Computers and Electrical Engineering 37 (2011) 768–781

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Robust transmission of scalable video stream using modified LT codes ☆

Ehsan Namjoo [a], Ali Aghagolzadeh [a,b,*], Javad Museviniya [a]

[a] *Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran*
[b] *Faculty of Electrical and Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran*

ARTICLE INFO

Article history:
Available online 23 May 2011

ABSTRACT

LT codes are convenient and popular kind of rateless codes that could easily tolerate different patterns of loss in erasure channels. In this paper an LT code with unequal packet protection (UPP) property is proposed. The proposed code could provide unequal packet recovery to any importance-sorted data packets. Simulation results indicate the enhanced performance of the suggested scheme and its ability to increase the probability of early decoding of more important parts of data rather than the rest. Also it is shown that the proposed scheme could provide comparable bit error rates in comparison to the one of the well known previous methods. The code with modified encoding graph has been utilized for transmitting of the scalable data-partitioned video stream. Simulation results also illustrate the performance of the suggested approach in early delivery of the most important parts of a video sequence.

1. Introduction

Recently the evolution of network communication has driven outstanding effects on human life. Recent advances in network technology have widely covered a large variety of communication issues, and tend to provide the most reliable and effective framework to communicate various data to diverse stations. A huge amount of the information flow in many networks is multimedia data. Also, transmission of video contents, which are mostly the output bitstream of video encoders, through erasure prone networks appears in many applications in real communicational networks. The diverse destination nodes necessarily do not have identical characteristics. Also they could not receive exactly the same media content. Therefore, scalable video coding seems a proper idea to feed the different receivers by contents with different spatial resolutions, frame rates or quality. A sub-bitstream could be extracted from a scalable bitstream by dropping some less important parts so that the resulting sub-stream still remains meaningful at the receiver's point of view. This property brings enough flexibility for different nodes to have their video content according to their playback capabilities.

Reliable transmission of video content through erasure channels and lossy networks is a vital issue in many applications and there are many transmission techniques to reliable delivery of the encapsulated video contents. Reed–Solomon (RS) codes have been utilized for packet protection of video streaming [1]. Network Working Group has attempted to specify internet standard tracks for the internet community to apply reliable and real-time forward error protection on RTP packets [2,3]. A detailed specification about how to put H.264/AVC video bitstream into RTP packets was presented in [4]. The same effort has been made for the scalable version of H.264/AVC, SVC (scalable video coding), in [5].

Since the encoding and decoding complexity of RS codes significantly increase for long packets, RS forward error protection could not make an appropriate choice. Protection of media packets by LT (Luby Transform) codes [6] provides a much better choice because in using LT codes, there is no need to employ time-consuming former RS methods that

☆ Reviews processed and approved for publication to Dr. Ferat Sahin.
* Corresponding author at: Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran. Tel.: +98 4113393720; fax: +98 4113300829.
E-mail addresses: e.namjoo@ieee.org (E. Namjoo), aghagol@ieee.org (A. Aghagolzadeh), niya@tabrizu.ac.ir (J. Museviniya).

0045-7906/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compeleceng.2011.04.011

may be impossible to apply to long video packets. LT codes are an important class of the probabilistic forward error correction (FEC) erasure codes called Fountain codes [7]. Whereas traditional erasure codes like RS codes have a fixed code rate that must be chosen before the encoding begins, Fountain codes are rateless as the encoder can generate as many encoded symbols as needed. This is an advantage for the channel with unknown or time-varying conditions. Using a fixed rate channel code would lead to either bandwidth wasting if the erasure rate is overestimated or poor performance if the erasure rate is underestimated. The encoder and decoder for LT codes are not as complicated as those for RS codes. Coding and decoding complexity for LT codes increases linearly by growing the packet length. In addition, any subset of adequate coded symbols could be chosen to recover the whole original source. The later property provides a sort of flexibility that makes the code robust against different patterns of packet loss. It means that independent of the channel's loss pattern, it is just enough for the receiver to receive only a specific number of source packets to successfully recover the whole coded message.

In transmitting data with different importance hierarchy, like data-partitioned video bit streams, the protection scenario should be considered in a way that more important parts get higher order of protection. Codes that protect some part of data better than the other parts are called unequal error protection (UEP) codes. In contrast to equal error protection (EEP) codes where the same level of forward error correction is applied to all information symbols, UEP codes assign different levels of protection to the different information symbols. UEP has been already used to protect the scalable image data and also for video transmission. A detailed review of FEC techniques for the scalable image coders is presented in [8]. It surveys recent progress made in system design and discusses efficient source-channel bit allocation techniques, with emphasis on UEP. Also an UEP video transmission scheme based on the dynamic resource allocation in a MIMO-OFDM system is presented in [9] for improving the video transmission performance over the wideband wireless channels. As mentioned, independent of the packet loss pattern, LT codes' decoder can recover the whole source data as soon as it receives adequate number of coded packets. Therefore, in a loss prone channel that the source data are to be protected using LT codes, UEP scenario is considered as the ability of the code to recover the high important or priority data sooner than the other parts.

In this paper, first, a modification on LT code's encoding graph is performed whereby the input nodes in encoding graph are sorted according to their degrees. The resultant code is able to provide different levels of protection to source data packets. The protection strength would be maximized for the packets that correspond to the maximum degree input nodes at the beginning of the graph; and the protection strength rhythm is decreased by moving toward the end. The modified code is used for robust transmission of the scalable video stream. For this purpose, a new arrangement of data packets, as a data partitioning approach based on the importance of video packets of SVC bit stream, is applied. In data-partitioned stream, video packets are arranged according to their relative importance. The outstanding property of the proposed code is that it provides a condition under it the beginning parts of data have more chance to be decoded earlier. In general, packets closer to the beginning of the stream have more chance to be decoded earlier; so applying the code to the data-partitioned SVC video packet provides unequal packet recovery property for the video packets. Simulation results will show the effectiveness of the proposed approach.

The remaining of the paper is as follows. To make it self-contained, the most related and essential materials to our work are given in Sections 2 and 3. Section 2 briefly skims SVC and NAL layer specifications of the new scalable extension of H.264 video codec. Section 3 covers the essential basis of LT codes, a review of a well known UEP fountain code, followed by the proposed modification that is made on the LT code's encoding graph to provide unequal packet recovery. The simulation results that show the effectiveness of the proposed scheme are also given in this section. In Section 4, the concept of SVC quality layers and the extraction rules to derive reasonable sub-streams is presented and the modified LT code is applied to the data-partitioned SVC bitstream to provide reliable unequal packet recovery for the video bit stream. Finally Section 5 concludes the paper.

2. SVC and NAL layer specifications

Scalability for the first time was embedded in MPEG2 [10] codec by means of some rarely used profiles. MPEG2 was a very successful standard widely used in different applications like satellite broadcasting, cable TV, and DVD storage and play back systems. However, the scalability provided by MPEG2 was not so efficient, so it couldn't attract enough attention as a real demand at the time of release. Also, the other subsequent standards failed to be efficient enough for scalability purposes. The most important reason for being unprofitable may be the fact that using spatial and quality scalability features, came along with a significant loss in coding efficiency as well as a large increase in decoder's complexity.

By recent advances in video coding technology, the new scalable video coding standard [11], SVC, that was built based on the pervious well-known standard, H.264/AVC [12], has fabulously refined the pervious video coding standards and introduced effective tools such as inter-layer prediction to solve the inefficiency of the former endeavors [13]. In addition to H264/AVC features, the scalable extension efficiently supports temporal, spatial and SNR scalability [14].

Similar to H.264/AVC, SVC consists of two parts: Video Coding Layer (VCL) and Network Abstraction Layer (NAL). VCL is responsible to all coding procedures and could be considered as the heart of the SVC codec. Operations like motion prediction and compensation, transform coding and quantization are accomplished in VCL. NAL organizes the output of VCL layer in packets called NAL-unit. Each NAL-unit in SVC bitstream has different importance depending on which layer it belongs to [15].

SVC is a highly suitable solution to the situations in which the characteristics of receivers, specially their play-back capabilities, are different. A scalable bitstream refers to the sequence of coded video content that intelligent removal of some of its parts provides another feasible and valid sub-stream sequence of NAL unit that may be used to feed the other receiver with some restricted capabilities. A bit stream that doesn't have this property is called single-layer stream. A scalable bit stream consists of a base layer and one or more enhancement layers. Base layer is the most important part of the stream and contains the essential parts that correspond to the reduced version of the original content. Enhancement layers, dependent on the mode of scalability, may increase the frame rate (temporal scalability), resolution (spatial scalability) or quality (SNR scalability) of the base layer.

A set of consecutive NAL units in an original SVC sequence with the specific properties is referred to as an access unit (AU). The decoding of an AU results in an exactly one decoded picture. Also, a set of consecutive AUs with certain properties is referred to as a coded video sequence. A coded video sequence represents an independently decodable part of the bit stream. It always starts with an instantaneous decoding refresh (IDR) AU, which signals that the IDR-AU and all of the following AUs can be decoded without decoding any previous pictures of the bit stream [2].

Temporal scalability describes a case in which the sub-stream presents a video content with reduced frame rate. A bit stream provides temporal scalability when the set of the corresponding AUs can be partitioned into a temporal base layer and one or more temporal enhancement layers. Let temporal layers be identified by a *temporal identifier*, *T*, which is a non-negative integer, starts from 0 for the base layer and increases by one from one enhancement layer to the next one. For any natural number, *K*, a bit stream derived from the original sequence by discarding all AU with temporal layer identifier, *T*, greater than *K*, form another valid bit stream. Temporal scalability can be enabled by restricting the motion-compensated prediction to the reference pictures with temporal layer identifier less than or equal to the temporal layer identifier of the picture to be predicted. Fig. 1 shows the hierarchical prediction structure for GOP of size 8. The set of pictures between two successive temporal base layer pictures including a base layer picture is called group of pictures (GOP). The arrows in Fig. 1 indicate the direction of the motion prediction.

In spatial scalability each layer corresponds to a spatial resolution and is identified by a *dependency identifier, D*. The base layer provides the lowest spatial resolution with dependency identifier equal to 0. The dependency identifier increases by one from one spatial layer to the next one. The spatial resolution of the content is increased by adding the spatial enhancement layers to the base layer. In each spatial layer, the motion-compensated prediction and intra-prediction are employed as for a single-layer coding; but in order to improve the coding efficiency, the SVC encoder utilizes a mechanism called inter-layer prediction. The main goal of designing the inter-layer prediction is to enable the using information from the lower layers as much as possible to improve the rate-distortion efficiency of the enhancement layers. Fig. 2 shows the coding structure when there are two spatial representations. With different spatial representations for a given time instant, AUs should be transmitted in increasing order according to their spatial layer identifiers, *D*. Also, there is no need for the lower layers to present in all AUs which make it possible to combine the spatial and temporal scalabilities as illustrated in Fig. 2.

SNR or quality scalability can be considered as a special case of the spatial scalability in which the spatial resolution of the base layer and the enhancement layers are the same. In this mode of scalability, adding the enhancement layers improves the quality of the base layer. In SVC, two different possibilities exist for providing SNR scalable bit-streams. The first one is referred to as coarse-grain scalability (CGS) and the second one is referred to as medium-grain scalability (MGS). In CGS, similar to the spatial scalable coding, only a limited set of the extractable rate points is included in the bitstream. In contrast to CGS coding, MGS supports many more extractable rate points. Furthermore, for MGS, an encoder may choose to partition the transform coefficients of a layer into up to 16 MGS layers which increases the number of packets and also the number of possible extractable sub-streams. In this way, a finer granularity is achieved [16]. More information about this mode and the details about the scalable extension of H.264/AVC can be found in [14].

NAL units in H.264/AVC consist of one byte header at the beginning followed by a varying size payload. The structure of a NAL unit's header in SVC is the same except for prefix NAL units and SVC coded slice NAL units (types 14 and 20). The structure of a NAL unit's header in SVC is depicted in Fig. 3 [5,15]. In this structure, each NAL unit consists of four-byte header followed by varying length payload information. The first byte of this structure is exactly the same as the H.264/AVC NAL unit header. It contains three syntax elements of 1, 2, and 5 bits, respectively: forbidden zero bit (F), nal_ref_idc (NRI),

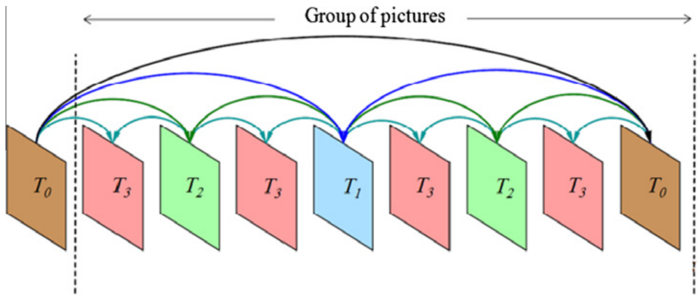

Fig. 1. Hierarchical prediction structure for GOP of size 8 [16].

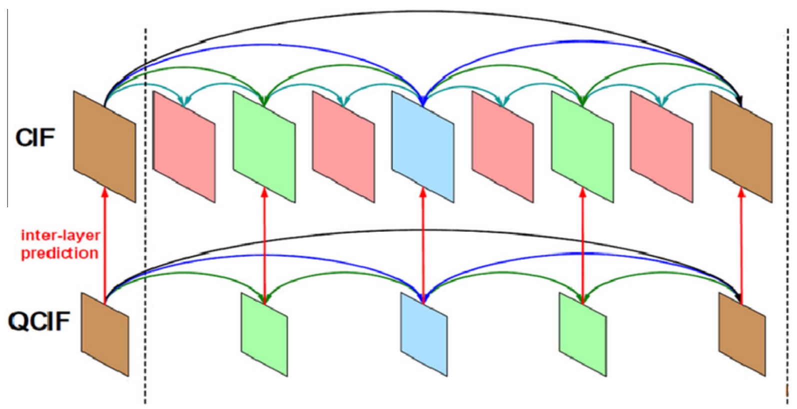

Fig. 2. Coding structure with two spatial layers [16].

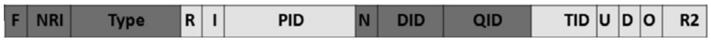

Fig. 3. 4-Byte SVC NAL unit header structure.

and nal_unit_type (NUT). The value of forbidden zero bit must be zero in a conforming bit streams. A value of 1 may be used to indicate that the NAL unit is corrupted, but it may make sense for an error-resilient decoder implementation to attempt to decode it. The value of nal_ref_idc indicates the relevance of the NAL unit for reconstruction of the sample values. Nonzero values of nal_ref_idc must be used for coded slice and slice data partition NAL units of the reference pictures as well as for the parameter set NAL units. The value of nal_ref_idc must be equal to 0 for slices and slice data partitions of the non_reference pictures and for NAL units that do not affect the reconstruction of the sample values, such as SEI (Supplemental Enhancement Information) NAL units which have extra information that can be inserted into the bitstream to enhance the use of the video for a wide variety of purposes. External specifications may specify the allocation or semantics of the nonzero values of nal_ref_idc. For example, nal_ref_idc indicates the relative transmission priority of the NAL unit according to the RTP payload format for H.264/AVC [4]. H.264/AVC specifies a number of different NAL unit payload structures, and the value of nal_unit_type indicates which payload structure is in use. Bytes 2 to 4 of the SVC NAL unit header consist of the following syntax elements: R is a reserved bit for next extension. Idr_flag (I) indicates weather the layer picture is an IDR picture (when equal to 1) or not (when equal to 0). Priority_id (PID) specifies a priority identifier for the NAL unit. The lower value of priority_id indicates the higher priority. no_inter_layer_pred_flag (N) specifies, when presents in the coded slice NAL unit, weather inter-layer prediction may be used for decoding the coded slice (when equal to 1) or not (when equal to 0). Dependency_id (DID) denotes the inter-layer coding dependency hierarchy. At any AU, a layer picture with a less dependency_id may be used for the inter-layer prediction for coding of a layer picture with a greater dependency_id, while a layer picture with a greater dependency_id shall not be used for the inter-layer prediction for coding of a layer picture with a less dependency_id. Quality_id (QID) designates the quality level hierarchy of a MGS layer picture. For all AUs with identical dependency_id values, a layer picture with quality_id equal to q_l uses a layer picture with quality_id equal to q_l-1 for inter-layer prediction. Temporal_id (TID) indicates the temporal layer (or frame rate) hierarchy. Informally, a layer consisted of pictures with less values of temporal_id has a lower frame rate. A given temporal layer typically depends on the lower temporal layers (i.e., the temporal layers with less temporal_id) but never depends on any higher temporal layer. A value of use_ref_base_pic_flag (U) equal to 1 indicates that only reference base pictures are used during the inter layer prediction process and a value of 0 indicates that the reference base pictures are not used during the inter layer prediction process. Value of 1 for discardable_flag (D) indicates that the current NAL units of the current AU is not used for decoding NAL units with values of dependency_id higher than the one of the current NAL unit in the current and all subsequent AUs. Such NAL units can be discarded without risking the integrity of the higher layers with greater dependency_id. Value of 0 for discardable_flag (D) indicates that the decoding of the NAL unit is required to maintain the integrity of the higher layers with greater dependency_id. Output_flag (O) affects the decoded picture output process as defined in annex C of SVC; and finally R2 is consist of two bits that are reserved for the future extension [5].

3. Methods and materials

In this section, a brief preview of LT codes is given and a well known approach to provide unequal error property on LT codes is skimmed. Then, a modification to LT code's encoding graph is proposed that provides unequal error protection for the LT code. Comparisons based on the simulation results show the performance of our proposed scheme.

3.1. Essentials of LT codes

LT code utilizes sparse bipartite graph to generate codewords. The most distinguishing characteristic of an LT code is that it employs a simple algorithm based on the XOR (exclusive OR) operation to encode and decode messages. The encoding process begins by dividing the message into K blocks of equal length, each block is considered as an input symbol; then, a random number, say d, $1 \leqslant d \leqslant K$, ($K$ is the number of input symbols) is generated by a random number generator function called robust Soliton distribution (RSD). Then a subset of d input symbols out of K input symbols is randomly selected. An output symbol is generated by applying XOR operation to the elements of the selected subset. In the same way, as many output symbols as required are generated; that's why an LT code is considered as a kind of rateless codes [7]. The decoder can retrieve all input symbols if it receives enough, often a bit more than K, encoded symbols. The encoding graph consists of two parts: the input symbol part with K input nodes and the second part that consists of all generated codewords (output nodes). Fig. 4 indicates a very simple encoding graph. Graph's edges merge into an output node from the input selected nodes. Each of the input and output nodes could be bits, or in general case, equal size packets of bits.

The decoding of LT codes is performed according to the belief propagation algorithm. The decoding process is as follows [17]:

Find an input node t_n which is connected to only one source packet s_k (if there is no such check node, the decoding process halts at this point and fails to recover all the source packets), and then

i. Set $s_k = t_n$.
ii. Add (operate binary XOR) s_k to all output nodes that are connected to s_k.
iii. Remove all the edges connected to the source packet s_k.
iv. Repeat (i–iii) until all the K input symbols are determined.

3.2. Output nodes' degree distribution

The heart of an LT code is its output nodes' degree generator function. The degree generator function should be designed in a way that causes small decoding error probability δ. Luby has shown that for a universal decodable LT code, the average degree of the output nodes should be $\ln(K)$ and also the average number of XOR operations needed to decode a message is $K\ln(K)$. To keep these conditions satisfied, he suggested a degree distribution function $\rho(d)$ called Soliton distribution [6]. The suggested degree distribution function is as follows:

$$\rho(d=1) = 1/K, \quad \rho(d) = \frac{1}{d(d-1)} \quad \textit{for} \quad d = 2,3,\ldots,K \tag{1}$$

Although the Eq. (1) satisfies the conditions theoretically, but it causes to increase the probability of decoding failure. To solve the problem, the robust Soliton distribution was proposed as Eq. (2) [6]:

$$\mu(d) = (\rho(d) + \tau(d))/Z \tag{2}$$

where $\tau(d)$ and Z are given in Eqs. (3) and (4), respectively:

$$\tau(d) = \begin{cases} S/Kd, & \textit{for} \quad d = 1,2,\ldots,(K/S)-1 \\ \frac{S}{K}\ln(S/\delta) & \textit{for} \quad d = K/S \\ 0 & \textit{for} \quad d > K/S \end{cases} \tag{3}$$

$$Z = \sum_d \rho(d) + \tau(d) \tag{4}$$

S is the number of degree-one output symbols and a very important parameter called ripple size. Ripple size is given as Eq. (5):

$$S = c \cdot \ln(K/\delta)\sqrt{K}. \tag{5}$$

Where, c is a constant real value in the range [0 1]. The ripple size should not be so large; otherwise, the number of overhead packets required to reconstruct the whole message is increased. On the other hand, small values for ripple size increase the decoding failure probability; it means that in some decoding step there might be no other degree-one output symbol to carry on decoding procedure. Robust Soliton function for the values $K = 10000$, $c = 0.2$ and $\delta = 0.05$ is illustrated in Fig. 5.

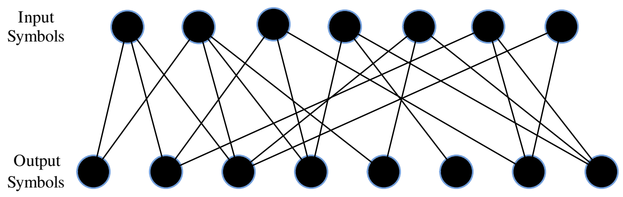

Fig. 4. Simple example of encoding graph.

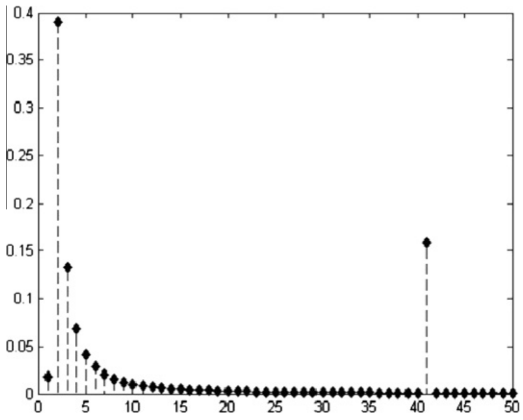

Fig. 5. Robust soliton probability mass function.

Luby's analysis explains how designing τ and ρ ensures that the decoding process gets started. Spike shown in Fig. 5 at $d = K/S$ ensures that every source packet is likely to be connected to a check node, at least once.

3.3. Rateless codes with unequal error protection property

A rateless code with unequal error protection property was proposed in [18]. Supposing that there are K input symbols, the input symbols are partitioned into r sets $s_1, s_2, \ldots, s_r$ of size $\alpha_1 K, \alpha_2 K, \ldots, \alpha_r K$ such that $\sum_{i=1}^{r} \alpha_i = 1$. Let $P_i(K)$ is the probability that an edge is connected to a particular input node in s_i for $i = 1, 2, \ldots, r$. It is clear that $\sum_{i=1}^{r} \alpha_i K P_i(K) = 1$. Assuming that $P_i(K) = K_i/K$, the aforementioned equation can be written as $\sum_{i=1}^{r} \alpha_i K_i = 1$.

The encoding procedure for a sequence of n output symbols is as follows:

i. Select randomly a value for degree d_n, $d_n \in \{1, 2, \ldots, K\}$, according to robust soliton function.
ii. Select d_n distinct input symbols. To select a symbol, at first select one of the r predefined sets, s_i, and then uniformly choose one of the input symbols covered by s_i.
iii. Set an output node equal to bitwise module 2 sum of the selected input nodes.

As mentioned in [18], to ensure that the average bit error rate for the input symbols covers by s_i is smaller than the average bit error rate of those from any set s_j, the probability of choosing any input symbol from s_i should be greater than the probability of choosing the input nodes from s_j.

Assuming that there are just two sets s_1 and s_2, the set s_1 is called the set of most important symbols (MIS) and the set s_2 is called the set of least important symbols (LIS). In this case, deciding the probability of choosing any symbol from s_1 greater than the probability of choosing that from s_2, ensures that the average bit error rate for the symbols covered by set s_1 is smaller than the average bit error rate for elements of the set s_2 at the decoder side. To achieved this, $P_1(K)$ is set to K_M/K and $P_2(K)$ is set to K_L/K where $K_M = \frac{1-(1-\alpha)K_L}{\alpha}$ and $0 < K_L < 1$. The parameter K_M indicates the relative importance of the symbols from s_1.

3.4. Sorted encoding graph LT code

In many studies on rateless codes, EEP of all parts of data is considered. The EEP property would be appropriate for applications such as multicasting bulk data (e.g., a software file) [7]. In some other applications, a portion of data may need to be recovered prior to the rest. An example could be video-on-demand systems, in which the stream should be reconstructed in sequence [19,20]. The protection technique, in such scenarios, is often performed using UEP codes. These codes can also be employed in applications for which *unequal recovery time* (URT) is desirable, i.e., the number of received symbols for recovering more important parts is less than that number for recovering less important parts. In this paper, we propose a change in the LT code's encoding graph, whereby the more important parts of data stream get more chance to be decoded earlier than the other parts. From now on, we refer to this property as *unequal packet recovery* (UPR).

Most of the modern CODECs have the optional data partitioning modes to improve the error resiliency in which the video macroblock header data, the motion vector data, and DCT coefficients data are separated into their separated data partitions. For example, in MPEG-4 (visual simple profile), there is an optional data partitioning mode. When this mode is enabled, the video macroblock header and motion vector partitions, which are much more important to the quality of the video reconstruction, are transmitted in the partitions at the beginning of the video stream, while residual DCT coefficient partitions, which are less important, are transmitted in the partitions closer to the end of the stream. Since the data is arranged in descending order of importance, it would be beneficial to provide more protection to the beginning part of the stream in transmission [2].

In encoding graph, the input nodes with larger degree are more likely to be decoded earlier. This is due to the fact that the input nodes with larger degree are more likely to be connected to a degree-one output node before the decoding procedure gets started. Also, if there is no such a degree-one output node, still there is chance for nodes with larger degree to be released by decoding the other input nodes very soon. Considering this fact, a modification to the encoding graph is proposed that increases the probability of early decoding of more important parts of data.

The output nodes' degrees are generated by the robust Soliton distribution and input nodes are randomly selected. Thus, by considering the encoding graph unchanged throughout the whole encoding and decoding procedure, changing the input nodes' arrangement by sorting them according to their degrees could provide UPR property for the input symbols. The input symbols with the most degree appear first followed by the other input symbols based on their degrees. The last input node would be the one with the least degree. Fig. 6a shows a simplified example of the encoding graph in its original form and Fig. 6b indicates the situation that the input nodes have been sorted by their degrees.

Considering importance-sorted data as the input nodes provides UPR for the modified graph of the proposed code. This is equivalent to the situation that more important data packets are placed on the position of the input nodes with large degree in the unsorted original encoding graph. Since the input nodes in the modified encoding graph are sorted by their degrees, by considering more important packets as the input nodes at the beginning of the encoding graph, the chance of early decoding of them is increased. Thus by assumption that the importance of source data increases by moving towards the beginning, the proposed idea could provide UPR property for any importance-sorted sequence. The superiority of utilizing the modified encoding graph over the original one can be assessed by employing it for a stream of packets that the first part of stream is considered to be more important than the second part. Comparing the modified encoding graph with the unchanged original graph shows the effectiveness of the proposed approach. The performance criterion can be simply defined as the ratio of the decoded high priority packets to the number of total high priority packets at each step of the decoding procedure (iteration). For simplicity, we call it *performance ratio*. Percentage of the performance ratio at a specific iteration may be presented as Eq. (6):

$$\text{Performance ratio} = \frac{\text{the number of decoded high priority packets}}{\text{total number of high priority source packets}} \tag{6}$$

Figs. 7–9 show the performance ratio curves for all iterations in the case that the number of total source packets, K, is considered to be 3000. The numbers of the most important packets (high priority packets) at the beginning of the stream have been assumed to be 200, 400 and 600, respectively. The values for two parameters δ and c are set as $\delta = 0.01$ and $c = 0.12$.

According to the results shown in Figs. 7–9, at any iteration, the performance ratio curves for the proposed modified graph is above the case with the original encoding graph. In random base view of the code, performance ratio doesn't follow a smooth and predictable rhythm; for example, in Fig. 7 while the number of more important packets has assumed to be 200, but the gap between the performance ratio curves, in some regions, is smaller comparing with Fig. 8 where the number of important packets has assumed to be 400. The gap gets tighter by choosing the number of high priority packets equal to 600 in Fig. 9 that seems reasonable. Again putting in account the random nature of the code, further computation of the performance ratio will cause similar but not exactly the same curves.

As an advantage, the proposed method could make an appropriate choice and also a suitable match for the source materials that have a decreasing (or increasing) rhythm of importance in their source packets. As an outstanding example of such materials, we could consider the data-partitioned video packets.

The effectiveness of the proposed approach could also be assessed in comparison with a well known pervious work that was done by Rahnavard and Vellambi [18]. The basis of the approach was given in Section 3.3. From now on we refer to it as Rahnavard unequal error protection scheme (RUEP). The comparison is made by using bit error rate (BER) curves versus the overhead values for simple LT code, RUEP and our proposed approach. It is assumed that there are two sets of the input nodes: s_1 at the beginning of the stream, considered as the set of the most important symbols, and s_2 covers the remaining

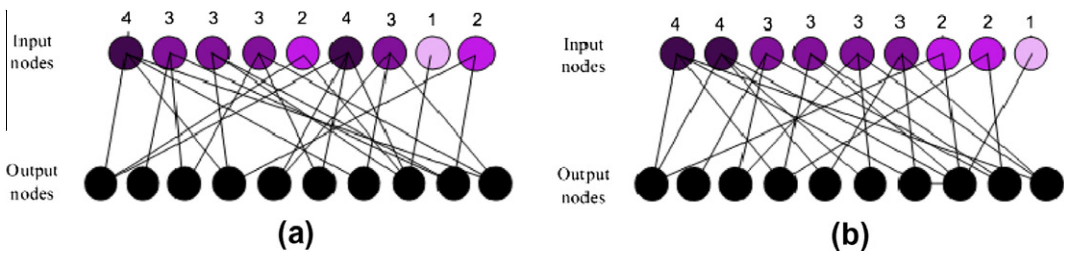

Fig. 6. Encoding graphs, (a) the original form and (b) the modified form.

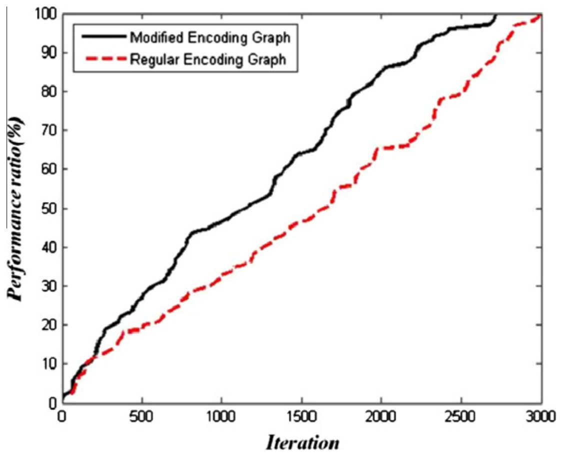

Fig. 7. Performance ratio curves (number of more important packet is 200).

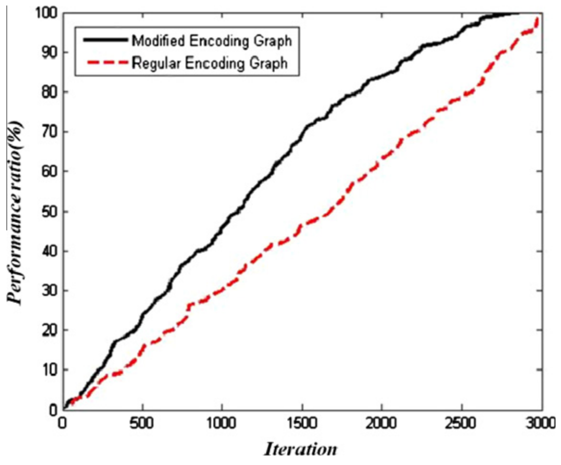

Fig. 8. Performance ratio curves (number of more important packet is 400).

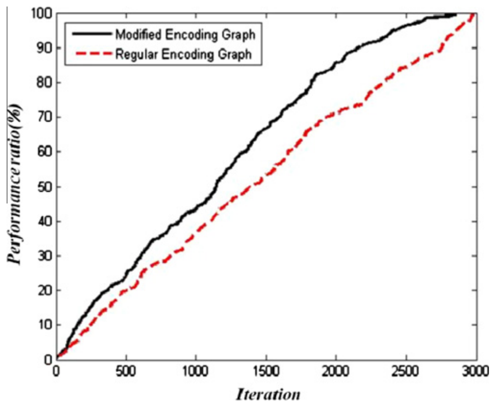

Fig. 9. Performance ratio curves (number of more important packet is 600).

least important symbols. The value of overhead is changed and, each time, the proportion of the input source nodes that could not be recovered at the decoder side is computed for each set and considered as the bit error rate.

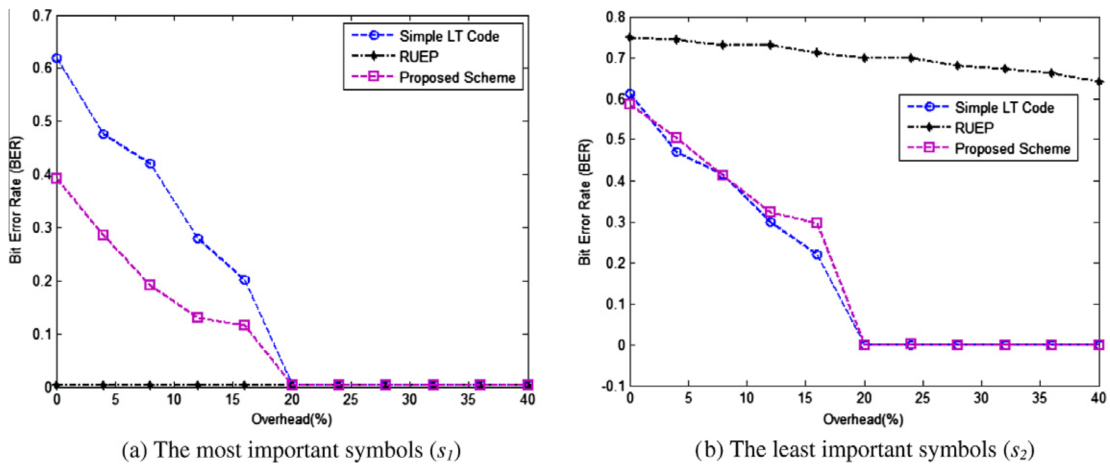

Fig. 10. Bit error rate (BER) curves for $K = 2500$, $\delta = 0.05$, $c = 0.2$, $\alpha = 0.1$, and $K_M = 9.1$.

Fig. 10 shows the BER curves for the sets s_1 and s_2 in the case that K_M, the relative importance of symbols in set s_1 for RUEP approach, is considered equal to 9.1, K is equal to 2500, α, the proportion of the most important nodes is also considered equal to 0.1. As shown in Fig. 10a, the BER curve for the proposed method is below the same curve for the simple LT code that ensures the UEP property for the proposed code. The RUEP's bit error rate curve for s_1 is below the similar curves for the proposed approach and the simple LT approach. Actually, the performance of RUEP to protect the most important part of data is marvelously better than the other approaches. But according to the results shown in Fig. 10b, the protection performance over s_2, when using RUEP is not as satisfactory as it is for the proposed method and the simple LT; or we could say that it is pretty poor. So the proposed method not only presents pretty good protection performance over the most important parts rather than the least important parts, but also it ensures the perfect reconstruction of the whole data in the case that sufficient overhead symbol is considered. The RUEP fails to do so even when much greater value of overhead is available. The same simulation results have been shown in Fig. 11. The parameters are the same except for K that is set to 5000. In this case, the performance of error protection of the proposed scheme over set s_1 is enhanced in comparison with the previous case shown in Fig. 10a; but again RUEP presents better performance over s_1 and worse performance over s_2, whilst the proposed method presents a satisfactory performance over both. The same simulation results for $K = 10{,}000$ have been shown in Fig. 12. By increasing the number of input symbols to $K = 10{,}000$, the same results are achieved, except that the BER performance over s_1 for the proposed method is enhanced in comparison with the previous cases while the performance of RUEP over s_2 gets worse. The proposed scheme could reconstruct the whole input nodes when the overhead is chosen to some specific value, whilst the RUEP could not do so even using greater values of the overhead.

The results shown in Fig. 10–12 can be justified by the following reasoning. RUEP encoding procedure provides a condition that the probability of choosing the most important parts of data increases and, as a result, the most important parts get

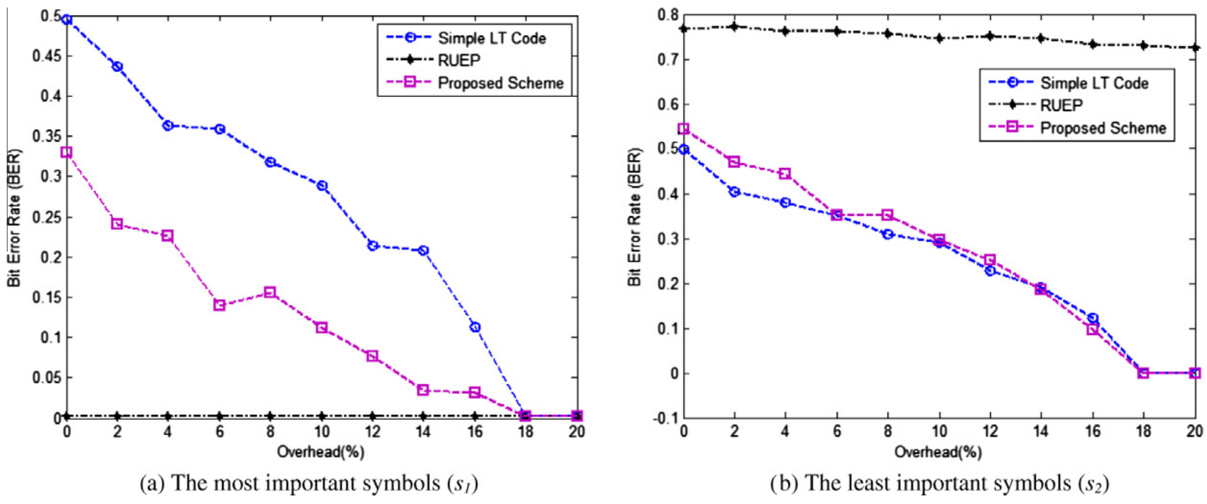

Fig. 11. Bit error rate (BER) curves for $K = 5000$, $\delta = 0.05$, $c = 0.2$, $\alpha = 0.1$, and $K_M = 9.1$.

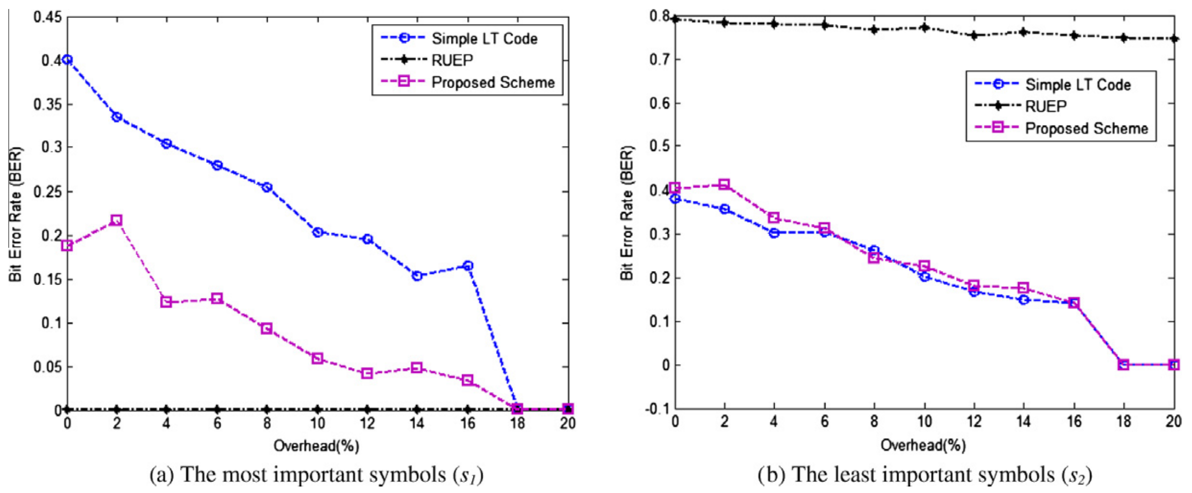

Fig. 12. Bit error rate (BER) curves for $K = 10000$, $\delta = 0.05$, $c = 0.2$, $\alpha = 0.1$, and $K_M = 9.1$.

more protection by dedicating the chance of choosing symbols from s_1 more than that from s_2. Actually, the symbols from s_1 get more protection by decreasing the chance of choosing the symbols from s_2. In other words, the encoder sacrifices the protection performance of symbols from s_2 to provide more protection on symbols of s_1; and that's why the RUEP performance over set s_1 is fabulously good. In the proposed scheme, just by sorting the input nodes in the encoding graph, the principals of the encoding procedure keeps unchanged so that the BER for symbols from s_1 increases, while the BER for elements of s_2 does not differ so much in comparison with the simple LT code. Also, since the structure of the proposed code is similar to the simple LT code, so by increasing the number of input symbols, the ability of perfect reconstruction of the input symbols increases just like the simple LT code. Concurrently, RUEP could do so only with very large values of overhead that does not seem so practical. That means the proposed code will be able to reconstruct the whole input source data by using much smaller values of overhead. In next section, the proposed code is used for transmitting the scalable video stream.

4. Transmitting SVC packets using proposed modified graph LT codes

A sub-stream could be extracted from a scalable bitstream by considering some restrictions. The standard sets the following constraints on the sub-stream that is to be extracted: Let *pId* be any value larger than or equal to *Pmin*, where *Pmin* is the minimum value of priority_id in the coded video sequence. Let *dId* be any value larger than or equal to *Dmin*, where *Dmin* is the minimum value of dependency_id in the coded video sequence. Let *tId* be any value larger than or equal to *Tmin*, where *Tmin* is the minimum value of temporal_id in the coded video sequence. The bitstream obtained by discarding all NAL units with priority_id greater than *pId* or dependency_id greater than *dId* or temporal_id greater than *tId* could make a conforming sub-stream [21]. Also according to the restrictions about assigning the syntax element priority_id in [22], in NAL units that belong to the same dependency layer and the same temporal layer, priority_id is assigned so that the NAL units with smaller values of quality_id get higher priority.

By considering the above restrictions, a video sequence in CIF resolution, BUS [23], is encoded using JSVM software [24] in MGS scalable mode. The overall frame rate of the original video is 30 frames per second and it contains 150 frames. The software parameters are set as GOP = 16 and the number of MGS layer is set to 7. The distribution of transform coefficients among the quality slices of the quality layers is determined by a vector called MGSVector. It is an adjustable parameter. Its elements should be positive integers and also the overall sum of the elements of the vector must be equal to 16. The *i*th element of the MGSVector shows the proportion of transform coefficients that contribute to the *i*th MGS level. The MGSVector is set as Eq. (7).

$$MGSVector = [3\,3\,2\,2\,2\,2\,2] \tag{7}$$

According to JSVM software manual, for rate-distortion efficient MGS coding, the values MeQP that are used for determining the Lagrangian parameters should be set to a value smaller than the actual quantization parameter. Here, for base layer, the values MeQP are set to 30 and the actual value of quantization parameter is set to 32. For all MGS layers, the values MeQP are set to 24 and the actual value of quantization parameter is set to 26. The other parameters remain unchanged and set to the software default values.

After running JSVM encoder, according to the restrictions stated above, 40 sub-streams could be extracted from the coded SVC bitstream. At this mode, all the NAL units belong to the same layer representation (CIF), with the same parameter for dependency_id equal to 0. All the extracted sub-streams could be decoded by JSVM decoder. Table 1 shows the corresponding bit rates (Kb/s) and PSNR (dB) values of the decoded sub-streams. The first row of Table 1 corresponds to the base layer

Table 1
Extractable sub-streams.

	$T = 0$ Frame rate = 1.875	$T = 1$ Frame rate = 3.75	$T = 2$ Frame rate = 7.5	$T = 3$ Frame rate =15	$T = 4$ Frame rate = 30
$Q = 0$	Rate = 19.0394	Rate = 51.8388	Rate = 130.0847	Rate =325.0304	Rate = 796.6400
	PSNR = 37.7403	PSNR = 36.0735	PSNR = 34.8742	PSNR = 34.0182	PSNR = 33.3614
$Q = 1$	Rate = 23.9253	Rate = 68.4620	Rate = 183.0067	Rate = 485.9416	Rate = 1252.5904
	PSNR = 38.7792	PSNR = 36.1549	PSNR = 34.7328	PSNR = 33.8031	PSNR = 33.1734
$Q = 2$	Rate = 26.3027	Rate = 76.2559	Rate = 206.5188	Rate = 555.9512	Rate = 1452.6368
	PSNR = 39.3617	PSNR = 37.1907	PSNR = 35.9689	PSNR = 35.1298	PSNR = 34.5003
$Q = 3$	Rate = 28.3655	Rate = 82.6538	Rate = 225.2755	Rate = 611,1008	Rate = 1608.8752
	PSNR = 39.9506	PSNR = 37.9750	PSNR = 36.8091	PSNR = 35.9796	PSNR = 35.3050
$Q = 4$	Rate = 30.2760	Rate = 88.8639	Rate = 244.6176	Rate = 669.7312	Rate = 1773.7472
	PSNR =40.6408	PSNR = 38.8124	PSNR = 37.6788	PSNR = 36.8410	PSNR = 36.0896
$Q = 5$	Rate = 31.8755	Rate = 93.7464	Rate = 258.6584	Rate = 709.7904	Rate = 1884.4992
	PSNR = 41.2789	PSNR = 39.4347	PSNR = 38.2516	PSNR = 37.3570	PSNR = 36.5299
$Q = 6$	Rate = 33.3358	Rate = 97.9028	Rate = 269.8847	Rate = 741.1168	Rate = 1971.8736
	PSNR = 41.9014	PSNR = 39.9802	PSNR =38.7095	PSNR = 37.7465	PSNR = 36.8483
$Q = 7$	Rate = 34.1916	Rate = 100.2789	Rate = 276.3051	Rate = 759.2264	Rate =2024.4144
	PSNR = 42.1995	PSNR = 40.1967	PSNR = 38.8659	PSNR = 37.8637	PSNR = 36.9346

that temporal_id values are chosen from 0 to 4. Temporal_id (T) equals to 0 corresponds to the NAL units that form the lowest and the most important temporal layer pictures and also provide the lowest frame rate 1.875 frames per second. The values 1, 2, 3 and 4 correspond to NAL units that provide frame rates 3.75, 7.5, 15, and 30 frames per second, respectively. In SVC, the temporal scalability mode completely depends on the parameter GOP. Choosing GOP equal to 16 provides the flexibility for the encoder to manage the motion compensation mechanism in a way that there are up to five hierarchical temporal layers. The mechanism is the same as shown in Figs. 1 and 2. So, regardless of the scalability mode, weather spatial or SNR, stream is always temporally scalable. The Second row of Table 1 indicates the extractable rates and their corresponding PSNR values for the first quality layer. The set of extracted NAL units corresponding to the second row of Table 1 includes all NAL units of the base layer. In MGS scalable coding, any higher layer has all the NAL units of the lower layer.

A data-partitioning scheme is proposed so that the following conditions are satisfied. First, all NAL units in data-partitioned stream should be independent of the following ones. This condition minimizes the decoding delay and also forces the more important NAL units to be placed at the beginning of the data-partitioned sequence for better protection. Second, the NAL units belong to the quality layer L with different values of temporal_id should appear in data-partitioned stream before the NAL units that belong to the quality layer L + 1. Also, in each quality layer, the NAL units corresponding to the smaller values of tempotal_id should appear before those with greater values of temporal_id. To satisfy the mentioned conditions, the base layer NAL units (all NAL units with quality_id equal to 0) are placed in the sequence according to their temporal_id identifiers. So, the NAL units with temporal_id equal to 0, which are mostly IDR pictures, are positioned at the beginning of the sequence. Then, the first quality layer, the sequence of NAL units with quality_id equal to 1 arranged according to their temporal_id identifiers, is followed. This procedure is similarly done for the other quality layers. By this arrangement, not only the importance of each layer is more than the successive one, but also in each layer itself the NAL units are arranged in a way that those with smaller values of temporal_id (mostly slice pictures used as the reference picture for motion compensation mechanism) are placed before the others. Assuming n quality layers, Fig. 13 indicates a general view of the proposed data-partitioned stream. Fig. 14 shows the internal arrangement of the quality layer *i* for the special case of GOP = 16.

The generated data-partitioned sequence is fed to the modified encoding graph as indicated in Fig. 15. Performance ratio for a specific layer L in data-partitioned sequence can be defined as the ratio of the number of decoded packets of layer L to the number of all packets in layer L at each step of the decoding procedure. Percentage of performance ratio for a specific layer L at a specific iteration is given as Eq. (8):

$$\text{Performance ratio for layer L} = \frac{\text{the number of decoded packets of layer L}}{\text{total number of packets in layer L}} \tag{8}$$

The encoding procedure for BUS sequence in MGS scalable mode with the mentioned parameters produces 1355 NAL units. So the number of input nodes in the encoding graph, *K*, is 1355. The LT code's parameters are set as $\delta = 0.01$ and $c = 0.12$. For the coded BUS sequence, the performance ratio curves for each layer are shown in Fig. 16. As shown in Fig. 16, most of the time, the performance ratio curves for more important layers stand above the curves corresponding to those for less important layers. But, due to random nature of the encoding graph, there is no guarantee that the more

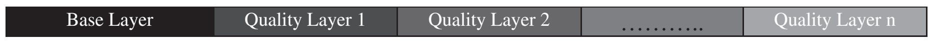

Fig. 13. General view of a data-partitioned scalable stream.

NAL units with Quality_id equal to i and temporal_id equal to **0**	NAL units with Quality_id equal to i and temporal_id equal to **1**	NAL units with Quality_id equal to i and temporal_id equal to **2**	NAL units with Quality_id equal to i and temporal_id equal to **3**	NAL units with Quality_id equal to i and temporal_id equal to **4**

Fig. 14. The internal arrangement of the quality layer *i* in data-partitioned stream for GOP = 16.

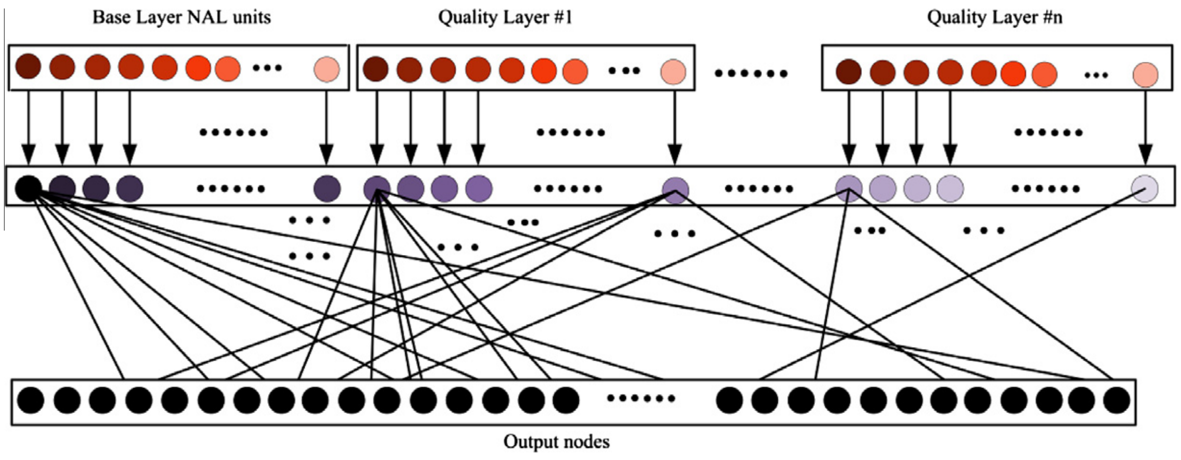

Fig. 15. Data-partitioned sequence instead of input nodes in modified encoding graph.

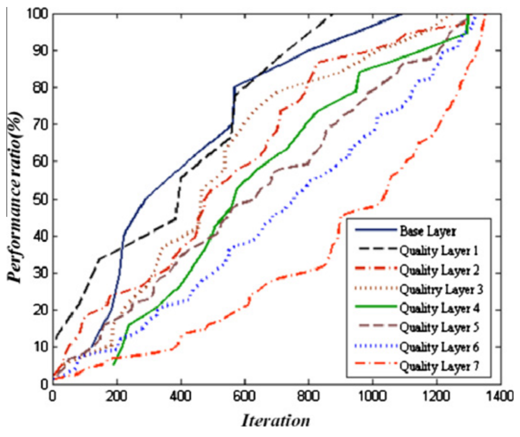

Fig. 16. Performance ratio curves for the coded dada-partitioned BUS sequence.

important layers certainly get decoded before the others. As shown in Fig. 16, for some iteration, the performance ratio of the quality layer 1 is above the performance curve for the base layer and even this quality layer is decoded before the base layer. In general, using the proposed modified encoding graph increases the probability of the early decoding of the more important parts of a sequence; but it doesn't mean that the more important parts definitely get decoded before the others. Also, because of the random nature of the code, running the encoding and decoding procedure anew provides similar but not exactly the same results. The unequal packet property introduced by using the proposed modified encoding graph LT codes provides conditions under which the more important layers have more chance to be safely delivered at the destination side before the others. This property offers a suitable solution for the scenarios that one source tries to communicate video contents to the multiple receivers with different play back capabilities. In that case, the transmitter produces a high quality original content that may be delivered to the sophisticated receivers. Other receivers with the lower capability could also safely receive the extracted version of the original content. Therefore, the complicated encoding operation is done just once and any receiver could receive parts of the original sequence depending on its capability. Using the proposed modified encoding graph LT codes, not only provides a robust approach for the contents to be delivered to the destinations safely, but also offers an appropriate scheme to reduce the recovery time for the receivers with lower capabilities. In contrast to the sophisticated receivers, that need to receive all parts of the original sequence, receivers with lower capability just need to receive some

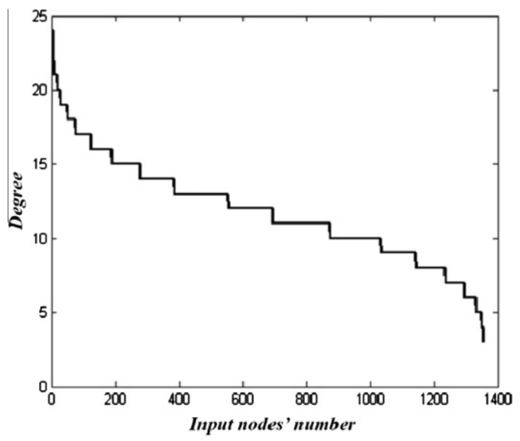

Fig. 17. Input nodes' degree distribution in the modified encoding graph for $K = 1355$, $\delta = 0.01$ and $c = 0.12$.

essential parts of the original stream to perform the decoding procedure. Most of the time, these parts are delivered before the rest, since they are often assumed as the high degree input nodes in the proposed modified encoding graph. So, it's not necessary for the low capability receivers to wait as much as the sophisticated receivers do to recover all parts of the original content. They simply could start play back procedure as soon as the essential parts are recovered.

By applying the proposed modified encoding graph LT codes to data-partitioned stream, chance of early decoding of NAL units of each layer is directly dependent to the degree of its corresponding input node on the encoding graph. Fig. 17 shows the input nodes' degree distribution of the proposed modified encoding graph used for BUS sequence. As shown in Fig. 17, the input nodes' degree is decreasing by moving toward the end of stream; so, considering data-partitioned stream as the input nodes in the modified encoding graph (Fig. 15) increases the chance of early decoding of NAL units that are closer to the beginning of the stream. Therefore, the proposed scheme can also improve the error resiliency mechanism. For example, if the link between receivers and transmitter is disconnected in some stage of transmission, then the packets received before the disconnection time are likely to be more important parts of the stream that leads to better error resiliency performance.

5. Conclusions

Unequal packet recovery is intended to deliver the most important data packets to the destination earlier than the rest. The proposed modified encoding graph LT code was aimed to fulfill this goal. Feeding the modified encoding graph with the data-partitioned SVC video sequence as its input nodes provides the unequal packet recovery for the SVC packets. Data partitioning was also performed in a way that the most important video packets are placed in the data-partitioned sequence according to their importance as discussed. The proposed method has several outstanding features. As mentioned, if the link between receivers and transmitter is disconnected in some stage of transmission, the packets received before the disconnection time would be likely the more important parts of the stream that provide better error resiliency performance. Moreover, the proposed scheme offers a suitable solution for the scenarios that one source tries to communicate video contents to the multiple receivers with different play back capabilities. Receivers with the lower capability could extract their contents from the original contents according to their capabilities. The modified encoding graph LT codes, not only provides a robust approach for the contents to be delivered to the destinations safely, but also offers an appropriate scheme to reduce the recovery time for the destinations with the lower capabilities. Finally, due to the decreasing rhythm of the input nodes' degree in the modified encoding graph, the proposed scheme is thus appropriate to be used for providing unequal packet recovery for any source data that has different important hierarchies.

References

[1] Ahmed T, Mehaoua A, Boutaba R, Iraqi Y. Adaptive packet video streaming over IP networks: a cross layer approach. IEEE J Sel Areas Commun 2005;23(2):385–401.

[2] Rosenberg J, Schulzrinne H. An RTP payload format for generic forward error protection. Network Working Group, RFC 2733; 1999. URL: <http://www.faqs.org/rfcs/rfc2733.html>.

[3] Li A. RTP payload format for generic forward error correction. Network Working Group, RFC 5109; 2007. URL: <http://www.faqs.org/rfcs/rfc5109.html>.

[4] Wenger S, Hannuksela MM, Westerland M, Singer D. RTP payload format for H.264 video. Network Working Group, RFC 3984; 2005. URL: <http://www.faqs.org/rfcs/rfc3984.html>.

[5] Wenger S, Wang YK, Schierl T, Eleftheriadis A. RTP payload format for SVC video. Internet draft; October 2009. URL: <http://tools.ietf.org/html/draft-ietf-avt-rtp-svc-22>.

[6] Luby M. LT codes. In: The 43rd annual IEEE symposium on foundations of computer science; 2002.

[7] Byers JW, Luby M, Mitzenmacher M. A digital fountain approach to asynchronous reliable multicast. IEEE J Sel Areas Commun 2002;20:1528–40.
[8] Hamzaoui R, Stanković V, Xiong Z. Optimized error protection of scalable image bit streams. IEEE Signal Proc Mag 2005;22:91–107.
[9] Ru C, Yin L, Lu J, Chen CW. Unequal video transmission based on dynamic resource allocation in MIMO OFDM system. In: IEEE conference on wireless communication and networking, (WCNC 2007); 2007.
[10] Generic coding of moving pictures and associated audio information – part 1: systems, ITU-T Rec. H.222 and ISO/IEC 13818-1 (MPEG-2 systems). ITU-T and ISO/IEC JTC 1; November 1994.
[11] Text of ISO/IEC 14496-10:2005/FDAM 3 scalable video coding, joint video team (JVT) of ISO-IEC and ITU-T VCEG. Lausanne, N9179; September 2007.
[12] Advanced video coding for generic audiovisual services, ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC). ITU-T and ISO/IEC JTC 1; May 2003.
[13] Segall CA, Sullivan GJ. Spatial scalability within H.264/AVC scalable video coding extension. IEEE Trans Circuits Syst Video Technol 17(9);2007:1121–35.
[14] Schwarz H, Marp D, Wiegand T. Overview of scalable extension of the H.264/MPEG-4 AVC video coding standard. IEEE Trans Circuits Syst Video Technol 17(9);2007:1103–20.
[15] Wang Y, Hannuksela MM, Pateux S, Elteftheriadis A, Wenger S. System and transport interface of SVC. IEEE Trans Circuits Syst Video Technol 17(9);2007:1149–63.
[16] JSVM software manual available at: <http://ube.ege.edu.tr/~boztok/JSVM/SoftwareManual.pdf>.
[17] Mackay DJC. Fountain codes. IEE Proc Commun 2005;152:1062–8.
[18] Rahnavard N, Vellambi BN, Fekri F. Rateless codes with unequal error protection property. IEEE Trans Inf Theory 2007;53(4).
[19] Mitzenmacher M. Digital fountains: a survey and look forward. In: Proceedings of information theory workshop (IEEE Cat. No. 04EX944); October 2004.
[20] Xu L. Resource-efficient delivery of on-demand streaming data using UEP codes. IEEE Trans Commun 51(1);2003:63–71.
[21] Amonou I, Cammas N, Kervade S, Pateux S. Optimized rate-distortion extraction with quality layers in the scalable extension of H.264/AVC. IEEE Trans Circuits Syst Video Technol 17(9);2007:1186–93.
[22] Isabelle Amonou, Nathalie Cammas, Sylvain Kervadec, Stephane Pateux. Some considerations about restriction on priority_id syntax element, JVT-T051, Join Video Team of ISO/IEC and ITU-T. 20th Meeting, Klagenfurt, AU; 15–21 July, 2006.
[23] <ftp.tnt.uni-hannover.de/pub/svc/testsequences/>; 2010 [accessed December 2010].
[24] <http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm> [accessed December 2010].

Ehsan Namjoo received the B.Sc. degree in electrical engineering from the KNTU University of Technology, Tehran, Iran, in 2004 and his M.Sc. degree in communication systems from the University of Tabriz, Tabriz, Iran in 2006. He is currently pursuing Ph.D. program at University of Tabriz. His current research interests are image processing, rateless codes, pattern analysis and video coding.

Ali Aghagolzadeh received Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 1991 in electrical engineering. He is currently professor of electrical engineering in University of Tabriz, Tabriz, Iran, and in Babol Noshirvani University of Technology, Babol, Iran. His research interests include image processing, video coding and compression, information theory, computer vision.

Javad Musevinia received his B.S. degree from the University of Tehran and his M.Sc. and Ph.D. degrees both in communications from Sharif University of Technology and the University of Tabriz, respectively. He is currently assistant professor of electrical engineering of the University of Tabriz. His current research interests include wireless communication, multimedia networks and signal processing for communication systems.