used to quantize time. In contrast, exhaustive search has $(RTT \times R_{\max} \times Q \times Q)^j$ time complexity.

Algorithm 1

Input: j, $R_{\max}$, $l_i, p(l_i)$, $i = 1, \ldots, j$, T, FTT, RTT, k, ϵ
Output: Optimized transmission strategy $(R_1, \ldots, R_j, w_1, \ldots, w_j)$, expected overhead E_j, f_i, $i = 1, \ldots, j$
for $i = 1$ to j **do**
  for $t = 0$ to $T - FTT$ **do**
    Compute $\widehat{E}_i(t)$, $\widehat{R}_{i,i}(t)$ and $\widehat{w}_{i-1,i}(t)$
  end for
end for
Set $f_j = T - FTT$ and $E_j = \infty$
for $t = 0$ to $T - FTT$ **do**
  if $\widehat{E}_j(T - FTT - t) \le E_j$ **then**
    $E_j = \widehat{E}_j(T - FTT - t)$ and $f_j = T - FTT - t$
  else
    break
  end if
end for
$w_j = T - FTT - f_j$
for $i = j$ to 1 **do**
  $R_i = \widehat{R}_{i,i}(f_i)$ and $w_{i-1} = \widehat{w}_{i-1,i}(f_i)$
  $f_{i-1} = f_i - c_i/R_i - w_{i-1}$
end for

V. Experimental results

A. Efficiency of the proposed algorithm

Our algorithm tries to minimize the expected overhead for a given expected outage rate. To study the efficiency of this algorithm, we compared it with exhaustive search. The comparison was done

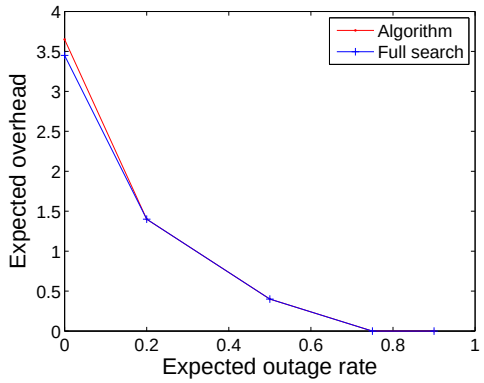


Fig. 8. Comparison of the proposed algorithm with exhaustive search.

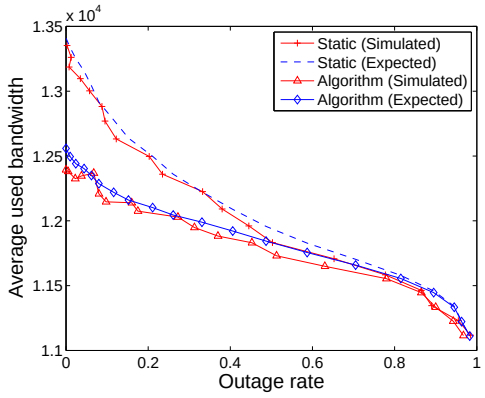


Fig. 9. Used bandwidth vs. outage rate for *Algorithm* and *Static*. For the simulations, a real LT code was used.

for small instances of the problem since, otherwise, exhaustive search would not be feasible. Fig. 8 shows results for $k = 130$, $T = 1$ s, $R_{\max} = 200$ symbols/s, $\epsilon = 0.05$, $FTT = 0.06$ s, $RTT = 0.12$ s, and the packet loss rates 0.03, 0.06, 0.09, 0.12, 0.15 with probabilities 0.1, 0.15, 0.25, 0.3, 0.2, respectively. In most cases, our algorithm produced an almost optimal solution. The negligible quality loss was compensated for by a significant speed up. For example, on a PC running an Intel Core 2 Duo 2 GHz processor and 2 GB RAM, exhaustive search needed 34 mn to compute the solution corresponding to $j = 5$, while our algorithm took only 126 ms.

The expected overhead (or equivalently used bandwidth) used by our algorithm as an objective function assumes a hypothetical rateless code with parameter ϵ (see Section IV-B). In practice, however, a real rateless code must be used. In the next experiment, we compared the expected used bandwidth derived analytically with the real used bandwidth obtained by simulations with a real rateless code. The simulations were done with MLDesigner [19]. We used $N = 21$ packet loss rates 0, 0.01, 0.02, 0.03, ..., 0.19, 0.2 with probabilities 0.010, 0.007, 0.020, 0.018, 0.050, 0.080, 0.110, 0.118, 0.100, 0.081, 0.075, 0.070, 0.050, 0.059, 0.036, 0.036, 0.018, 0.018, 0.020, 0.014, 0.010, respectively. Both FTT and BTT were set to 0.05 s. There were no packet losses in the feedback channel. The maximum transmission rate $R_{\max}$ was equal to $20,000$ symbols/s. One symbol consisted of one byte. Each source block consisted of $k = 10,000$ source symbols and was transmitted 500 times. The length of the time interval was $T = 1$ s. A real LT code was used in the simulations. The parameter ϵ used in the analytical expressions to optimize the transmission strategy was set to 0.1.

Fig. 9 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 scheme over a conventional static approach (see Section IV-B).

We also tested the performance of our system for the $N = 11$ packet loss rates 0, 0.02, 0.04, 0.06, ..., 0.2 with probabilities 0.017, 0.038, 0.130, 0.218, 0.181, 0.145, 0.109, 0.072, 0.036, 0.034, 0.02, respectively. For all source blocks, the number of symbols was $k = 10,000$, the length of the time interval was $T = 1$ s, and the maximum transmission rate was $R_{\max} = 20,000$ symbols/s. A hypothetical rateless code with $\epsilon = 0.05$ was assumed. Fig. 10 shows the results. Each curve corresponds to the $N = 11$ expected outage rates $1 - \sum_{i=1}^{j} p(l_i)$ $(j = 1, \ldots, 11)$. The expected overhead was computed according to (8). We used $Q = 1000$ time steps per second to quantize time. The gain increased with decreasing round trip time because short trip times allow the sender to quickly know the status of the receiver and stop transmitting redundant encoded symbols.

Fig. 11 shows the transmission strategy that our algorithm selected to ensure zero expected outage rate when $RTT = 0.2$ s. Note how the first transmission rate was the highest one, which is a reasonable choice since the first c_1 symbols have to be sent in the shortest time to minimize overhead.

The maximum transmission rate $R_{\max}$ should be

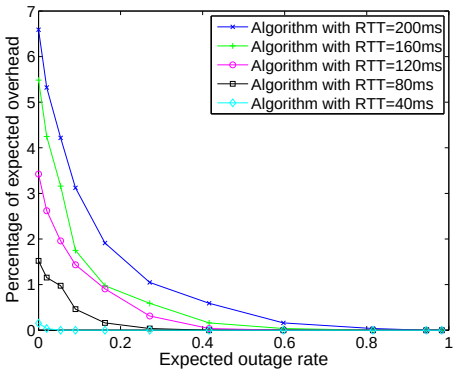


Fig. 10. Performance for various round trip times. For each curve $FTT = RTT/2$.

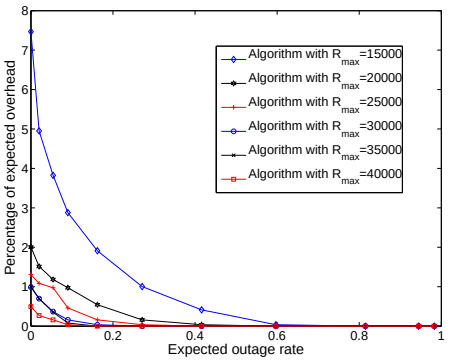


Fig. 12. Performance for various maximum transmission rates.

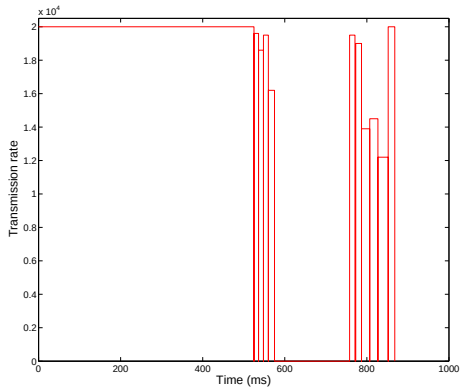


Fig. 11. Optimized transmission strategy showing transmission rates $R_1, \ldots, R_{11}$ and waiting times $w_1, \ldots, w_{11}$ as computed by Algorithm 1. Apart from w_5, all other waiting times are negligible.

large enough to transmit the number of encoded symbols corresponding to the worst-case packet loss rate. Otherwise, one has to decrease the source rate accordingly. Fig. 12 shows the expected overhead when $R_{\max}$ is decreased. The results are given for the $N = 11$ packet loss rates used in the experiment of Fig. 10, $RTT = 0.1$ s, $FTT = 0.05$ s, $k = 10000$, $\epsilon = 0.05$, and $T = 1$ s. The performance of our system improves with increasing $R_{\max}$ because an increase in $R_{\max}$ permits a decrease in the duration of the ith transmission burst ($i = 1, \ldots, j$), leaving more margin for w_i.

B. Streaming H.264 video over the Internet

The results provided in Sections V-A were for general data and an artificial channel. We now test the performance of our system for H.264 compressed video data and a real Internet link.

We used a server in Konstanz to send ICMP [20] 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 measure the channel characteristics without requiring any deployment on the machine in Beijing. The maximum transmission rate $R_{\max}$ was chosen as the average available bandwidth which was estimated as the maximum sending rate at which the average observed forward trip time ($\overline{FTT}$) did not increase. This gave $R_{\max} = 30,000$ symbols per second (i.e., 240 kbps) and $\overline{FTT} = 270$ ms. More sophisticated techniques for the estimation of the available bandwidth can also be used (see [21] for an overview and [22] 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 $R_{\max}$. The packet loss rate was 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. Fig. 13 shows the resulting packet loss rate histogram. All packet loss rates in a histogram bin were quantized to the bin center.

We continued the measurement for another 40 mn and recorded the FTT for all packets. If a packet was not received within its corresponding time interval of length T, its FTT was set to a negative value, indicating that it was lost. Fig. 14 shows the variation of the packet loss rate for the first 60 time intervals.

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

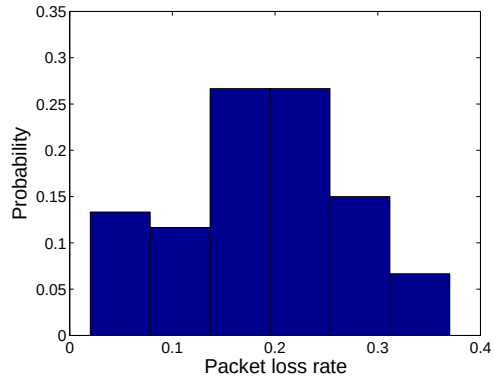


Fig. 13. Packet loss rate histogram for the link Konstanz-Beijing-Konstanz.

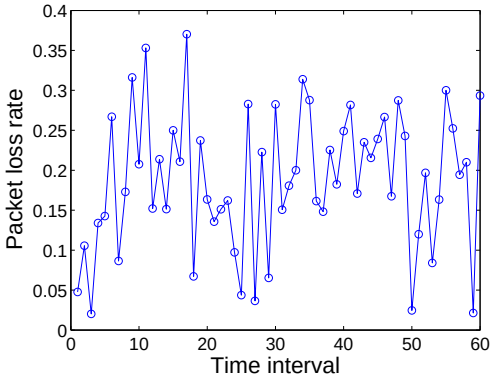


Fig. 14. Packet loss rate as a function of the time interval for the link Konstanz-Beijing-Konstanz.

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.

We considered two standard video test sequences in QCIF format: Foreman (400 frames) and carphone (300 frames). The sequences were partitioned into 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 two bytes gave the number k of symbols in the source block. 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.

Each sequence was transmitted 200 times. At the client side, freeze-frame error concealment where a missing frame is replaced with the last decoded frame was used. Fig. 15 and 16 show 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 described in Section IV-C. The algorithm was run with $\epsilon = 0.1$. The quantization step size was 50 ms for time and 200 bytes/s for the transmission rate (that is, $Q = 1000$ and $M = 200$, see Section IV-C), giving a running time of less than 100 ms on our PC with Intel Core 2 Duo 2 GHz processor. We provide results for the packet loss rate histogram of Fig. 13 (*Algorithm-1*) and for an adaptive histogram (*Algorithm-2*). Each point on the curves corresponds to the output of the algorithm for a different $j \in \{1, \cdots, 6\}$. The histogram of (*Algorithm-2*) is initialized with a packet loss rate of 0 with probability 1. At the end of each period of length T, the receiver informs the sender about the number of packets received. The sender uses this feedback to update the histogram. Note that this information is negligible compared to the video bit rate.
2) *Static*. This scheme keeps on sending the LT encoded symbols at a fixed transmission rate until an acknowledgment is received. The transmission rate is fixed to $\mathbf{R}_j = \mathbf{C}_j/(T - \overline{FTT})$, where $\mathbf{C}_j = k(1+\epsilon)/(1-l_j)$. Each point on the curve corresponds to a different $j \in \{1, \cdots, 6\}$. Here $\epsilon = 0.1$.
3) *Adaptive*. This scheme also keeps on sending the LT encoded symbols with a fixed transmission rate until an acknowledgment is received. However, the transmission 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 $\mathbf{C}_i = k(1+\epsilon)/(1-l_i)$. Here also $\epsilon = 0.1$.
4) *Hybrid ARQ*. This scheme is a type 2 hybrid-ARQ scheme. The sender sets the amount

of LT redundancy according to the last observed packet loss rate l_i and sends the LT encoded symbols at a transmission rate equal to $R_{\max}$. The receiver keeps on trying to decode the received symbols and sends an acknowledgment to stop the transmission of further encoded symbols if LT decoding is successful. If after time $\frac{k(1+\epsilon)}{(1-l_i)R_{\max}} + \overline{FTT}$, LT decoding is not successful, the receiver sends a negative acknowledgment (NACK) asking for additional encoded symbols. The NACK packet gives the sender the number of encoded symbols received. The sender infers the current packet loss rate and sends the number of encoded symbols needed to decode the block successfully. This NACK packet is sent only if the extra encoded symbols are expected to be received on time. Again $\epsilon = 0.1$.

5) *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, for the Foreman sequence at 89.86 kilobits per second (kbps), *Algorithm-1* provided an average PSNR of 32.02 dB at 89.86 kbps, while *Static* reached an average PSNR of 28.36 dB at 90.04 kbps. *Algorithm-2* had a slightly worse performance than *Algorithm-1* because it did not exploit prior information about the channel. Similar results were obtained for the carphone sequence.

Note how *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.

Fig. 17 shows that the performance of our scheme improves when we increase the number of bins in the histogram. However, the performance stagnates after a certain number of bins is reached. The experiment indicates that the histogram should be fine enough to reflect any major variations of the packet loss rate but that a too fine histogram is not necessary as it would increase the time complexity of the algorithm without producing any significant

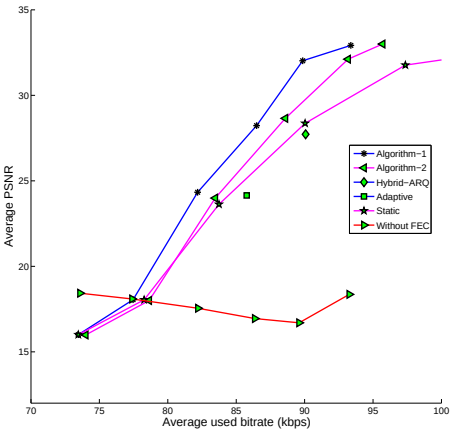


Fig. 15. Average PSNR vs. average used bandwidth for streaming the H.264 encoded Foreman sequence over the link Konstanz-Beijing-Konstanz. *Algorithm-1* uses the algorithm of Section IV-C to compute the transmission strategy. *Algorithm-2* uses the same approach, but the packet loss rate histogram is 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. *Hybrid ARQ* is a type 2 hybrid-ARQ scheme. *Without FEC* does not use FEC.

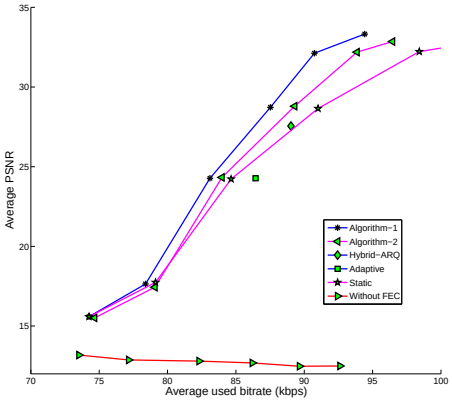


Fig. 16. Average PSNR vs. average used bandwidth for streaming the H.264 encoded carphone sequence over the link Konstanz-Beijing-Konstanz.

performance gain.

To study the effects of bursty packet losses on the performance of our scheme, we did experiments on a two-state Markov channel. Fig. 18 shows that the average used bit rate needed to achieve a given PSNR increases with the average burst error length. This is because a higher burst error length increases the variance of the packet loss rate. When the average burst error length is small, the packet loss rate usually remains close to the mean packet loss

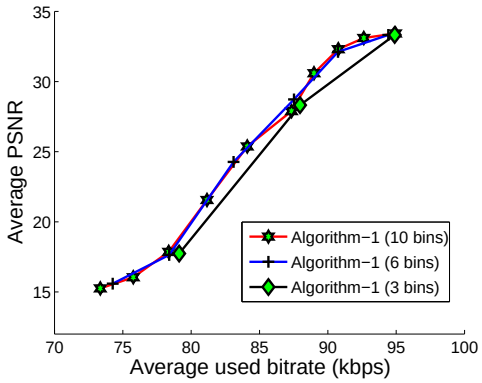


Fig. 17. Average PSNR vs. average used bandwidth for streaming the H.264 encoded carphone sequence over the link Konstanz-Beijing-Konstanz. Each curve shows the performance of *Algorithm-1* for a different number of bins in the packet loss rate histogram. The experimental setup is as in Fig. 16.

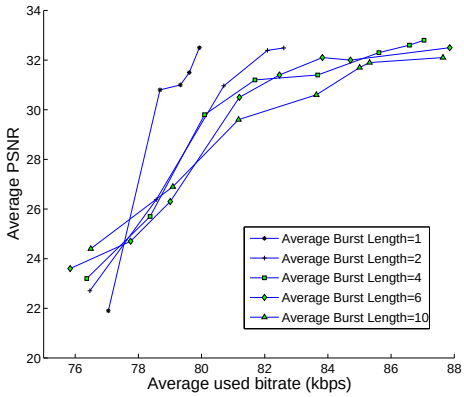


Fig. 18. Average PSNR vs. average used bandwidth for streaming the H.264 encoded carphone sequence over a two-state Markov channel. Each curve shows the performance of *Algorithm-2* for a different average burst length and a fixed mean packet loss rate of 0.1.

rate, whereas at higher values of the average burst error length, the packet loss rate spreads around the mean packet loss rate. This also explains why at the lowest bit rates the average PSNR is slightly higher for the larger average burst lengths.

VI. Conclusion

We proposed a channel-adaptive system for streaming live or pre-recorded video over packet erasure channels. The system copes with the problem of fluctuating and unpredictable packet loss rate by using an efficient transmission strategy, which is optimized according to a packet loss rate histogram. The strategy typically consists of a sequence of alternating transmission bursts and waiting periods. Results for H.264 encoded video sequences, LT codes, and Internet links showed that our system can significantly outperform previous FEC schemes over a wide range of transmission bit rates.

Applications that use a particular channel frequently can benefit most from our approach because they can measure the packet loss rate histogram in advance. If the histogram is not available, one can start with an arbitrary one and update it during transmission. Our experiments showed that while the performance would degrade slightly, it remains better than that of the standard approaches.

A lost or delayed acknowledgment degrades the performance of our system. In the worst case, when the packet loss rate in the feedback channel gets close to 1, the performance of our system would approach that of the static scheme.

The performance of our system may deteriorate if the value of $R_{\max}$ in the algorithm exceeds the available bandwidth. On the other hand, if the value of $R_{\max}$ is much lower than the available bandwidth, the link will be underutilized. To address this problem, one can estimate the available bandwidth periodically and update $R_{\max}$ accordingly. Bandwidth estimation can be done using the observed packet loss rate as in [23], [24] or both the packet loss rate and the average packet inter-arrival time as in [25].

Appendix

In this appendix, we summarize the most frequently used notations of the paper.

- k: number of source symbols.
- ϵ: a small real number that gives the trade-off between the error recovery property of a rateless code and the amount of redundancy it introduces. The channel decoder uses $k(1+\epsilon)$ encoded symbols to decode k source symbols.
- FTT: forward trip time.
- BTT: backward trip time.
- RTT: round trip time.
- T: time in which the receiver accepts the encoded symbols corresponding to a source block.
- l: packet loss rate in a time interval of length T.
- $l_1, \ldots, l_N$: packet loss rates obtained by discretizing the packet loss rate distribution into a histogram of N bins.
- $p(l_i)$: probability of packet loss rate l_i.

- $\pi = (R_1, \ldots, R_j, w_1, \ldots, w_j)$: transmission strategy consisting of j transmission bursts of rates $R_1, \ldots, R_j$ and j waiting times $w_1, \ldots, w_j$.
- $R_{\max}$: maximum transmission rate used by a transmission strategy.
- $c_i = k(1+\epsilon)/(1-l_i) - \sum_{m=0}^{i-1} c_m$ with $c_0 = 0$.
- s_i: starting time of ith transmission burst.
- f_i: ending time of ith transmission burst.
- $a_i = f_i + RTT$.
- $E_j(\pi)$: expected overhead for transmission strategy $\pi = (R_1, \ldots, R_j, w_1, \ldots, w_j)$.
- $B_j(\pi)$: expected used bandwidth for transmission strategy $\pi = (R_1, \ldots, R_j, w_1, \ldots, w_j)$.
- $\mathbf{C}_j = k(1+\epsilon)/(1-l_j)$.
- $\mathbf{R}_j = \mathbf{C}_j/(T - FTT)$.
- M: number of encoded symbols in a packet.
- Q: number of steps per second used to quantize time.
- $\overline{FTT}$: average observed FTT.
- $\overline{BTT}$: average observed BTT.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their comments and suggestions. We also thank Martin Röder for helpful discussions. This work was supported by the DFG Research Training Group GK-1042.

REFERENCES

[1] M. van der Schaar and P. A. Chou (eds.), "Multimedia over IP and Wireless Networks," Academic Press, 2007.

[2] A. Albanese, J. Bloemer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," IEEE Trans. Inf. Theory, vol. 42, pp. 1737–1747, Nov. 1996.

[3] 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.

[4] 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.

[5] 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 Tech., vol. 11, pp. 282–300, March 2001.

[6] 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.

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

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

[9] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Robust live unicast video streaming with rateless codes," in Proc. 16th Int. Packet Video Workshop, Lausanne, pp. 78–84, Nov. 2007.

[10] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Practical channel-adaptive video streaming with fountain codes," in Proc. ICPCA 08, Proc. Third International Conference on Pervasive Computing and Applications, Alexandria, Oct. 2008.

[11] 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.

[12] 3GPP TS 26.346 V6.6.0, "Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs", Oct. 2006.

[13] J. Afzal, T. Stockhammer, T. Gasiba, and W. Xu, "System design options for video broadcasting over wireless networks," in Proc. 3rd IEEE Consumer Communications and Networking Conference, pp. 938- 943, Jan. 2006.

[14] J. Afzal, T. Stockhammer, T. Gasiba , and W. Xu, "Video streaming over MBMS: A system design approach", Journal of Multimedia, vol. 1, no. 5, pp. 25–35, Aug. 2006.

[15] D. Vukobratovic, V. Stankovic, V. Sejdinovic, L. Stankovic, and Z. Xiong, "Expanding Window Fountain codes for scalable video multicast," in Proc. IEEE International Conference on Multimedia and Expo, vol. 1, pp. 77–80, Hannover, April 2008.

[16] J.P. Wagner, J. Chakareski, and P. Frossard, "Streaming of scalable video from multiple servers using rateless codes," in Proc. IEEE International Conference on Multimedia and Expo, Toronto, pp. 1501–1504, July 2006.

[17] T. Schierl, S. Johansen, C. Hellge, T. Stockhammer, and T. Wiegand, "Distributed rate-distortion optimization for rateless coded scalable video in Mobile Ad Hoc Networks," in Proc. IEEE International Conference on Image Processing, vol. 6, pp. 497–500, San Antonio, TX, Sept. 2007.

[18] A. S. Tan, A. Aksay, C. Bilen, G.B. Akar, and E. Arikan, "Rate-distortion optimized layered stereoscopic video streaming with raptor codes," in Proc. 16th International Packet Video Workshop, Lausanne, pp. 98–104, Nov. 2007.

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

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

[21] 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.

[22] 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.

[23] T. Turletti and C. Huitema, "Videoconferencing on the Internet," IEEE Trans. Networking, vol. 4, pp. 340–351, June 1996.

[24] D. Wu, Y. 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 Trans. Circuits Syst. Video Technol., vol. 10, pp. 923–941, Sept. 2000.

[25] Y. J. Chung, J. Kim, and C.-C. J. Kuo, "Real-time streaming video with adaptive bandwidth control and DCT-based error concealment", IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, vol. 46, pp. 951–956, July 1999.