PAPER

A Novel Structure of HTTP Adaptive Streaming Based on Unequal Error Protection Rateless Code

Yun SHEN[†a)], **Yitong LIU**[†], **Jing LIU**[†], *Nonmembers*, **Hongwen YANG**[†], *Member*, *and* **Dacheng YANG**[†], *Nonmember*

SUMMARY In this paper, we design an Unequal Error Protection (UEP) rateless code with special coding graph and apply it to propose a novel HTTP adaptive streaming based on UEP rateless code (HASUR). Our designed UEP rateless code provides high diversity on decoding probability and priority for data in different important level with overhead smaller than 0.27. By adopting this UEP rateless channel coding and scalable video source coding, our HASUR ensures symbols with basic quality to be decoded first to guarantee fluent playback experience. Besides, it also provides multiple layers to ensure the most suitable quality for fluctuant bandwidth and packet loss rate (PLR) without estimating them in advance. We evaluate our HASUR against the alternative solutions. Simulation results show that HASUR provides higher video quality and more adapts to bandwidth and PLR than other two commercial schemes under End-to-End transmission.

key words: *adaptive streaming, decode probability and priority, overhead, rateless code, unequal error protection, video quality*

1. Introduction

Currently, increasing demand on multimedia services motivates the evolution of multimedia technology, shifting from constant bitrate streaming to HTTP adaptive streaming. As a representation, Dynamic Adaptive Streaming based on HTTP (DASH) [1] is proposed as a 3GPP standardization. The DASH client can dynamically adjust video bitrate based on the available network bandwidth to provide the most fluent playback experience. Thus HTTP adaptive streaming is much more flexible and powerful to adapt changeable network situation.

DASH splits the media content into a series of small segments, encodes them into several versions with different bitrates or resolutions, which is a waste of storage since only one encoded version will be transmitted per segment period. When a DASH service is on demand, the client requests a suitable encoded version based on the estimated bandwidth. The server then sends the requested segment to the client. Such delivery operation is repeated every segment period until the end of connection.

The above delivery requires the client to estimate the current bandwidth before requesting data. However, the bandwidth is quite hard to predict and is changing over time even though there have been various bandwidth estimating methods proposed in previous papers [2]–[4]. Therefore, misestimating the fluctuant bandwidth would lead to poor performance on self-adaptation to bandwidth for adaptive streaming.

Same problem also exits when symbols are encoded by application-layer forward error correction (FEC) [5]–[7] channel code. The code rate is fixed or updated based on the packet loss rate (PLR) prediction. The mismatching between the predicted and the actual PLR may lead to decoding failure or bandwidth wasting in the case of under- or over-estimating respectively.

Therefore, we plan to adopt rateless code (LT code [8]) to solve the above problems. The code rate of it does not need to be predetermined since the encoder can generate a potentially infinite stream on the fly. Thus, the PLR prediction becomes unnecessary for rateless code. Meanwhile, considering symbols in adaptive streaming with different quality levels, it is required to have unequal error protection (UEP) on them, which makes the client decode the most important symbols successfully with the highest priority to ensure the fluent playback.

Several work have focused on UEP rateless code. [9] proposes expanding window fountain (EWF) codes for UEP based on PLR estimation, which also exists the problem on misestimating PLR. [10] deals with UEP on precoder before rateless code, however, the low-degree encoded symbols are hard to connect with high-priority data. And a UEP scheme via distributed rateless code is provided in [11], which consumes high computation complexity. Another UEP scheme is designed in [12] by rebuilding LT code structure, while at the cost of high overhead. [13] proposes a degree-dependent selection concept for UEP rateless codes. While it may be not suitable for real-time streaming due to its complex design. Besides, a Layered-Aware FEC method is implemented to Raptor codes in [14] to achieve UEP on dependent layers also at the cost of high overhead.

Compared with the above UEP schemes, we propose a novel UEP rateless code with special coding graph to guarantee high diversity on decoding probability and priority for different important level symbols with low computation complexity and little overhead. And we design a novel structure of HTTP adaptive streaming based on this UEP rateless code, named as HASUR. It adopts H.264/SVC to encode the raw stream into a single SVC bitstream with one base layer and several enhancement layers, which gradually improve the video quality. Then the symbols are encoded

Manuscript received October 17, 2013.
Manuscript revised June 16, 2014.

[†]The authors are with Wireless Theories and Technologies Lab, Beijing University of Posts and Telecommunications (BUPT), Beijing 100876, China.
a) E-mail: shen0069@gmail.com

DOI: 10.1587/transinf.2013EDP7364

by the UEP rateless code scheme to ensure the high priority symbols (symbols in the base layer) have higher decoding probability than the low priority ones.

Our main contribution is to design a UEP rateless code scheme and apply it to propose a novel structure of adaptive streaming. Our proposed UEP rateless code guarantees perfect UEP performance on different quality level symbols with minimal overhead. And our designed HASUR makes the video stream become more adaptive to the network to ensure the most fluency playback under the fluctuant bandwidth and changeable PLR without estimating them in advance.

The remainder of this paper is organized as follows. In Sect. 2, we illustrate our design on the proposed UEP rateless code. And the performance is analyzed in Sect. 3. Then the structure of our designed HASUR is described in Sect. 4. Section 5 will be the simulation of our HASUR, compared with DASH and HLS. The conclusion will be stated in Sect. 6.

2. The Proposed UEP Rateless Code Based on Hierarchical Coding Graph

To ensure the proposed HASUR becoming more adaptive to the fluctuant bandwidth, an efficient UEP rateless code is necessary to guarantee the diversity on decoding probability and decoding priority for symbols in different quality layers. In this section, we design a UEP rateless code with a hierarchical coding graph to make sure the high priority symbols are recovered before the low priority ones, thus the more important data (symbols in lower layers) can be recovered with lower latency, lower computation and higher probability. Besides, symbols in different layers are encoded dependently to help recovering each other.

2.1 Degree Distribution Analysis

The representative rateless code is LT code, proposed by Luby in 2001 [8]. The *pdf* of degree distribution for LT code obeys the Robust Soliton distribution (RSD) [8], according to which, the belief propagation (BP) decoder can recover k input symbols from any $k + O(k^{\frac{1}{2}} \ln^2(k/\delta))$ encoded symbols with probability $(1-\delta)$ on average $(k \cdot \ln(k/\delta))$ symbol operation [8]. It is obvious that as k turns large enough, the overhead becomes arbitrarily small, as shown in (1).

$$\lim_{k\to\infty} overhead = \lim_{k\to\infty} \frac{k^{\frac{1}{2}} \ln^2(k/\delta)}{k} = 0 \tag{1}$$

It indicates that rateless code with RSD enjoys perfect performance for data with long size. However, under some special situations, RSD is not the optimum solution anymore. For example, many download services tend to be small in size, i.e., k is not large enough, which results in excessive overhead for RSD. There are also plenty work have focused on overhead optimization. [15] proposes a rateless code scheme with non-binary LDPC code to reduce the overhead for small input symbol size. While the scheme is unable to consider the UEP property. In our proposed adaptive streaming scheme, the unit of source block to be encoded is Group of Picture (GOP). Take an example for the video streaming with bitrate = 512kbps, framerate = 30, and GOPsize = 16, the size of each GOP block is $k = 2.80\times10^5$, then according to (1), the overhead is about 0.4565 for $\delta = 0.05$. And it will increase greatly along with bitrate decreasing, which indicates that when the bandwidth turns bad, only low bitrate video streaming is allowed to be transmitted, the excessive overhead brings greatly burden to network, leading to not only decoding failure, but also more serious network situations.

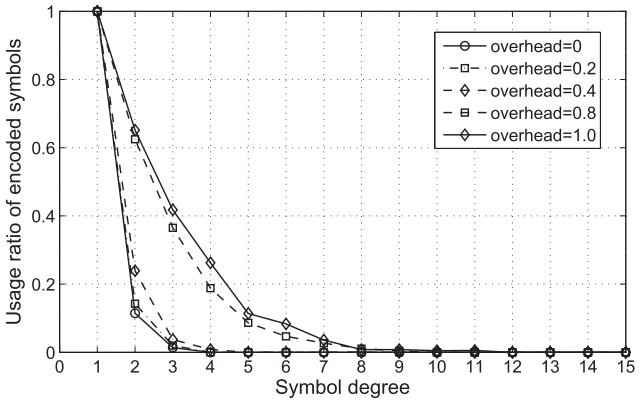


Fig. 1 Usage ratio of encoded symbols with different degrees in decoding process under different overhead.

Except for the extremely overhead of LT code in the above situation, the encoded symbols with high degree are usually of less help for decoding, compared with low degree symbols. Here, an encoded symbol is regarded as *helpful* when an input symbol is recovered through it after the decoding process, which greatly relies on the symbol degree, as shown in Fig. 1, where 10000 input symbols are recovered by encoded symbols with different overhead. From Fig. 1, the encoded symbols with degree 1 are fully used, while the usage ratio of encoded symbols decreases extremely along with degree increasing, especially for small overhead. Here the usage ratio of encoded symbols is defined as the proportion of the *helpful* symbols in the whole encoded symbols with degree i, as expressed in (2).

$$usageratio_i = num_i(helpful)/num_i(total) \tag{2}$$

Considering the curves with $overhead \leq 0.4$ in Fig. 1, it is obvious that only encoded symbols with degree ≤ 3 are much more *helpful* while the rest symbols become nearly useless for decoding process. In this situation, it can be concluded that the low degree encoded symbols are more helpful for decoding than those with high degree.

In view of BP decoder, all input symbols connected with degree 1 encoded symbols are recovered first after the 1*st* decoding iteration, enjoying the highest decoding priority. While those input symbols, recovered by higher degree encoded symbols, are decoded after several iterations. It indicates that these symbols have lower priority, especially

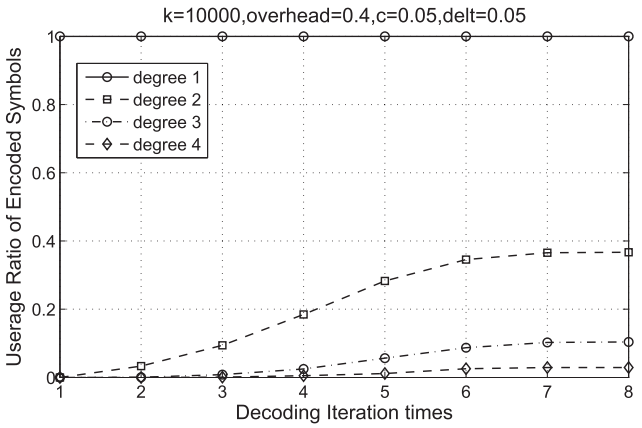


Fig. 2 Cumulative usage ratio of encoded symbols with different degrees along with decoding iteration.

along with degree increasing, as shown in Fig. 2, in which all input symbols are recovered after the 8*th* decoding iteration. Figure 2 gives cumulative usage ratio of encoded symbols with different degrees along with decoding iteration times increasing. From which, the usage ratio of encoded symbols with degree 4 keeps lower than 2% after all decoding iterations, indicating that encoded symbols with higher degree give even less help for decoding.

2.2 UEP Rateless Code Design

From Fig. 1 and Fig. 2, we can conclude that the encoded symbols with low degree are more useful for decoding than those with high degree, especially in the case of low overhead. Besides, an input symbol enjoys higher decoding priority when it is recovered earlier than others. Thus according to Fig. 2, the input symbols connected with degree 1 encoded symbols enjoy the highest decoding priority, and the decoding priority decreases along with the increasing of encoded symbol degree.

Thus, considering the usage ratio of encoded symbols with different degrees, the degree distribution of our rateless code does not follow the robust soliton distribution. Besides, we also design a UEP hierarchical coding graph based on decoding priority of encoded symbols with different degrees to keep high diversity on decoding probability and priority for different important level symbols. Therefore, we abandon encoded symbols with degree larger than 3 and classify input symbols into n layers based on the important level, that is, the symbols in Layer i are more important than those in Layer $i+1$. The most important input symbols are gathered into Layer 1 and encoded with degree 1 to ensure the highest decoding probability and priority. While the input symbols in Layer 2 to Layer m are encoded with degree 2 for some positive integer $m \leq n-1$, and the rest symbols in Layer k ($k > m$) generate encoded symbols with degree 3 or 2.

The coding rules of our proposed UEP rateless code are illustrated as follows. The input symbols in Layer 1 duplicate themselves directly to produce encoded symbols with degree 1. For the encoded symbols in Layer k ($2 \leq k \leq m$),

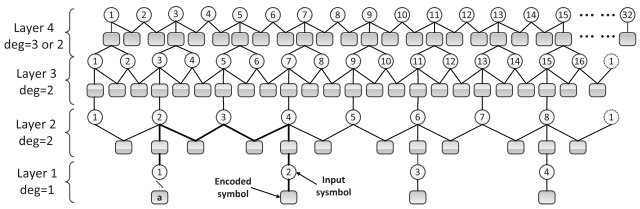


Fig. 3 Topological structure of our proposed coding graph with 4 layers (The bold curve presents an alternative decoding path when encoded symbol a is lost, and the circle number is the index of input symbols in each layer).

parts of them are generated by *XOR*ing one input symbol in Layer k and one in layer $k-1$, and the rest are generated by *XOR*ing adjacent two input symbols in Layer k. Then in Layer k for $m+1 \leq k \leq n$, the encoded symbols with degree 3 are generated by *XOR*ing one input symbol in Layer k and two in Layer $k-1$, besides the adjacent two input symbols in Layer k are *XOR*ed with each other to produce encoded symbols with degree 2. Here we take a topological structure of our proposed coding graph with 4 layers for example as shown in Fig. 3.

From Fig. 3, 60 input symbols are divided into 4 layers to generate 80 encoded symbols. The most important input symbols are contained in Layer 1 at the bottom to be protected by degree 1 encoded symbols. The input symbols in Layer 2 to Layer m ($m = 3$) are protected by degree 2 encoded symbols and in the top layer (Layer 4), the rest input symbols generate encoded symbols with degree 3 and 2. While in decoding process, the input symbols in Layer 1 are recovered firstly as they are connected with degree 1 encoded symbols and then the rest input symbols are decoded step by step from the lower layer to the higher layer. Thus it can be figured out that the more important symbol enjoys higher priority to be recovered.

Besides, an input symbol can be recovered not only from lower layers but also from higher layers due to the encoding dependency between adjacent layers. An input symbol is recovered from the higher layers only when all *helpful* encoded symbols in lower layers are lost. For example, the input symbol 1 in Layer 1 can be recovered from input symbol 2 in Layer 2 along with the decoding path as shown in the bold curve in Fig. 3 when its connected degree 1 encoded symbol a is lost. Such protective measure improves the decoding probability of input symbols at the cost of proper computation complexity increasing.

The above designed UEP code is still a finite-length code with a fixed code rate $R = \frac{N}{K}$, where N and K are the total number of encoded symbols and input symbols. Due to the drawback of fixed-rate FEC code as mentioned in Sect. 1, we take some methods to make our designed UEP code become rateless code. The ideal way to achieve nearly arbitrary code rate is to extend the number of encoded symbols to unlimited large, which leads to the no upper bound for R. Thus, in our proposed adaptive video streaming scheme, the UEP rateless encoding will be operated multiple times in each transmission period T to realize

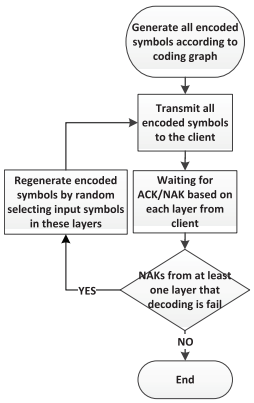


Fig. 4 UEP rateless encoding operation delivery procedure.

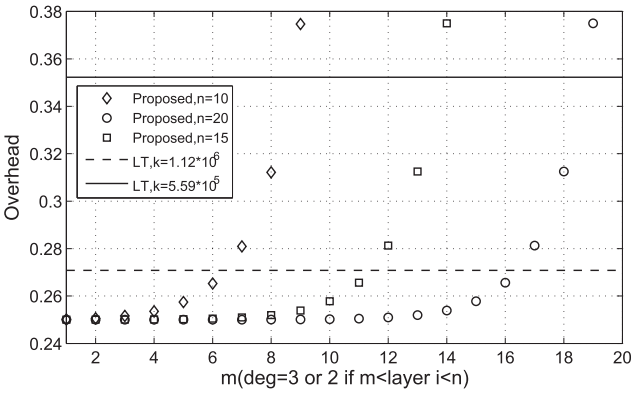


Fig. 5 Overhead Comparing between our proposed UEP rateless code and LT with RSD.

the arbitrary code rate. The delivery procedure is shown in Fig. 4.

According to Fig. 4, the encoder generates all encoded symbols by randomly selecting input symbols according to the proposed coding graph, and then sends them to the client. The client will send back an ACK/NAK based on each layer to inform the server whether a layer is decoded successfully or not. After sending all encoded symbols to the client, when the server receives a NAK that Layer i is failed to be decoded, it will regenerate encoded symbols in Layer i by randomly selecting input symbols in this layer and resend them to the client until no NAK has been sent back to the server, i.e., all layers have been decoded successfully. Such delivery operation effectively increases the number of *helpful* encoded symbols of an input symbol, which increases the decoding probability greatly and keeps the decoding priority unchangeable.

3. Performance Analysis of UEP Rateless Code

The purpose to design this UEP rateless code is to ensure the high diversity on decoding probability and decoding priority for different important level symbols at the cost of a little overhead. Thus, in this section, the performance on our UEP rateless code is analyzed in terms of overhead, decoding probability and decoding priority.

3.1 Overhead

The topological structure of our proposed UEP rateless code can also be expressed as mathematical expression. Define $k(i)$ and $N(i)$ as the number of input symbols and encoded symbols in Layer i. Let K and N be the total number of input symbols and encoded symbols in all layers. Then $k(i)$, $N(i)$, K and N of the UEP rateless code with n layers can be expressed as follows.

$$k(i) = 2^{q+i-1}.\ 1 \le i \le n \tag{3}$$

$$N(i) = \begin{cases} 2^q, & i = 1 \\ 2^{q+i-2} + 2^{q+i-1}, & 2 \le i \le m \\ 2^{q+i-3} + 2^{q+i-1}. & m+1 \le i \le n \end{cases} \tag{4}$$

$$K = \sum_{i=1}^{n} k(i) = 2^q(2^n - 1) \tag{5}$$

$$N = \sum_{i=1}^{n} N(i) = 2^q(5 \cdot 2^{n-2} + 2^{m-2} - 2) \tag{6}$$

where 2^q is the size of Layer 1 for some positive integer q, and the range of m in (4), (6) is $2 \le m \le n-1$. Therefore, the overhead of our proposed UEP rateless code is defined as follows.

$$\begin{aligned} overhead &= \frac{N-K}{K} = \frac{2^{n-2} + 2^{m-2} - 1}{2^n - 1} \\ &\le \frac{1}{4}(1 + 2^{m-n}) \le 0.375 \end{aligned} \tag{7}$$

Besides, *overhead* in (7) can be regarded as a monotonic increasing function of m in range of $2 \le m \le n-1$. Then *overhead* is expressed in (8).

$$overhead = f(m) \ge f(2) \ge 0.25 \tag{8}$$

Confined by (7) and (8), the overhead of our proposed UEP rateless code is in range of $0.25 \le overhead \le 0.375$, much lower than that provided in [12] ($0.31 \le overhead^{[12]} \le 0.5$).

Besides, in the situation mentioned in Sect. 2.1, when the number of input symbols K is not large enough, the overhead of our proposed scheme is comparable with that of LT code with RSD, as shown in Fig. 5. The full line in Fig. 5 presents the overhead of a GOP for a video streaming with bitrate = 1Mbps, framerate = 30 via LT encoding, and the dotted line is the overhead (= 0.27) of that with bitrate = 2Mbps via LT encoding. According to Fig. 5, along with the decreasing of m, the overhead of our UEP rateless code keeps decreasing, even lower than 0.27 (the dotted line), where m is the boundary that the symbols will be encoded with degree 2 if Layer $i \le m$, otherwise, the degree will be 2 or 3. Especially when m is in range of $(n-3) \le m \le (n-1)$, the overhead decreases rapidly. Such phenomenon is suitable for different value of n, that is by choosing suitable m, such as $m = (n-4)$, the overhead of our proposed UEP rateless code with n layers can be further limited under 0.27 for any value of n, as shown

in (9), and is even lower than the overhead of a GOP with $bitrate = 2Mbps$ via LT coding with RSD.

$$0.25 \le overhead \le 0.27 \quad \text{for } 2 \le m \le (n-4) \tag{9}$$

3.2 Decoding Probability and Priority

The input symbols in our proposed coding graph can be recovered through multi-decoding paths. Considering the degree of input symbols, the input symbols in the proposed coding graph can be classified into 5 types, as shown in Fig. 6, where the input symbol B_i in Layer i can be recovered from an encoded symbol b_{i+1} in Layer $i+1$ or one of three encoded symbols a_i in Layer i, and the input symbols D_i in Layer i can be decoded through one of two encoded symbols in Layer i. Here, an input symbol decoded through encoded symbols a_i in the same layer is defined as bottom-to-top decoding, while if an input symbol is recovered by encoded symbols b_{i+1}, we call it top-to-bottom decoding. As mentioned before, the input symbols can be recovered from both bottom-to-top direction and top-to-bottom direction. No matter what decoding direction is, the decoding probability of any input symbols in our proposed coding graph can be expressed by the combination of decoding probability of these 5 types input symbols.

Define $P(X)$ be the decoding probability of input symbol X with loss rate $(1-p)$, and $P(x_i)$ be the probability that any input symbol is recovered by encoded symbol x_i, where $x \in \{a, b\}$. Then, $P(A_i)$ can be expressed in (10).

$$P(A_i) = P(a_i) + P(b_{i+1}) - P(a_i)P(b_{i+1}) \tag{10}$$

where $P(a_i)$ and $P(b_i)$ is expressed by following equations:

$$P(a_i) = \begin{cases} p, & i = 1 \\ \frac{p[p+P(A_i)+P(C_i)]}{3}, & i = 2 \\ \frac{p[P(E_{i-1})+P(D_{i-1})+2(P(A_i)+P(C_i))]}{6}, & 3 \le i \le m \\ \frac{p[P(E_{i-1})P(D_{i-1})+2(P(A_i)+P(C_i))]}{5}, & m < i < n \\ \frac{p[P(E_{i-1})P(D_{i-1})+2(1+p)P(D_i)]}{5}. & i = n \end{cases} \tag{11}$$

$$P(b_i) = \begin{cases} pP(B_i), & 2 \le i \le m \\ \frac{pP(C_i)[P(D_{i-1})+P(E_{i-1})]}{2}, & m < i < n \\ \frac{pP(D_i)[P(D_{i-1})+P(E_{i-1})]}{2}. & i = n \end{cases} \tag{12}$$

Besides A_i types, input symbols can also be decoded via other directions. The decoding probability of other 4 input symbol types are shown as following equations:

$$P(B_i) = 1 - [1 - P(b_{i+1})][1 - P(a_i)]^3 \tag{13}$$

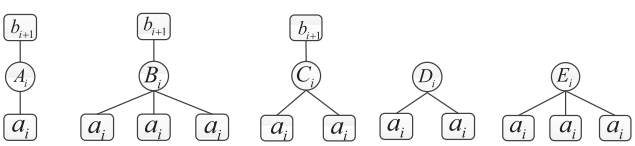


Fig. 6 5 different types of input symbols in proposed coding graph.

$$P(C_i) = 1 - [1 - P(b_{i+1})][1 - P(a_i)]^2 \tag{14}$$

$$P(D_i) = 2P(a_i) - P(a_i)^2 \tag{15}$$

$$P(E_i) = 1 - [1 - P(a_i)]^3 \tag{16}$$

With the combination of the above 5 types input symbols, the decoding probability of arbitrary input symbol X_i in Layer i can be expressed as

$$P(X_i) = \begin{cases} P(A_i), & i = 1 \\ [P(B_i) + P(C_i)]/2, & 2 \le i \le m \\ [P(B_i) + 3P(C_i)]/4, & m < i < n \\ [P(E_i) + 3P(D_i)]/4. & i = n \end{cases} \tag{17}$$

The above equations give the general form for decoding probability of arbitrary input symbols in our proposed coding graph, according to which, it is obvious that the decoding probability in different layers is interdependent with each other.

However, it is too complicated to give an exact value for a symbol according to (17). For simplicity, only considering bottom-to-top decoding path for Layer i, where $2 \le i \le n$. Thus, input symbols B_i and C_i are simplified to type E_i and D_i respectively. Due to $P(B_i) \ge P(E_i), P(C_i) \ge P(D_i)$, the lower bound of decoding probability is calculated in (18).

$$P_{l.b.}(X_i) = \begin{cases} P(A_i), & i = 1 \\ [P(E_i) + P(D_i)]/2, & 2 \le i \le m \\ [P(E_i) + 3P(D_i)]/4. & m < i \le n \end{cases} \tag{18}$$

where

$$P(X_1) = P(A_1) = 1 - (1-p)(1 - pP(E_2))$$

$$P(D_i) = 1 - (1 - pP(E_i))^2$$

$$P(E_i) = \begin{cases} 1 - (1 - pP(X_{i-1}))(1 - pP(D_i)))^2, & i \le m \\ 1 - (1 - pP(X_{i-1})^2)(1 - pP(D_i)))^2, & o.w. \end{cases}$$

Here we present an intuitive description on decoding probability in each layer, as shown in Fig. 7, where input symbols are encoded into 4 layers. The lines with circular marker in Fig. 7 are the decoding probability of each layer calculated by (18) for $n = 4$, $m = 3$, and the lines with rectangle marker

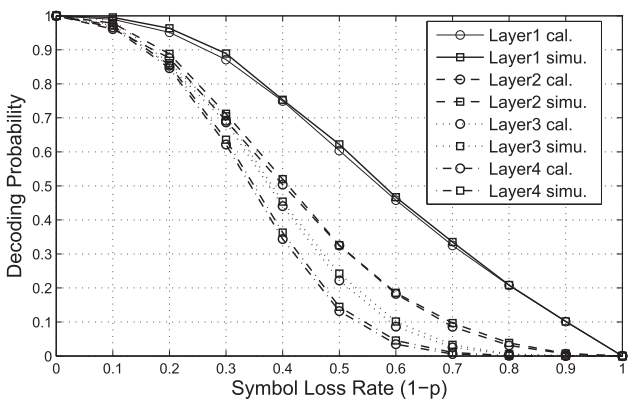


Fig. 7 Decoding probability for different layer under increasing symbol loss rate (cal.: calculated by (17), simu: simulation result).

are the simulation results with $K = 10000$. Since the value calculated by (18) is the lower bound of decoding probability, the calculated decoding probability is a little smaller than the simulation result, as shown in Fig. 7. While the mismatch between the calculated and simulated value is no larger than 0.02.

Besides, from both calculated and simulation results in Fig. 7, only a few layers lose a little information when the symbol loss rate is small, while along with the increasing of symbol loss rate, the decoding probability of the more important layers become much higher than the layers with the important level. It can be concluded that with our designed UEP rateless encoding, the input symbols in arbitrary layer enjoys greatly high decoding probability when the symbol loss rate is small, while with the loss rate increasing, it provides high diversity on decoding probability between different layers to achieve our purpose on unequal error protection to different important symbols.

After decoding probability, decoding priority is to be discussed. Any input symbols X_i in our proposed coding graph, except those in the top layer, can be recovered from both a_i and b_{i+1} encoded symbols. Then the input symbol X_i enjoys the *ith* decoding priority when it is recovered from a_i in Layer i, otherwise, it will have the *jth* decoding priority, where j is in range of $(i+1) \le j \le (2n-i-1)$. Notes that the longest decoding priority $j = 2n - i - 1$ is from Layer 1 up to Layer n (the top layer), then following the top-to-bottom path, goes down to the X_i in Layer i. Considering that we adopt BP decoding process, the decoder will firstly recover the encoded symbols with degree 1 and decrease the degree of other encoded symbols connected with the decoded input symbol. For instance, after the first decoding iteration, all input symbols in Layer 1 have been recovered, and encoded symbols in Layer 2 will decrease their degree and to be decoded in the next decoding iteration. Thus in BP decoding process, the input symbol X_i will be recovered from a_i in Layer i with the *ith* decoding priority. The situation that X_i decoded by b_{i+1} only happens when encoded symbol a_i is lost. In our discussion about the decoding priority, we assume that all encoded symbols are sent to the decoder. Figure 8 presents the decoding priority of our proposed UEP rateless code with 248 input symbols. The recovered symbols are assigned from Layer 1 to Layer 5.

According to Fig. 8, the decoder recovers all input symbols in Layer 1 after the first decoding iteration, and then all in Layer 2 and parts in Layer 3 in the 2*nd* iteration. Thus it is obvious that the *ith* decoding operation can recover the whole input symbols in Layer i and part in Layer $i+1$, proves that the more important symbols enjoy higher decoding priority than other symbols.

4. Adaptive Streaming Based on UEP Rateless Code

Based on the UEP rateless code designed above, the delivery mechanism of our HASUR is shown in Fig. 9. The raw video stream captured by the camera from time $t = 0$ to $t = T$ is encoded as the first source block by H.264/SVC encoder. The source block is partitioned into one base layer, which provides the basic quality by decoding itself independently, and several enhancement layers, providing higher quality gradually together with the base layer. For simplicity, we ignore source encoding time, which depends on the implementation of encoder and is usually very short. Then at $t = T$, the server applies our designed UEP rateless code to the source block, which has been illustrated in Sect. 2. After channel coding, the encoded block is sent to the client and the client tries to recover the source block from the received symbols. Because of characteristics of UEP rateless code, the client will firstly decoded the base layer successfully and then each enhancement layers. At $t = 2T$, the client refuses to receive any symbols from the server and stops channel decoding. While due to the packet loss or bandwidth fluctuation, some encoded symbols are lost or arrive too late to be useful, thus not all layers can be recovered successfully within the stipulated time $\Delta t = 2T - T$. Then those recovered symbols in successfully decoded layers will be fed into the source decoder at $t = 2T$. Source decoding can be done with almost no delay to provide the first frame of the recovered video stream for playback at $t = 2T$, which ensures the maximum playback latency is $2T$. Increasing T will increase the size of source block, which leads to more efficient coding but also to a longer playback latency.

The same process is repeated at $t = iT$, $i = 1, 2, \ldots$. In a word, each encoded block is generated and transmitted at

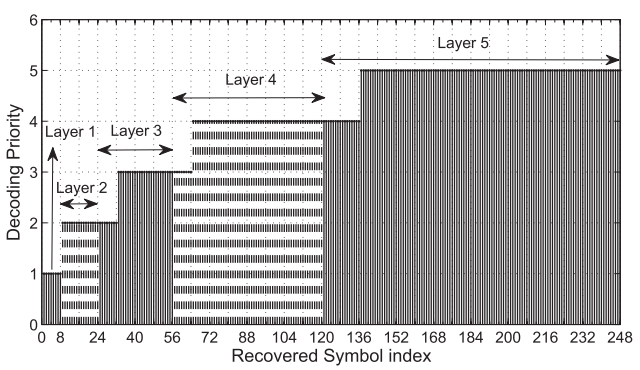


Fig. 8 Decoding priority for input symbols in different layers.

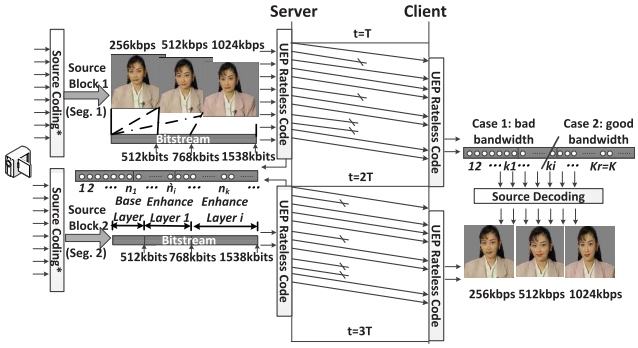


Fig. 9 Delivery mechanism of the proposed adaptive streaming structure (Source Coding*: H.264/SVC).

$t = iT$, while at $t = (i + 1)T$, the client decodes each layer based on the encoded symbols received during the transmission period of T. The above process makes the video streaming more adapt to the bandwidth fluctuation without predicting the bandwidth and PLR *in prior*. When the bandwidth turns worse at one transmission period, the client receives only a little encoded symbols, and decodes only the base layer, providing the basic quality to the users. While in the next period, the bandwidth becomes better, and the client recovers not only the base layer, but also many enhancement layers, giving users much better playback experience.

5. Simulation Results

The performance of our designed UEP rateless code analyzed in Sect. 3 is suitable for general data. In this section, we will test the performance of our HASUR mentioned in Sect. 4 with this UEP rateless code.

5.1 Test Bed Environment

In the simulation, we compare our HASUR with two commercial HTTP adaptive streaming schemes, DASH and HLS, in the case of fluctuant bandwidth and changeable PLR respectively. To this end, we implement the network simulator application-layer module, named *Dummynet*, between server and client to control bandwidth and PLR. The server of HASUR processes source coding (H.264/SVC) and channel coding (proposed UEP rateless code), packages encoded symbols into ip packet and sends them through the network simulator to the client. While the client is in charge of decoding received symbols, reconstructing images and playing back video streaming. Figure 10 shows the topology for simulation, from which, the network simulator takes charge of bandwidth fluctuating and arbitrary changeable PLR in real-time.

The media used in the simulation is the CIF *tempete* video sequence (30*fps*, 352x288), encoded by H.264/SVC, with one base layer and four enhancement layers, as shown in Table 1. The sequence is segmented into GOPs in size of 16 frames, and every $\frac{16}{30}$ seconds, our UEP rateless encoder is supplied by a new GOP data as the source block, the size of which is about $K = 1.6 * 10^5$ Byte. It is obvious that in our simulation, the symbols in base layer are with the most important level and are encoded with degree 1, those in enhancement layer 1 and 2 are then encoded with degree 2, while the rest layers are encoded with degree 3 and 2, which gives the overhead of each segment about 0.28.

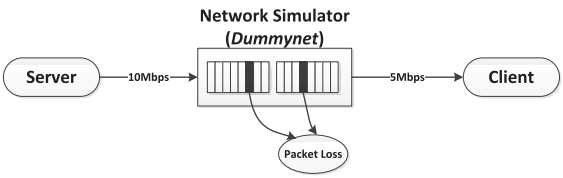


Fig. 10 Simulation topology.

5.2 Experimental Results

In this subsection, we discuss the quality of received media at the client of HASUR, DASH and HLS, considering different PLR and the fluctuant bandwidth respectively. Here we adopt Peak Signal to Noise Ratio (PSNR) as the metric of the media quality, as shown in Table 1, the higher quality enjoys higher PSNR.

Firstly, we fix bandwidth at 5Mbps and limit PLR various in range of 1% to 5%. Under this network situation, the client requests streaming service via HASUR, DASH and HLS. Due to packet loss, the above 3 adaptive streaming adjust their own segment bitrate automatically to self-adapt to the current network situation. The average PSNR is calculated to give the entire judgement of received media quality under different PLR, as shown in Fig. 11, It is obvious that along with the increasing of PLR, all of 3 adaptive streaming degrade their video quality to ensure their playback fluency. While the quality degradation of our HASUR is the weakest, which indicates that our proposed scheme guarantees higher video quality than DASH and HLS under various PLR. In other word, our HASUR gives good performance on efficient resilience against packet loss.

Besides, The situation about the fluctuant bandwidth is also considered. When an adaptive streaming is transmitting and playing back, the network simulator changes the bandwidth from 256kbps to 3Mbps randomly, which leads to our HASUR, DASH and HLS keeping changing their segment bitrate to suit the real-time fluctuant bandwidth as possible as they can. Figure 12 gives the situation of real-time streaming bitrate change under the fluctuant bandwidth when the adaptive streaming is on-demand.

Table 1 H.264/SVC encoded *tempete* video sequence.

Layer	Bitrate (kbps)	PSNR-Y (dB)
Base Layer	79.72	28.3996
Enhancement Layer 1	239.98	29.8272
Enhancement Layer 2	533.02	31.8428
Enhancement Layer 3	1265.20	34.2681
Enhancement Layer 4	2696.40	37.7825

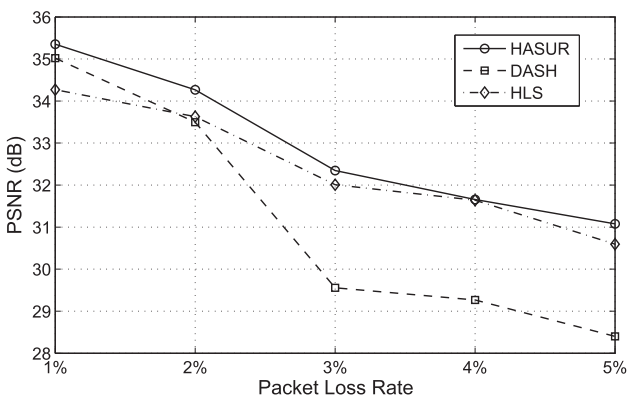


Fig. 11 PSNR of adaptive streaming under different PLR.

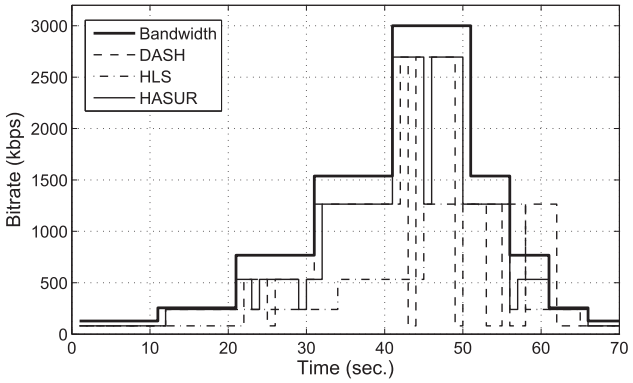


Fig. 12 The real-time streaming bitrate under the fluctuant bandwidth when the adaptive streaming is on-demand.

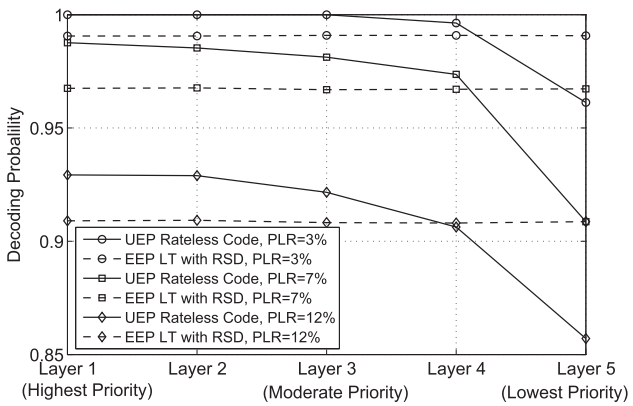


Fig. 13 Decoding probability of different layers with the proposed UEP rateless code and EEP LT code respectively under different PLRs. (overhead = 0.28).

From Fig. 12, when the bandwidth increases, all of the 3 adaptive streaming improve their bitrate as a whole and decrease them when the bandwidth becomes worse. While compared with HLS, HASUR is much more suitable to the fluctuant bandwidth, which even enjoys similar performance with DASH. It should be noted that we come to this conclusion under the situation with End-to-End transmission, ignoring the interference caused by multi-users and only considering bandwidth fluctuating.

In addition, we also research the decoding probability of different layers based on the UEP code scheme when compared with Equal Error Protection (EEP) LT code scheme. The simulation results are shown in Fig. 13. According to it, under the same PLR, the decoding probability of each layer based on EEP scheme keeps unchanged due to the equal protection on each layer. While, with the UEP scheme, the decoding probability of high priority symbols are higher than that of the low priority ones. Besides, compared with EEP scheme, our UEP scheme enhances the decoding probability of the higher priority symbols by sacrificing that of the lowest priority symbols.

6. Conclusion

In this paper, we designed a UEP rateless code and proposed HASUR, a novel structure of HTTP adaptive streaming using this UEP rateless code. Our adaptive streaming scheme makes video stream become more adaptive to the network. It also ensures the most fluency playback and satisfactory video quality under the fluctuant bandwidth and changeable PLR, which are unnecessary to be estimated in advance.

Our designed UEP rateless code in this adaptive streaming is very suitable in best-effort packet network. It guarantees the high decoding probability and priority of important data with very little overhead and computation complexity in minimal decoding iteration. Our UEP rateless code enjoys even lower overhead than that of LT code with RSD under the situation that the size of input symbols is not large enough.

The performance of our proposed HASUR has been verified by comparing with DASH and HLS. Under the situation of End-to-End transmission, HASUR provides better video quality under changeable packet loss rate and becomes more adaptive to bandwidth fluctuating, especially than HLS.

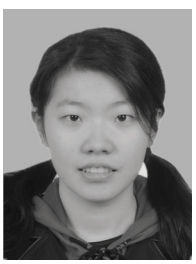
References

[1] Thomas Stockhammer, “Dynamic adaptive streaming over HTTP–standards and design principles,” MMSys ’11 Proc. Second Annual ACM Conference on Multimedia Systems, pp.133–144, 2011.
[2] S.-C. Son, B.-T. Lee, Y.-W. Gwak, and J.-S. Nam, “Fast required bandwidth estimation technique for network adaptive streaming,” IEEE Trans. Consum. Electron., vol.56, no.3, pp.1442–1449, Aug. 2010.
[3] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, “Adaptive unicast video streaming with rateless codes and feedback,” IEEE Trans. Circuits Syst. Video Technol., vol.20, no.2, pp.275–285, Feb. 2010.
[4] T.C. Thang, Q.-D. Ho, J.W. Kang, and A.T. Pham, “Adaptive streaming of audiovisual content using MPEG DASH,” IEEE Trans. Consum. Electron., vol.58, no.1, pp.78–85, Feb. 2012.
[5] A. Albanese, J. Bloemer, J. Edmonds, and M. Luby, “Pri-ority encoding transmission,” IEEE Trans. Inf. Theory, vol.42, no.6, pp.1737–1747, Nov. 1996.
[6] U. Horn, K. Stuhlmuller, M. Link, and B. Girod, “Robust internet video transmission based on scalable coding and unequal error protection,” Signal Process. Image Commun., vol.15, no.1, pp.77–94, Sept. 1999.
[7] D. Wu, Y.T. Hou, W. Zhu, Y.-Q. Zhang, and J.M. Peha, “Stream-ing video over the Internet: Approaches and directions,” IEEE Trans. Circuits Syst. Video Technol., vol.11, no.3, pp.282–300, March 2001.
[8] M. Luby, “LT Codes,” Proc. 43rd Ann. IEEE Symp. Foundations of Computer Science, pp.271–282, 2002.
[9] D. Sejdinovic and D. Vukobratovic, “Expanding window fountain codes for unequal error protection,” IEEE Trans. Commun., vol.57, no.9, pp.2510–2516, Sept. 2009.
[10] C.-H. Wang, J.K. Zao, and H.-M. Chen, “A rateless UEP convolutional code for robust SVC/MGS wireless broadcasting,” IEEE International Symposium on Multimedia, pp.279–284, 2011.
[11] A. Talari and N. Rahnavard, “Distributed rateless codes with UEP property,” ISIT, pp.2453–2457, June 2010.
[12] K.-C. Yang and J.-S. Wang, “Unequal error protection for streaming

media based on rateless codes," IEEE Trans. Comput., vol.61, no.5, pp.666–675, May 2012.
[13] S.S. Arslan, P.C. Cosman, and L.B. Milstein, "Generalized unequal error protection LT codes for progressive data transmission," IEEE Trans. Image Process., vol.21, no.8, pp.3586–3597, Aug. 2012.
[14] C. Hellge, D. Gomez-Barquero, T.S. Milstein, and T. Wiegand, "Layer-aware forward error correction for mobile broadcast of layered media," IEEE Trans. Multimedia, vol.13, no.3, pp.551–562, June 2011.
[15] K. Kasai, D. Declercq, and K. Sakaniwa, "Fountain coding via multiplicatively repeated non-binary LDPC codes," IEEE Trans. Commun., vol.60, no.8, pp.2077–2083, Aug. 2012.

Yun Shen received B.E. degree in communication engineering from Beijing University of Posts and Telecommunications (BUPT), China, in 2010. He is currently a Ph.D. student in BUPT and working in Wireless Theories and Technologies (WT&T) Lab. His research interests include source/channel coding, multimedia communication, and QoE study on application services.

Yitong Liu received Master degree in communication engineering from BUPT, China, in 2007. She is currently a Ph.D. student in School of Information and Communication Engineering in BUPT and working in WT&T Lab. Her research interests include QoE of streaming service, quality and performance for wireless network, and wireless applications.



Jing Liu received B.E. degree in Electronic and Information Engineering from BUPT, China, in 2012. She is currently a Master student candidate in School of Information and Communication Engineering (SICE) in BUPT and working in WT&T Lab. Her main research interests include multimedia communication and video coding technology in streaming media transmissions.

Hongwen Yang is currently a professor in BUPT, and is the director of Wireless Communication Center of School of Information and Communication Engineering in BUPT. His research mainly concentrates on wireless physical aspects, including modulation and channel codes, MIMO, CDMA, OFDM, etc.

Dacheng Yang received Ph.D. degree in communication engineering from BUPT in 1988, and is currently a professor in BUPT. He is also the director of the WT&T Lab at BUPT. Since 1988, he has engaged in studies on communication systems. His recent interests are in wireless transmission techniques and systems.