A Content-Aware Rateless Error Protection Scheme for Live Video Streaming Systems

Michael Schier
Institute of Computer Science
University of Innsbruck, Austria
michael.schier@uibk.ac.at

Michael Welzl
Department of Informatics
University of Oslo, Norway
michawe@ifi.uio.no

***Abstract*—Over the past years, several schemes have been proposed to estimate the perceptual distortion in video quality caused by packet loss. Some of them are intuitively designed and do not correlate well with actual distortion values in most real-world scenarios. Others simulate decoders' error concealment measures and use the quality degradation of reconstructed regions as basis for calculating importance estimates. However, such techniques do not take error propagation in the temporal domain into account and are computationally expensive, especially when estimates cannot be derived in parallel to, or as side-product of the video encoding process. We therefore propose a novel approach which targets all previously mentioned problems and inspects dependencies between media units at the level of macroblocks. We show how our scheme can be applied to live video streaming systems using rateless codes for error protection and identify pitfalls which have to be considered. Test results indicate that the proposed unequal error protection scheme considerably outperforms previous approaches, independent of the resolution of the test sequences.**

I. Introduction

When transporting multimedia content over networks, data corruption and data loss due to transmission derogations or congestion may occur, degrading the consumer's quality of experience. As a consequence, it is vital for companies such as IPTV providers to develop and install appropriate countermeasures and to regularly monitor the delivered content's quality. Such countermeasures usually encompass two basic strategies: implementing retransmission requests over a feedback channel or increasing the data stream's robustness by injecting redundancies. In scenarios where bandwidth is not considered as a limiting factor, both mechanisms work well and content-awareness would not be of any benefit.

Unfortunately, this is not always the case. Especially in highly congested networks and wireless/mobile networks where other factors like weather, distance and environment play a crucial role, bandwidth fluctuations interfere with multimedia transmissions, and content-aware solutions are the obvious choice to minimize quality degradation at the consumer side. In this paper, we present a novel approach for protecting video streams. We use Digital Fountain codes (rateless codes) as a basis and build a data prioritization scheme on top of it which inspects certain aspects of the video stream and assigns estimates to elementary data units reflecting their perceptual impact in case of loss. This allows the rateless encoder to perform content-aware encoding operations by adapting the probability of correctly decoding data elements according to their relevance.

In Section III, we explain in detail which elements of the video stream are used as input for the estimation process and how the estimates are calculated. Section IV outlines how rateless codes work in detail and demonstrates how the previously mentioned estimates can be incorporated. In Section V, we describe all components involved in the video streaming process and specify a minimal set of parameters needed to control timely delivery of content. The performance gain in terms of visual quality improvement is evaluated in Section VI, and in Section II, we give a brief survey of related work. Section VII concludes.

II. Related Work

Over the last few years, several strategies have been proposed which try to quantize the relative importance of individual media units within a video stream. Many of them either presume that there exists a clearly defined layered structure between single units in case of layered encoding [1]–[4], or intuitively choose the frame type as sole decision criterion [5]–[8]. Such techniques are computational inexpensive and easy to integrate, but show a lack of accuracy in many situations. Especially in combination with video codecs which have a complex hierarchical structure due to temporal and spatial prediction techniques, a distinction at such a high level is not sufficient. An intuitive solution to estimate the distortion caused by the loss of media units is to simulate it and measure the resulting video quality for each possible loss pattern. Obviously, this is not feasible in practice due to the exponential number of possible derogations. A compromise is to only consider isolated packet loss, as proposed by Baldi et al. [9]–[12]. This technique, which they call analysis-by-synthesis, works well for low packet loss rates, but estimates are getting worse the more packets are affected. To incorporate the co-impact of multiple losses, Korhonen et al. [13] modify the additive distortion model of Baldi et al. and apply weights

978-1-4577-0351-5/11/$26.00

to distortion estimates based on the GOP structure used. However, this still does not take temporal dependencies into account and the propagation of errors caused by misconcealed frame regions across depending frames. This turns out to be especially a problem with scenes having a moderate, unsteady amount of motion where the encoder is still able to identify a sufficient amount of temporal redundancies and consequently decides against the premature insertion of intra-coded segments which would stop the error propagation. By comparison, the distortion estimation scheme proposed in this paper considers both temporal error propagation as well as the amount of motion present, and it can be efficiently calculated without the necessity of modifying the encoder.

With regard to unequal error protection in connection with rateless codes, few related work exists. Rahnavard et al. [14] present a method to realize unequal error protection by grouping input symbols according to their relevance, analyze in detail the probability theory behind the system, and study the special case having two levels of protection. Furthermore, they make the crucial observation that the gradation in terms of the relative importance of input symbols has to be well chosen in order to ensure a steady decoding progress. This issue is directly related with the appropriate selection of the value t_Ψ, as outlined in Section IV. In a further work, the same authors apply these concepts to video transmission, but consider the type of frames as sole distinction criterion for protection levels to media units [8]. Additionally, they measure the obtained outcome only in terms of decodable frame ratio, which is well known in literature to badly correlate with the actual delivered video quality. Similarly, Dimakis et al. [5] propose a system using unequal growth codes and present promising results. Regarding prioritization, they also merely distinguish between intra coded and forward-predictive coded frames, and do not consider more advanced schemes.

III. Perceptual Importance Estimation Process

The estimation process is responsible for mapping priorities to fundamental building blocks of MPEG-like video streams. We already demonstrated the effectiveness of an earlier version of such a mechanism in [15], [16]. In the context of this paper, we now focus on one specific video codec, H.264/MPEG-4 AVC, being the current state of the art of video compression.

Video codecs can target a broad range of compression ratios by adopting a combination of lossy and lossless compression techniques. The main goal is the reduction of temporal and spatial redundancies. The latter is limited by the size of the slice the element belongs to, which implies that redundancies of e.g. macroblocks belonging to the same frame but different slices cannot be exploited in the spatial domain. Furthermore, slices are encapsulated by network abstraction layer (NAL) units which are designed to be used as payload for network packets. The proposed error protection scheme regards such NAL units as input symbols for the encoding process, and a differentiation below this level is therefore not reasonable. As a consequence, our perceptual estimation scheme mainly concentrates on dependencies between NAL units in the temporal domain.

Figure 1. Macroblock partitioning at regions with increased spatial complexity.

The perceptual importance estimation process considers major characteristic properties of the video stream: the types of macroblocks contained in the NAL unit under consideration as well as their partitioning, the amount of motion present in the current frame and the spatial distance of macroblocks they refer to, and the temporal prediction dependency graph. In the remainder of this paper, we refer to the perceptual relevance of a specific region as Ψ, and consider it as the mean over all estimates $\Psi(m_i)$ of macroblocks m_i it contains. To compute Ψ, we combine all aspects mentioned above and assign weights ω_j ($\sum_j \omega_j = 1$) to them (see Equation 1).

$$\Psi(m) = \sum_{j \in \{\text{type, dep, mv}\}} \omega_j \cdot \Psi_j(m) \in [0;1] \qquad (1)$$

The output of $\Psi_{\text{type}}(m)$ primarily depends on the partitioning of macroblock m. m can be either of the form 16x16, 16x8, 8x16 or 8x8. Macroblocks of the form 8x8 can be further subdivided into 8x8, 8x4, 4x8 or 4x4 sub-partitions. The encoder's choice of the appropriate partitioning type depends on the spatial complexity of the content to encode, prevailingly sticking to the following rule of thumb: the higher the complexity, the higher the number of partitions of single macroblocks. As an example, Figure 1 shows the partitioning of (P-)frame 42 of the test sequence COASTGUARD. The regions having a rather inhomogeneous (luma) distribution are located close to the two boats and the

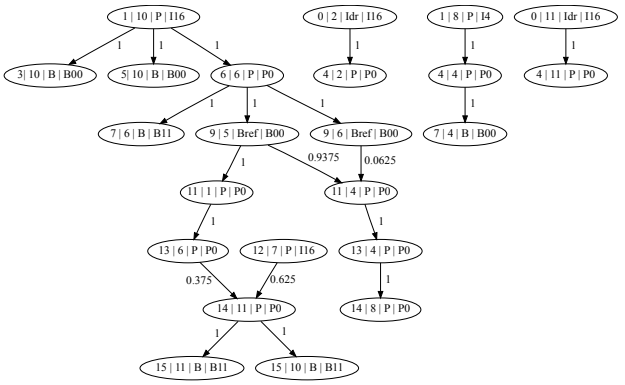

Figure 2. Temporal dependency graph of a 48x32 test pattern sequence, heavily reduced for improved clarity.

waves, which are at the same time the only places where macroblocks with a partitioning of 8x8 and below can be found.

For the determination of Ψ_{type}, we use a lookup-table whose values were derived by an iterative tuning process and refined throughout numerous simulations. It encompasses 22 entries which are disjoint subsets of possible macroblock types as defined in tables 7.11-7.14 and 7.17-7.18 of the H.264 standard [17]. By ignoring certain special cases, it can be said that $\Psi_{\text{type}}(m)$ positively correlates with the number of partitions of macroblock m, which means that e.g. the loss of a macroblock with 10 or more partitions generally causes a higher quality degradation than the loss of a 16x8 macroblock, premising similar conditions with regard to temporal dependencies.

Macroblocks of type `pcm` and `skip` are special cases, which have Ψ_{type} values of 1 and 0, respectively. `pcm` macroblocks are also called *lossless* macroblocks because the prediction, transformation and quantization steps are skipped and the CABAC entropy coder is bypassed. They usually occur in rare situations when the size of a macroblock after compression exceeds 384 bytes (this is the fixed size of uncompressed `pcm` macroblocks), which heavily depends on the current state of the CABAC encoder and the used quantization parameter. Consequently, the likelihood that error concealment mechanisms produce an acceptable interpolation of such complex regions is rather low, which justifies our choice of $\Psi_{\text{type}}(m|\,\text{type}(m) = \texttt{pcm}) = 1$. In contrast, `skip` macroblocks neither possess any motion information nor any residuals and can therefore in most cases be satisfyingly interpolated at decoding time by concealment algorithms.

While inspecting the video stream, our prioritization algorithm constructs a dependency graph at the level of macroblocks and below. The graph's edges are inverted, i.e., as opposed to motion vectors, dependencies point from the area used for reference to the dependent region. A tiny snippet of such a graph is depicted in Figure 2. The

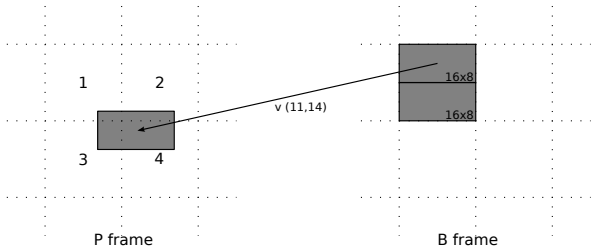

Figure 3. Derivation of α_i for motion vector v.

node labels have to be read as "`frameDecodingTimestamp macroblockIndex frameType macroblockType`", and the edges are labeled as α, which will be explained in the following paragraph. One important property of macroblock dependency graphs is that they are acyclic, which is implied by the fact that decoding timestamps are strictly increasing along an arbitrary path. Furthermore, such paths are limited by the size of the current group of pictures (GOP), presuming that the video sequence has a closed-GOP structure. In the opposite case, an additional threshold would have to be defined which limits the search depth of the algorithm in order to keep the computational cost at a reasonable level.

Based on that, for every macroblock m, we can compute the value $\delta_{\rightsquigarrow}(m)$ reflecting the size of the temporally dependent area which may be distributed over several frames. $\delta_{\rightsquigarrow}(m)$ is recursively defined in Equations 2 and 3.

$$\delta_{\rightsquigarrow}(m) = 1 + \sum_{v \in \text{MV}(m)} \delta'_{\rightsquigarrow}(v) \tag{2}$$

$$\delta'_{\rightsquigarrow}(v) = \sum_{i} \delta_{\rightsquigarrow}(m_i) \cdot \alpha_i(v) \tag{3}$$

In this context, $\text{MV}(m)$ is the set of motion vectors which use parts of macroblock m as reference. Such sets do not necessarily have to be disjoint because motion vectors can point to up to 4 different macroblocks. Consequently, possible domains for i in Equation 3 are $\{1\}$, $\{1, 2\}$, and $\{1, 2, 3, 4\}$, and m_i are the macroblocks which the motion vector v refers to. Motion vectors pointing to regions outside frames are clipped accordingly. $\alpha_i(v)$ reflects the percentage of the ith area contributing to the prediction. One possible scenario is depicted in Figure 3: The values for α_1 to α_4 are $5/128$, $11/128$, $15/128$, and $33/128$ respectively, and their sum is equal to the percentage of the size of the predicted macroblock region. The notation used here is quite generic to keep things simple. Nevertheless, our prioritization scheme covers both forward and backward references and supports multiple references per frame.

Equation 4 defines the mapping from $\delta_{\rightsquigarrow}$ to Ψ_{dep}, which is a strictly increasing function because a loss of macroblock m causes quality degradations in all dependent regions—

the larger these regions are, the more likely it is that the customer feels annoyed by visual artifacts.

$$\Psi_{\text{dep}} = 1 - \left(\frac{1}{\delta_{\rightsquigarrow}(m) + 1} \right)^{\kappa} \in [0; 1] \tag{4}$$

The factor κ is used to adapt Ψ_{dep} to the expected amount of temporal prediction, depending on the GOP structure used and the maximum number of reference frames.

The last sub-function Ψ_{mv} considers the length of motion vectors in both spatial and temporal domains. As mentioned before, broken dependencies lead to distortion in dependent macroblocks because decoders have to guess the region used for prediction. A common trick known as *motion copy* is to inspect motion vectors of neighboring macroblocks and interpolate the missing motion vector by taking the mean over available motion vectors within a certain search radius. Such algorithms usually work quite well, but results get increasingly inaccurate with growing distance to the reference macroblock. Accordingly, sequences with inhomogeneous motion characteristics are more vulnerable to data loss than homogeneous or slow-motion scenes with respect to decoded video quality. This motivates the formulation of Ψ_{mv}, defined in Equation 5.

$$\Psi_{\text{mv}}(m) = \sum_{v_i \in MV(m)} \frac{t_{\text{mv}} \cdot \frac{\text{len}(v_i)}{\sqrt{w_f^2 + h_f^2}} + (1 - t_{\text{mv}}) \cdot \frac{|\Delta_{\text{ref}}(v_i)|}{\max \Delta_{\text{ref}}}}{2 \cdot \#(\text{MV}(m))} \tag{5}$$

We inspect all motion vectors v_i of macroblock m and consider their spatial length $\text{len}(v_i) = \sqrt{x(v_i)^2 + y(v_i)^2}$ together with their temporal length $\Delta_{\text{ref}}(v_i)$. The latter is defined as the difference between the display timestamps of current frame and reference frame, and can be negative in case of backward predictions. $w_f \times h_f$ are the dimensions of the video sequence, and $\max \Delta_{\text{ref}}$ is the upper bound on the temporal prediction distance specified at encoding time. Furthermore, $\#(\text{MV}(m))$ is the number of motion vectors associated with macroblock m, and t_{mv} controls the balance between the impact of temporal and spatial distance on Ψ_{mv}. The basic idea behind Ψ_{mv} is to consider for each motion vector of a macroblock its spatial as well as its temporal expansion. At macroblocks belonging to regions with a high amount of motion, $\text{len}(v)$ is rather large, and loss concerning such regions is much more demanding to conceal. Additionally, error concealment mechanisms more likely use temporally closely co-located data, and consequently, macroblocks having motion vectors with high values of Δ_{ref} cause inadequate error masking with high probability.

With regard to performance, the computational effort to calculate the estimates is negligible because the bit stream has to be processed only partially—the rather expensive process of decoding residuals can be skipped. Memory consumption is also insignificant because the maximum number of frames to be simultaneously analyzed is limited by the current GOP size. Finally, it has to be mentioned that especially in live video streaming scenarios, it may happen that not the entire group of pictures is available at the sender's buffer upon sending of its first frame. Consequently, Ψ_{dep} cannot always be completely derived in such situations, and the algorithm is forced to assume that there are no further temporal dependencies, leading to decreased accuracy of estimates.

IV. Unequal Rateless Codes

Digital fountain codes, also referred to as rateless codes, are a special class of error protection codes for erasure channels in communication networks. The concept of *digital fountains* was introduced by Byers et al. [18], inspired by the perception of filling an empty bucket from an endless supply of digital water drops. As the name suggests, their code rates do not have to be fixed in advance, as opposed to previous coding techniques like Reed-Solomon codes and Tornado codes. Accordingly, the source can generate from a given set of input symbols of size k a potentially infinite number of encoded symbols in a computationally inexpensive fashion, which makes codes of this class applicable to channels with arbitrarily high loss probabilities. An idealized digital fountain has the property that any k encoded symbols given, the receiver is able to reconstruct the complete input symbol set. So far, no rateless codes are known which fulfill this property, but *near optimal* solutions exist—LT codes and Raptor codes are probably the most prominent of them.

The *rateless* property makes digital fountain codes attractive for a variety of applications—video streaming and multicast content delivery are only two of them. The major advantage is that retransmissions become expendable, which reduces delay and solves the problem of feedback implosion in case of multicast. In the context of this paper, we focus on LT codes, proposed by Luby [19]. The encoding process involves two distributions: a member distribution used for selecting addends (input symbols) to contribute to the generation of the current encoded symbol, and a degree distribution responsible for determining the degrees of encoded symbols. By default, LT codes use a discrete uniform member distribution. However, the design of the degree distribution is critical for the overall system and has a strong impact on coding efficiency. A well chosen degree distribution $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$ and a sufficiently large k provided, the overhead in realistic scenarios can be reduced down to 5%.

The creation of encoded symbols is simple: by using the degree distribution, we randomly choose a degree $d \leq k$ and subsequently select d pairwise unequal input symbols s_i. The encoded symbol is derived as $e = \bigoplus_{i=1}^{n} s_i$, having a computational complexity of at most $\mathcal{O}(ln(k))$ (a reasonable degree distribution is assumed). A packet carrying an encoded symbol additionally has to contain information from which the receiver can deduce the original addends of the

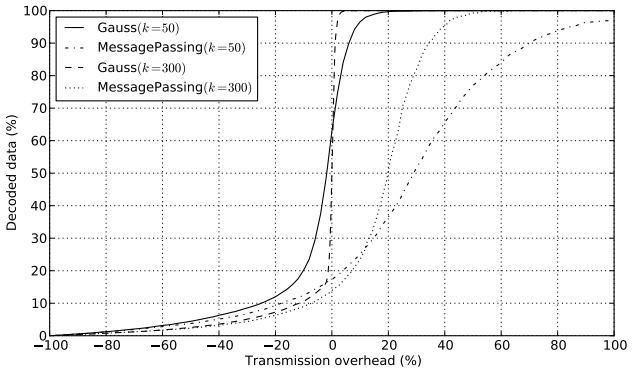

Figure 4. Comparing decoder performance using a Robust Soliton distribution with $\delta = 0.4$, $c = 0.1$; the graph shows the mean over $n = 1000$ tests.

symbol. One space-saving way is to ensure that sender and receiver use the same random generator and to send the seed used for selecting the addends as the ID of the encoded symbol. Decoding is normally done using message passing techniques with a complexity of $\mathcal{O}(k \cdot ln(k))$. However, due to k being rather small in our system ($50 \lesssim k \lesssim 400$), we implemented a modified binary Gaussian elimination solving mechanism which in the worst case needs $\mathcal{O}(k^2)$ operations per received symbol. The data structure used is a bit-matrix of variable size whose height grows with every incoming encoded, non-redundant symbol, and shrinks together with the matrix-width whenever such a symbol can be resolved. As depicted in Figure 4, this decoding mechanism allows us to considerably reduce the transmission overhead when k is chosen from the target range.

To adapt the existing approach to our needs, the mapping from data portions of the video stream to source blocks and input symbols of the rateless coder has to be fixed. Considering the self-contained nature of GOPs, they are the obvious choice for the former, and it seems natural to use NAL units as input symbols as their size can be adjusted to fit into network packets. The initially uniform member distribution has to be replaced by the one specified in Equation 6 to incorporate the importance of input symbols s_i of the current source block $\{s_1, \ldots, s_k\}$. Consequently, the probability of successfully decoding perceptually more important input symbols is increased whereas input symbols with low Ψ-values are less likely selected during symbol creation.

$$P_\Psi(s_i) = \frac{1 - t_\Psi \cdot (1 - \Psi(s_i))}{k - t_\Psi \cdot \left(k - \sum_{j=1}^{k} \Psi(s_j)\right)} \tag{6}$$

The appropriate choice of factor $t_\Psi \in [0;1]$ is crucial: if a too high value for t is chosen, the impact of NAL unit priorities is too strong, which leads to a significant transmission overhead. It might even happen that certain input symbols are undecodable at the sender, independent of the transmission overhead ε. On the other hand, values close to zero tend to eliminate the benefits of the perceptual estimation process.

By once again examining Figure 4, one can see that the intermediate decoding performance is poor when using the "standard" Robust Soliton distribution as proposed by Luby [19]. The curves are rather flat at the beginning and get steeper only around zero (meaning that the decoder roughly received k encoded symbols). Numerous publications exist, which deal with the design of appropriate degree distributions targeting the minimization of transmission overhead. Unfortunately, there is only a very limited amount of work that investigates intermediate decoding performance. This is due to the fact that partial information is normally considered useless in generic content distribution networks. In contrast, it is of great value to multimedia applications. As an example, Chen et al. [20] use a covariance matrix adaptation evolution strategy to, on the one hand, reduce the expected overhead ε, and on the other hand, maximize the percentage of decoded data when ε is fixed. The approach looks quite promising, but it would have been of much greater interest to find degree distributions which maximize the decodable symbol ratio at arbitrary, preferably negative values of ε. Sanghavi [21] focuses on that specific problem and derives optimal degree distributions for $k \to \infty$. Simulations showed that systems using such distributions indeed have a higher intermediate decoding ratio for huge values of k, but they perform unsatisfactorily for small source blocks. Rahnavard et al. [22] have the same objective and apply genetic algorithms to calculate optimal degree distributions for both finite and infinite values of k. Their argumentation is sound, but they merely concentrate on intermediate decoding performance, and it is unclear how fast the decoding process advances after the reception of at least k symbols. Furthermore, both approaches presume that the typically used belief propagation decoder is in place, which contrasts with our setting.

We therefore use the hereinafter defined custom mechanism $\mathcal{R}(\tau, \beta, \delta, c)$ for choosing encoding symbol degrees, operating as follows: as long as the number of produced encoded symbols k' satisfies $\frac{k'}{k} \leq \tau$, all encoded symbols have degree one. Beyond this threshold, a shifted and appropriately scaled Robust Soliton distribution[1] $\mathcal{D}_{\text{rs}}$ is used as specified in Equation 7.

$$\mathcal{R}(\tau, \beta, \delta, c) = \begin{cases} 1 & \text{if } k' \leq \tau k \\ \lceil \beta k + (1-\beta)\text{rnd}(\mathcal{D}_{\text{rs}}(\delta, c)) \rceil & \text{otherwise} \end{cases} \tag{7}$$

The mechanism's design is straightforward: we initially produce a certain amount of immediately decodable symbols and at some point switch to the production of higher degree symbols according to the distribution $\mathcal{D}_{\text{rs}}$, which is completely defined by its parameters δ and c. With β, we can

[1] A definition of the Robust Soliton distribution can be found in [19].

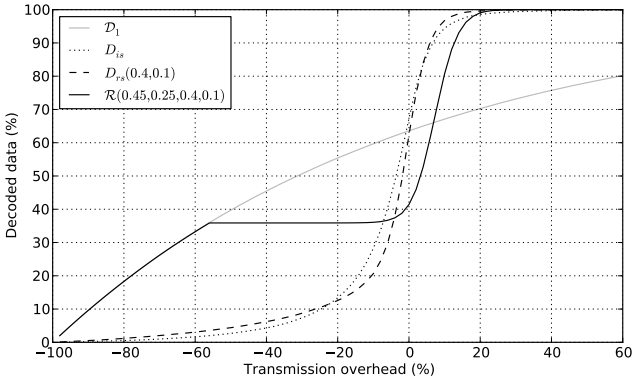

Figure 5. Intermediate decoding performance of degree distributions; Gauss elimination decoder; $k = 50$; the graph shows the mean over n = 5000 tests.

control the symbol degrees' lower bound, which determines the decoding curve's steepness when k' approaches k.

As a result, the percentage of intermediately decodable data is considerably higher as compared to plain $\mathcal{D}_{\text{rs}}$, which leads to a higher amount of recoverable data in case of early termination. Note that the proposed mechanism $\mathcal{R}$ benefits from the Gauss decoder's robustness—when using belief propagation decoding techniques, further adjustments would have to be considered to ensure a steady decoding progress. Figure 5 shows a comparison of the decoding performance of different degree-generating mechanisms. Due to $\tau = 0.45$, $\mathcal{R}$ behaves the same as a system which emits encoded symbols of constant degree one (referred to as $\mathcal{D}_1$) while sending the first 23 symbols. With regard to the expected transmission overhead, $\mathcal{D}_1$ performs very poor, whereas the difference between the proposed mechanism $\mathcal{R}$ and the commonly used distribution $\mathcal{D}_{\text{rs}}$ is negligible. When considering robustness, it is easy to see that packet loss affecting some of the first τk encoded symbols simply degrades $\mathcal{R}$ towards $\mathcal{D}_{\text{rs}}$, whereas loss beyond this threshold defers the decoding process in the same way as it would happen with a pure $\mathcal{D}_{\text{rs}}$-encoder.

In conclusion, it has to be mentioned that $\mathcal{R}$ was intuitively designed with a strong focus on simplicity and does not behave optimally at arbitrary loss rates although it satisfyingly meets our requirements, but this is not the goal of this paper and is a subject of further research.

V. Multicast Video Streaming

Fountain codes in combination with the unequal error protection scheme proposed in this work are suitable for integration in various video streaming scenarios. They range from simple unicast connections to one-to-many streaming applications and systems which support many-to-many multicast. Without loss of generality, we focus on the subsequently described scenario, which will be used throughout the remainder of this paper to point out the benefit of our approach. In that discussion, the current time t_{now} and the display time $t_{\text{play}}(f)$ are absolute values, and the latter refers to the earliest point in time when clients are expected to play back the corresponding frame f. When applied to a set of frames $\{f_1, \ldots, f_n\}$ such as a source block, display time $t_{\text{play}}(\{f_1, \ldots, f_n\})$ always relates to the smallest absolute picture time-stamp $\min(\{t_{\text{play}}(f_1), \cdots, t_{\text{play}}(f_n)\})$.

We consider an infrastructure which consists of one media server streaming live video data over UDP to an arbitrary number of multicast subscribers. All communication is unidirectional, and therefore, feedback does not occur, rendering the sending application unaware of delivery channels' characteristics. The server merely adapts its sending rate based on the stream's current bit rate plus an additional overhead Δs. This overhead can be adjusted according to the expected sum of both overhead originating from the rateless encoding scheme and additional symbols[2] needed to cope with packet loss along delivery paths. The source blocks $\{B_1, \ldots, B_m\}$ which constitute the video stream are encoded and sent successively, taking timing constraints $t_{\text{play}}(\cdot)$ into account. Intermediately joining receivers start playback immediately after the sender switches to the next source block. As a consequence, $t_{\text{play}}(B_i) - t_{\text{now}}$ is close to the duration of one GOP where B_i is the source block currently being emitted.

To fully cover the *live*-aspect of this scenario, we define two more parameters, $t_{w_\triangleright}$ and $t_{w_\triangleleft}$, which specify the allowed sending window of source blocks: the server may transmit symbols corresponding to a specific source block B_i only if $t_{w_\triangleright} \leq t_{\text{play}}(B_i) - t_{\text{now}} \leq t_{w_\triangleleft}$. $t_{w_\triangleleft}$ determines the difference in time between $t_{\text{play}}(B_i)$ and the time when B_i becomes available to the sender, being either received from a parent server or locally produced on-the-fly by a video encoder. On the other hand, $t_{w_\triangleright}$ is designed to keep the number of transmitted packets which arrive at the receiver after the display time of their content at a reasonable level. A fitting choice for $t_{w_\triangleright}$ would be a value greater than the packets' average transmission time.

VI. Performance Evaluation

To evaluate the performance of the proposed unequal error protection scheme $\mathcal{S}_\Psi$, we consider the one-to-many multicast scenario described in the previous section. Experiments were conducted in a local testbed consisting of five Linux nodes: one node serving as sender, three clients and one intermediate node acting as router. The fundamental parameters loss and delay were set via Netem [23], installed as scheduling policy at the router. In parallel, we created the simulator *dfSim* to conduct quick offline experiments, encompassing all network and content related aspects covered in this paper. During testing, it turned out that both

[2]Each network packet contains exactly one encoded symbol, hence we use both terms interchangeably.

experimental environments deliver almost the same results. Therefore, and due to the fact that tests conducted in the testbed come along with a significantly larger expenditure of time, the majority of results presented in this work was obtained by using *dfSim*. Moreover, using the simulator software has the advantage of making loss patterns, which affect video streams, reproducible, thus increasing the comparability of different content prioritization schemes under equal loss conditions. Up to now, *dfSim* supports various loss and delay models of which we chose Gilbert-Elliot $\mathrm{GE}(p_{\mathrm{G}\to\mathrm{B}}, p_{\mathrm{G}\leftarrow\mathrm{B}}, p_{\mathrm{G}}, p_{\mathrm{B}})$ for modeling loss [24] and a normal distribution $\mathcal{N}(\mu, \sigma^2)$ for creating forward trip delays. For encoding sequences, we used the open source H.264/MPEG-4 AVC encoder *x264*[3] along with the parameters *slice-max-size=1420*, *qp=30*, *keyint=16*, *bframes=5*, *ref=16*, *direct=temporal*, and *partitions=all*. The test sequences used are multifaceted in terms of resolution and temporal and spatial complexity. With regard to system parameters, we fixed some of them as follows: $\omega_{\text{type}} = 0.27$, $\omega_{\text{dep}} = 0.44$, $\omega_{\text{mv}} = 0.29$, $\kappa = 0.4$, $t_{\text{mv}} = 0.652$, $t_\Psi = 0.4$, and $\mathcal{R}(\frac{9}{20}, \frac{1}{4}, \frac{2}{5}, \frac{1}{10})$. Just as a quick reminder: ω_i controls the impact of the three components of Ψ, κ determines the degree of decrease of Ψ_{dep} when temporal prediction chains grow long, and t_{mv} balances the impact of spatial and temporal expansion of motion vectors. Additionally, t_Ψ specifies how much Ψ influences the member selection process of the rateless encoder, and $\mathcal{R}$ is the encoder's strategy to choose symbol degrees.

In what follows, we compare the performance of the proposed mechanism $\mathcal{S}_\Psi$ with those of two other prioritization strategies $\mathcal{S}_{\text{type}}$ and $\mathcal{S}_{\text{area}}$. $\mathcal{S}_{\text{type}}$ ranks slices according to their type and issues priorities based on the assumption that intra-coded slices are the most important components, and bidirectionally-coded slices are those which can be concealed best in case of loss. $\mathcal{S}_{\text{area}}$ counts the number of macroblocks each slice contains and favors those covering a large area of a frame over slices which only carry a minor number of macroblocks. So far, few related work exists which combines unequal error protection in rateless coding with video streaming (cf. Section II), and they all stick to the types of frames as intuitive assignment criterion for protection levels. The idea behind $\mathcal{S}_{\text{type}}$ is quite similar, but it instead considers the types of NAL units. For the sake of completeness, we also inspect the behavior of a pure rateless encoding scheme, denoted by $\mathcal{S}_{\text{equal}}$, which does not distinguish between input symbols and considers all of them as being equally important.

Figures 6 and 7 depict the results of applying those schemes to the low-resolution sequences FOREMAN and COASTGUARD. The box plots summarize the outcome of 100 repetitions of the same experiment and reflect the qualities of the received videos at all three receivers, having, for the sake of simplicity, the same channel characteristics. It can be observed that both $\mathcal{S}_\Psi$ and $\mathcal{S}_{\text{type}}$ perform considerably well and improve the overall perceptual quality of the video shown at a particular receiver. However, $\mathcal{S}_{\text{area}}$ behaves even worse than the non-adaptive scheme $\mathcal{S}_{\text{equal}}$, which implies that merely focusing on the protection of a maximal frame area does not necessarily lead to increased video quality.

[3]x264 rev. 1797: http://www.videolan.org/developers/x264.html

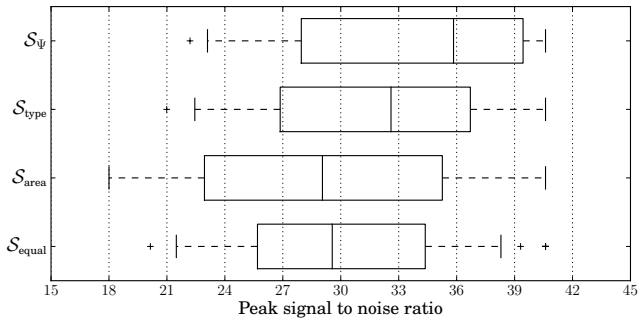

Figure 6. FOREMAN: performance of different error protection schemes; $n = 100$ tests, avg. bitrate: 77 kB/s, loss: $\mathrm{GE}(\frac{1}{20}, \frac{1}{10}, 0, \frac{3}{10})$, delay: $\mathcal{N}(150\,\text{ms}, 20\,\text{ms})$, $\Delta s = 15\,\text{kB}$, window: $t_{w_\triangleright} = 250ms$, $t_{w_\triangleleft} = 1s$.

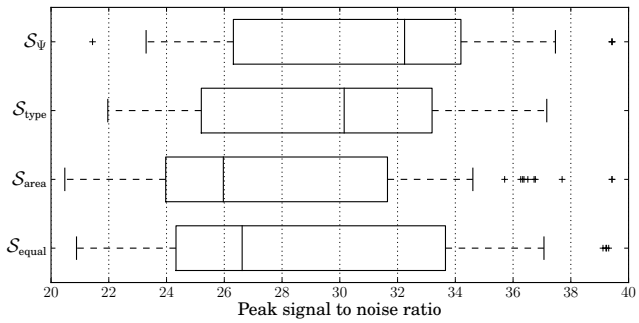

Figure 7. COASTGUARD: performance of different error protection schemes; $n = 100$ tests, avg. bitrate: 135 kB/s, loss: $\mathrm{GE}(\frac{1}{10}, \frac{1}{10}, 0, \frac{2}{5})$, delay: $\mathcal{N}(150\,\text{ms}, 20\,\text{ms})$, $\Delta s = 40\,\text{kB}$, window: $t_{w_\triangleright} = 250ms$, $t_{w_\triangleleft} = 1s$.

At this point, we have to note that the improvement in perceptual video quality is only due to sophisticated selection techniques during the rateless encoding process and not due to anomalies in terms of the amount of received/decodable data at each client. As an example, Figure 8 shows the tx-rate of the sender, the rx-rate at one specific receiver, and the video bit rate of an arbitrary test instance. Because of fixed seeds per test iteration to maximize the comparison's accuracy, the diagram is exactly the same for all four evaluated schemes. However, with regard to decoding efficiency, this fact does not imply that also the decodable symbol ratios are equivalent. Nevertheless, due to the rather moderate choice of $t_\Psi = 0.4$, the ratios strongly correlate, as can be seen in Figures 9 and 10. The diagrams depict the source block sizes and the amount of data processed by the decoder, which encompasses all timely received encoded symbols. More importantly, the amount of

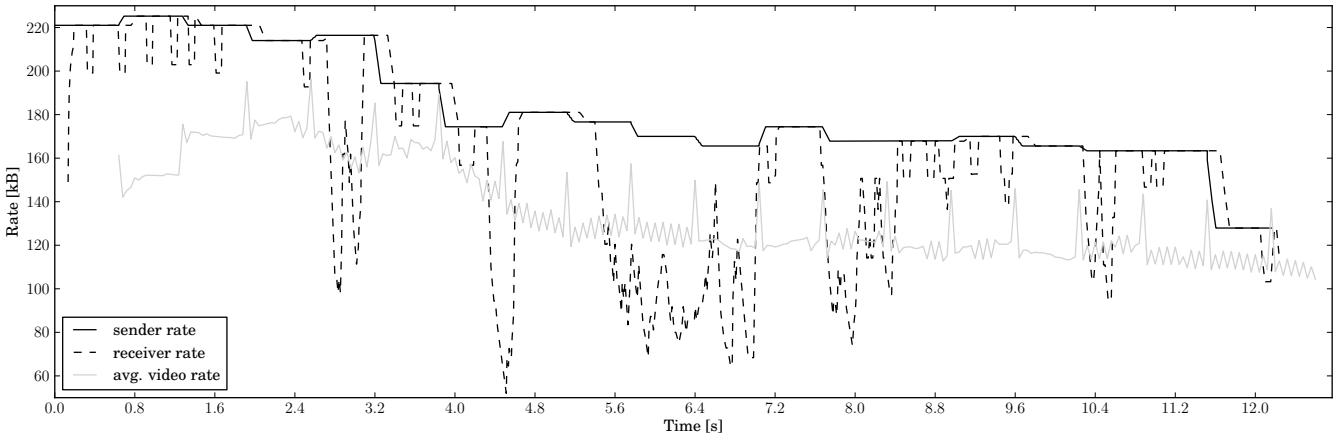

Figure 8. COASTGUARD: sender, receiver, and video bit rate; loss: GE($\frac{1}{20}$, $\frac{1}{10}$, 0, $\frac{3}{10}$), delay: $\mathcal{N}(150\,\text{ms}, 0\,\text{ms})$, $\Delta s = 40\,\text{kB}$, $t_{w_\triangleright} = 250ms$, $t_{w_\triangleleft} = 1s$.

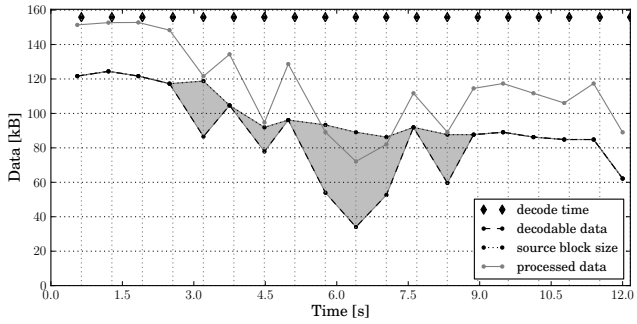

Figure 9. COASTGUARD: amount of recoverable data when using $\mathcal{S}_{\text{equal}}$.

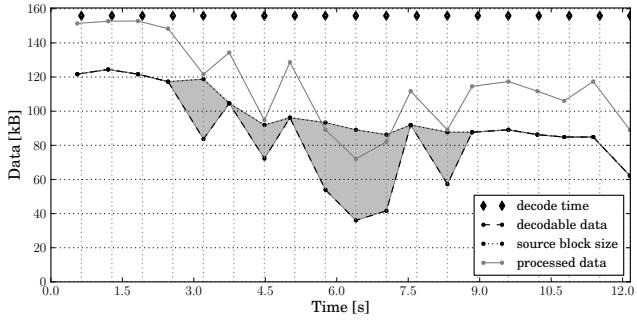

Figure 10. COASTGUARD: amount of recoverable data when using $\mathcal{S}_\Psi$.

recoverable data is displayed, whose negative peaks show strong similarities as pointed out above.

We conclude this section by demonstrating that the proposed strategy $\mathcal{S}_\Psi$ also works well when being applied to high-resolution content. For that purpose, we selected the full-HD (1920x1080) sequence IRONMAN, which has a length of 3000 frames and is characterized by a huge amount of temporal complexity. The result of comparing all four encoding schemes is depicted in Figure 11, and the advantage of favoring $\mathcal{S}_\Psi$ over the other strategies is again

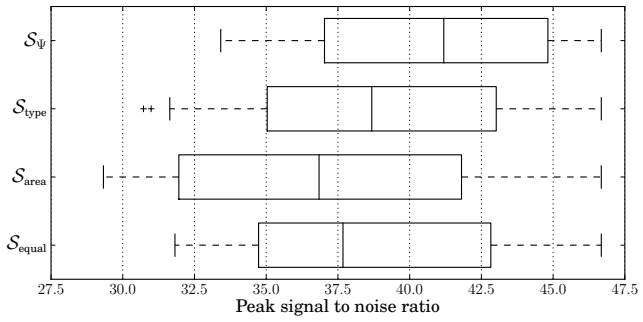

Figure 11. IRONMAN: performance of different error protection schemes; $n = 100$ tests, avg. bitrate: 881 kB/s, loss: GE($\frac{2}{10}$, $\frac{1}{10}$, $\frac{1}{50}$, $\frac{3}{10}$), delay: $\mathcal{N}(150\,\text{ms}, 20\,\text{ms})$, $\Delta s = 180\,\text{kB}$, window: $t_{w_\triangleright} = 250ms$, $t_{w_\triangleleft} = 1s$.

clearly visible. In addition to that, the average transmission overhead needed by clients to succeed in completely decoding source blocks is smaller than that of the previously used low-resolution videos. As described out in Section IV, this can be explained by the higher value of k (≈ 50), which leads to a faster convergence of non-decodable symbols towards zero at the decoder.

VII. Conclusion

In this paper, we proposed a novel distortion estimation scheme for H.264 video streams, which incorporates temporal error propagation effects as well as the amount of motion within scenes, having a considerable impact on error concealment mechanisms. The derivation procedure is computationally lightweight and is suitable for arbitrary loss rates. Next, we presented how such a prioritization scheme can be combined with LT codes and which steps have to be taken to adapt corresponding distributions accordingly. We evaluated the performance of the proposed system, verified the positive impact of sophisticatedly prioritizing media units by comparing it with other approaches, and pointed

out that a well-designed non-uniform member distribution only demands for a minimal number of additional symbols to let the decoding process succeed.

In conclusion, we denote that the proposed system is only of limited benefit in scenarios where the number of lost packets frequently exceeds the encoding overhead. In such situations, we recommend the application of selective reliability schemes. However, if transmission channels are characterized by infrequently occurring but bursty loss patterns, unequal error protection rateless codes represent a promising alternative to other protection techniques. Finally, we believe that the basic idea of dependency tracking and analysis at the macroblock level can be, besides to H.264, also beneficially applied to other, similarly designed video codecs.

REFERENCES

[1] M. Gallant and F. Kossentini, "Rate-distortion optimized layered coding with unequal error protection for robust internet video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 357–372, mar 2001.

[2] R. Hamzaoui, V. Stanković, and Z. Xiong, "Optimized error protection of scalable image bit streams," *IEEE Signal Process. Mag.*, vol. 22, no. 6, pp. 91–107, nov 2005.

[3] Y. Xiaogang, L. Jiqiang, and L. Ning, "Congestion control based on priority drop for H.264/SVC," in *Int. Conf. on Multimedia and Ubiquitous Engineering*, Seoul, South Korea, apr 2007, pp. 585–589.

[4] C. Hellge, T. Schierl, and T. Wiegand, "Mobile TV using scalable video coding and layer-aware forward error correction," in *IEEE Int. Conf. on Multimedia and Expo*, Hannover, Germany, apr 2008, pp. 1177–1180.

[5] A. G. Dimakis, J. Wang, and K. Ramchandran, "Unequal growth codes: Intermediate performance and unequal error protection for video streaming," in *IEEE MMSP*, Crete, Greece, oct 2007, pp. 107–110.

[6] A. Fiandrotti, D. Gallucci, E. Masala, and E. Magli, "Traffic prioritization of H.264/SVC video over 802.11e ad hoc wireless networks," in *Int. Conf. on Computer Communications and Networks*, St. Thomas, US Virgin Islands, USA, aug 2008, pp. 1–5.

[7] J. Korhonen and P. Frossard, "Flexible forward error correction codes with application to partial media data recovery," *Elsevier Signal Processing: Image Communications*, vol. 24, pp. 229–242, 2009.

[8] A. Talari and N. Rahnavard, "Unequal error protection rateless coding for efficient MPEG video transmission," in *IEEE Military Communications Conference*, Boston, MA, USA, oct 2009, pp. 1–7.

[9] E. Masala, A. Vesco, and M. Baldi, "Optimized H.264 video encoding and packetizaton for video transmission over pipeline forwarding networks," *IEEE Trans. Multimedia*, vol. 11, no. 5, pp. 972–985, 2009.

[10] M. Baldi, J. C. D. Martin, E. Masala, and A. Vesco, "Quality-oriented video transmission with pipeline forwarding," *IEEE Trans. Broadcast.*, vol. 54, no. 3, pp. 542–556, sep 2008.

[11] J. C. D. Martin, M. Baldi, E. Masala, and A. Vesco, "Distortion-aware video communication with pipeline forwarding," in *ACM Int. Conf. on Multimedia*. Santa Barbara, CA, USA: ACM, oct 2006, pp. 117–120.

[12] E. Masala and J. C. D. Martin, "Analysis-by-synthesis distortion computation for rate-distortion optimized multimedia streaming," in *IEEE ICME*, vol. 3, Baltimore, MD, USA, jul 2003, pp. 345–348.

[13] J. Korhonen and A. Perkis, "Loss-distortion estimation for practical H.264/AVC streams," in *IEEE Int. Conf. on Multimedia and Expo*, New York, NY, USA, jul 2009, pp. 766–769.

[14] N. Rahnavard, B. N. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1521–1532, apr 2007.

[15] M. Schier and M. Welzl, "Selective packet discard in mobile video delivery based on macroblock-level distortion estimation," in *IEEE Infocom MoViD Workshop*, Rio de Janeiro, Brazil, apr 2009, pp. 1–6.

[16] ——, "Content-aware selective reliability for DCCP video streaming," in *IEEE MCIT*, mar 2010, pp. 53–56.

[17] *H.264 - Advanced Video Coding for Generic Audiovisual Services*, Telecommunication Standardization Sector ITU Std., mar 2010.

[18] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1528–1540, oct 2002.

[19] M. Luby, "LT codes," in *Proc. of the 43rd Symposium on Foundations of Computer Science*. IEEE Computer Society, 2002, pp. 271–280.

[20] C.-M. Chen, Y.-P. Chen, T.-C. Shen, and J. K. Zao, "On the optimization of degree distributions in LT code with covariance matrix adaptation evolution strategy," in *IEEE Congress on Evolutionary Computation*, Barcelona, Spain, jul 2010, pp. 1–8.

[21] S. Sanghavi, "Intermediate performance of rateless codes," in *IEEE ITW*, Tahoe City, CA, USA, sep 2007, pp. 478–482.

[22] A. Talari and N. Rahnavard, "Rateless codes with optimum intermediate performance," in *IEEE Global Telecommunications Conference*, Honolulu, HI, USA, dec 2009, pp. 1–6.

[23] S. Hemminger, "Netem: Emulating real networks in the lab," in *Linux Conference Australia*, apr 2005.

[24] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, pp. 1253–1265, sep 1960.