JOINT SOURCE-NETWORK ERROR CONTROL CODING FOR SCALABLE OVERLAY VIDEO STREAMING

Yufeng Shan,[1] *Ivan V. Bajic,*[2] *Shivkumar Kalyanaraman,*[1] *and John W. Woods*[1]

1/ ECSE Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
2/ ECE Department, University of Miami, Coral Gables, FL 33146, USA

ABSTRACT

In this paper, we propose a Joint Source-Network error control coding (JSNC) scheme which efficiently integrates scalable video coding, error control coding and overlay infrastructure to stream video to heterogeneous users. The distributed overlay data service nodes over the IP network form an application layer media service overlay network and adapt both the video bitstream and error control coding based on both user requirements and available bandwidth. A novel Fine Granular Adaptive FEC (FGA-FEC) scheme, a generalization of MD-FEC, is proposed for error recovery during video transmission to heterogeneous users. Encoding once, the FGA-FEC can adapt the FEC encoded scalable MC-EZBC bitstream to satisfy multiple heterogeneous users simultaneously without decoding/re-encoding FEC at intermediate nodes, thus saving computation

1. INTRODUCTION

Simultaneously streaming video to heterogeneous users, such as powerful PCs, laptops and handset devices, is a challenging problem, since different users may have different video frame rate/quality/resolution requirements, computation capabilities and network connections. In order to serve heterogeneous users, traditional approaches (Windows media, Real player) maintain multiple versions of any piece of media that suit a variety of capabilities and preferences. While streaming, the server sends multiple copies of the same bitstream to different users. IP multicast is an efficient way for simultaneous bulk data delivery. Due to the deficiency of its deployment in the current Internet, application-layer multicast [3] is proposed. End systems, instead of routers, are organized into an overlay to relay data to each other in a peer-to-peer fashion. Padmanabhan *et al* in [4] discuss the problem of distributing streaming media content to a large number of hosts in a scalable way. They propose CoopNet, where clients cooperate to distribute content, thereby alleviating the load on the server. In [5], Cui *et al* propose a peer-to-peer streaming solution with cache-and-relay nodes. Most recently, service overlay network are gaining attention. Amir *et al* [8] proposed a specific architecture for active services that are used for deployment of user-defined application-level computation within the network.

The above papers consider network and video coding separately. We argue that this approach is not optimal. Currently, MPEG-21 aims to enable the use of multimedia resources across a wide range of network and devices. The Digital Item Adaptation (DIA) part raises the possibility of in-network video adaptation which fit well to the overlay infrastructure. In this paper, we consider networking and application as one system by efficiently joining scalable video coding algorithms and overlay infrastructure for streaming. The proposed JSNC scheme includes the following two novel concepts:

1. *Integrated Source-Network Video Coding (IVC)* Video coding function is distributed both at source and inside the network to facilitate simple and precise adaptation of bitstream for heterogeneous users.
2. *A novel Fine Granular Adaptive FEC (FGA-FEC)*. Encoding once, the proposed FGA-FEC scheme can adapt the FEC coded bitstream to satisfy multiple heterogeneous users simultaneously without FEC decoding/re-encoding at intermediate overlay nodes.

Our work is different from proxy based streaming and multicast layered streaming. Proxy streaming systems cache video content at proxy local disk and trans-codes (decode and re-encode) the video bitstream for different users. In multicast layered video streaming, receivers adapt to network conditions by joining and leaving multicast groups, which results in large amount of signaling traffic in a dynamic network. Further, the adaptation is limited to available layers and also need support of an application layer multicast infrastructure. On the other hand, our JSNC can arbitrarily adapt video frame rate, quality and resolution without transcoding. Moreover, our scheme can provide an efficient error control mechanism for heterogeneous users.

The rest of the paper is organized as follows. In Section 2 we describe the detail of JSNC scheme. Simulations and experimental results are given at Section 3, followed by conclusions at Section 4.

2. Joint Source Network Error Control Coding

JSNC uses an overlay infrastructure to assist video streaming to multiple users simultaneously by providing light weight support at intermediate overlay nodes. For example, in Fig. 1, overlay data service nodes (DSN) construct an

overlay network to serve heterogeneous users. Users "A" to " G" have different video requirements (shown as " frame rate /resolution/available bandwidth"). Here " C" and " Q" represent CIF and QCIF format, respectively. "P_a" to "P_g" are the loss rates of different overlay virtual links. In this paper, we assume an overlay network is ready for use.

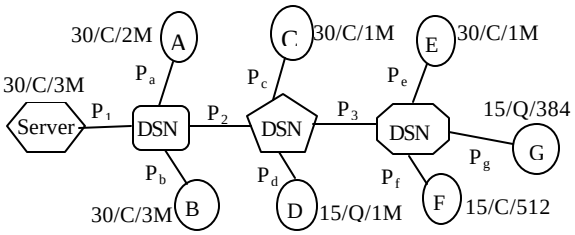


Fig. 1: Intermediate adaptation of the video bitstream

While streaming, the server sends out a single FGA-FEC coded bitstream based on the highest user requirement (30/C/3M) and network conditions. FGA-FEC divides each network packet into small blocks and packets the FEC coded bitstream in such a way that if any original data packets are actively dropped, the corresponding information in parity bits is also completely removed. The intermediate DSNs can adapt the FEC coded bitstream by simply dropping a packet or shortening a packet by removing some of its blocks. Since there is no FEC decoding/re-encoding, JSNC is very efficient in terms of computation. Furthermore, the data manipulation is at block level, which is precise in terms of adaptation.

2.1 Integrated Source Network Video Coding (IVC)

The main goal of IVC is to facilitate simple and precise adaptation of bitstream for heterogeneous users in an intermediate overly node. Based on this, the server first encodes the bitstream in such a way that the subsets corresponding to low resolution/frame rate/quality version of the video are embedded in bitstreams corresponding to higher resolution/frame rate/quality version of the video. Different sub-bitstreams can be extracted by intermediate DSNs in a simple manner without trans-coding, to readily accommodate a variety of users for a given user's computing power, connection bandwidth, and so on. In this paper, we use a fully scalable MC-EZBC video coder [6] to show the efficiency of IVC.

MC-EZBC produces embedded bitstreams supporting a full range of scalabilities – temporal, spatial and SNR. Here we use the same notation as [6]. Each GOP coding unit consists of independently decodable bitstreams $\{Q^{MV}, Q^{YUV}\}$. Let $l_t \in \{1,2,..., L_t\}$ denote the temporal scale, MV bitstream, Q^{MV}, can be divided into temporal scales and consists of Q_{lt}^{MV} for $2 \le l_t \le L_t$. Let $l_s \in \{1,2,..., L_s\}$ denote the spatial scale, Subband coefficient bitstream, Q^{YUV}, is also divided into temporal scales and further divided spatial scales as $Q_{lt,ls}^{YUV}$, for $2 \le l_t \le L_t$ and $1 \le l_s \le L_s$. The video at $(1/4)^r$ spatial resolution and $(1/2)^f$ frame rate is obtained from the bitstream as:

$$Q = \{Q_{lt}^{MV} : 1 \le l_t \le L_t - f\} \bigcup \{Q_{lt,ls}^{YUV} : 1 \le l_s \le L_s - r\}$$

In every sub-bitstream $Q_{lt,ls}^{YUV}$, the subbands from *Y*, *U* and *V* are progressively encoded from the most significant bitplane (MSB) to the least significant bitplane (LSB). Scaling in term of quality is obtained by stopping the decoding process at any point in bitstream *Q*. The MC-EZBC encoded bitstream can be further illustrated as Digital Items as in Fig. 2 [7], where shows the video bit stream in view of three forms of scalability.

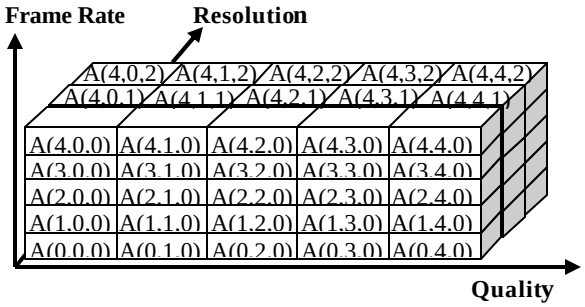


Fig. 2 3-D video scalability

In Fig. 2, the video bitstream is represented in terms of atoms, which are usually bit planes or fractions of bit planes. The notation *A*(*F*, *Q*, *R*) represents an atom of {frame rate, quality, resolution}. Choosing a particular subset of atoms corresponds to scaling the resulting video to the desired resolution, frame rate and quality. These small pieces of bitstream are interlaced in the embedded bitstream. Intermediate DSNs adapt the digital items according to user preferences and network conditions. Since the adaptation can be implemented as simple dropping of corresponding atoms, DSNs do not need to decode and re-encode the bitstream, which is very efficient. On the other hand, the adaptation is done based on atoms in a bitstream, which is almost as precise as pure source coding.

2.2 Fine Granular Adaptive FEC (FGA-FEC)

DSNs adapt video bitstream based on user requirements and available bandwidth. When parts of the video bitstream are actively dropped, FEC codes need to be updated accordingly. This update of FEC codes has the same basic requirements as the in-network video coding – efficiency (low computation cost) and precision (if a part of the video data is actively dropped, parity bits protecting that piece of data should also be removed). Based on these considerations, we propose a precise and efficient Fine Granular Adaptive FEC (FGA-FEC) scheme based on Reed-Solomon (RS) codes. Arbitrary adaptation of RS codeword is difficult. For example, RS(*n*, *k*) can not be adapted to RS(n-y, k-y) by simply dropping data packets. One way to adapt the RS(n,k) to RS(n-y, k-y) is to decode RS(n,k) and

then re-encode RS(n-y, k-y), which is not efficient in case of multiple adaptations along transmission path. FGA-FEC solves the problem by fine granular adapting the FEC to suit multiple users simultaneously.

Given a piece of video bit stream, shown in Fig.3 (top), divided into chunks as A, B, C, …, X, the FGA-FEC further divides each chunk of bitstream into equal size blocks. The precision of fine granularity of the FGA-FEC scheme is determined by the block size. Smaller block size means finer granularity and better adaptation precision. In Fig. 3, the bitstream is divided into blocks as (A1, A2; B1,..,Bi; C1,..,Cj; ..; X1,..,Xn). RS codes computation is applied across these blocks to generate parity blocks. Each vertical line represents a data chuck divided into blocks, followed by the generated parity blocks. More protection is added to the important part of the bitstream and less FEC is allocated to data with lower priority. The optimal allocation of FEC to different chunk of data is described at our previous work [1] and [2]. After FEC encoding, each horizontal line is packetized as one description. In this paper, one description is equivalent to one network packet. Similar as MD-FEC [2], FGA-FEC transforms a priority bitstream into non-priority descriptions. In addition, FGA-FEC scheme has the ability of fine granular adaptation at block level. Based on our experiment about a 1Mbps *Foreman* CIF bitstream with first GOP 16 frames, MD-FEC can generate block size varies from 1bits to 475 bits which is difficult to do in network adaptation.

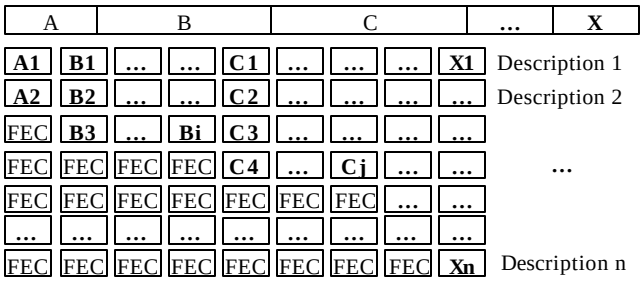


Fig. 3 FGA-FEC encoding scheme

The granularity of FGA-FEC adaptation is at block level. Assume, for instance, that an intermediate DSN needs to adapt the video bitstream by dropping one chunk of bitstream, say *C* in Fig. 3. This can be achieved by removing the original data and FEC blocks related to chunk *C* from each packet. The adaptation at intermediate overlay node is simple a filtering of a network packet as

$$M_p F_p = M_{p-updated}$$

where M_p = [b*1*,*b2*,…,*bn*] is a packet represented in the block-matrix format, the elements are blocks inside a packet. $M_{p\text{-}updated}$ is the updated packet with some blocks being dropped by the filter matrix F_p. F_p is an updated matrix from an *n* x *n* identity matrix with related columns removed. For example, if *jth* block need to be dropped, F_p is a *n-1* x *n* identity matrix with *j*th column removed from a *n* x *n* identity matrix as following

$$Fp = \overbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}}^{\text{n-1}} \Bigg|_{\text{n}} \quad \begin{array}{l} j\text{th column is removed from} \\ \text{a } n \text{ x } n \text{ identity matrix} \end{array}$$

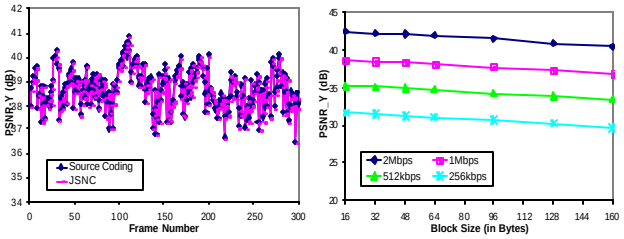
Hence, all descriptions (both data and parity) are shortened, and this is the only processing that needs to be done – no FEC trans-coding is necessary. Further, the removed parity bits correspond precisely to the data bits that are dropped.

To facilitate intermediate overlay node adaptation, an information packet is sent ahead of one GOP to tell the intermediate nodes about the block size, FEC codes and bitstream information in block level. Since the description is at block level, we can engineer the information packet to occupy less then 1% of the total bandwidth of a GOP.

3. RESULTS

We performed several simulations and experiments to show the effectiveness of our proposed building blocks. We use 300 frames of the *foreman* CIF sequence at 30 fps. Adaptations are done at intermediate overlay nodes using the encoded MC-EZBC bitstream. Each simulation is run at least ten times to obtain statistically meaningful results

The first question that needs to be answered is: does the in-network block-based adaptation of bitstream achieve the same quality as source coding?



(a) (b)

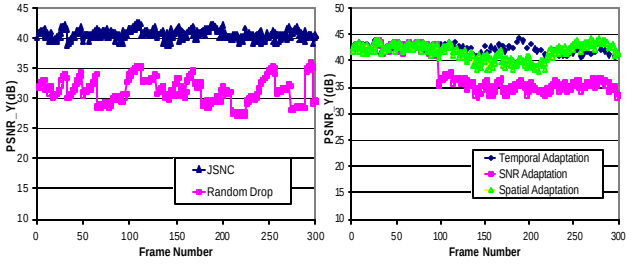
Fig. 4 (a) source coding vs JSNC; (b) effect of block size

Given a video bitstream at 1Mbps, JSNC packets the bit-steam into 512-byte network packet with block size at 8 bytes, thus, results in 128 packets per GOP on average. The adaptation at intermediate node is at block level. Therefore, the adaptation may not as precise as source coding. For example, JSNC can adapt 1Mbps bitstream to 992kbps and 976kbps by removing 2 or 3 blocks from each packet, respectively. Suppose the last mile available bandwidth of a user is 991kbps, JSNC can only adapt the bitstream to 976kbps. Fig. 4 (a) shows PSNR_Y of the source coded video at 991bps versus our proposed JSNC adaptation to network condition. The overall PSNR of JSNC is 0.08dB lower than source coding.

Obviously, the block size can affect the adaptation precision. Fig. 4(b), we show the granularity of adaptation in different block size. Here, one block is removed from each network packet. Clearly, smaller block size means finer granularity.

Traditionally, when network congestion occurs, data packets are randomly dropped to avoid congestion. On the other hand, our JSNC scheme adapts the packets in the intermediate network nodes to reduce the bandwidth requirement. Given a 1.5Mbps bitstream and the available bandwidth at 1455kbps, In Fig. 5 (a) we compare PSNR-Y of JSNC vs. random drop scheme with 3% packets being dropped. Observe that the proposed scheme significantly outperforms random dropping by about 10 dB.

A user may have requirements priorities to response the dynamic of network conditions. In Fig. 5 (b) we show the corresponding video quality when the available bandwidth changes. Originally, the user is receiving a 2 Mbps, CIF formant, 30 fps bitstream. Starting with frame 100, the user has only 512 kbps available bandwidth.



(a) (b)

Fig. 5. (a)JSNC vs. random drop; (b) 3-D adaptation

There are three possible choices for the user: (1) SNR adaptation to 512kbps; (2) Temporal adaptation to 1/4 of the original frame rate; (3) Spatial adaptation to QCIF. Both (2) and (3) need additional SNR adaptation to 512 kbps. User can choose its preference based on Fig. 5 (b).

Next, we compare the performance of JSNC with Hop by Hop FEC scheme. The reason we choose to compare with Hop by Hop FEC is that both of the schemes can do adaptation in intermediate nodes. Suppose server streams video to user "*E*" at Fig.1, the available bandwidth of *E* is 1Mbps, the loss rate is at $P_1=P_2=P_3=P_e=1.5\%$, we assume that there is enough bandwidth between DSNs. In order to fully recover the losses, JSNC adds FEC based on end to end loss rate which is approximately 6%. On the other hand, Hop by Hop FEC need only to protect 1.5% loss rate at each virtual link. Thus, the received video quality using Hop by Hop FEC is 0.22dB better than using our proposed JSNC, shown at Fig. 6. But the Hop by Hop FEC video quality gain is acquired by scarifying system computation performance, the intermediate nodes need to decode/re-encode the FEC at each DSN. Our JSNC adaptation is only actively removing blocks within one packet instead of complex FEC computation, the computation burden is very low. We test the FEC encoding/decoding time at a P4 2.0GHz Linux 8.2 PC, for RS(120,115), it approximately needs 2ms for the computation. We also tested JSNC adaptation burden at the same PC, the adaptation time to process the same among of packets is about $1x10^{-4}$ms. Thus, our JSNC scheme is efficient in term of computation.

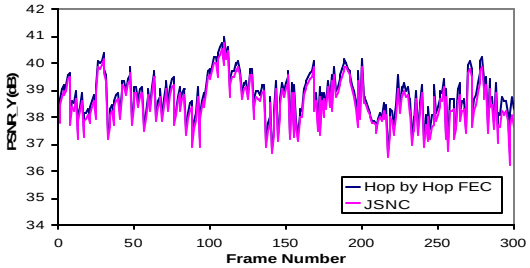


Fig. 6 JSNC vs Hop by Hop FEC

4. CONCLUSIONS

In this paper, we proposed a Joint Source-Network error control coding scheme for scalable video streaming. The adaptation in the intermediate overlay nodes is fine granular at block level. Adaptation quality is almost the same as pure source coding. A novel FGA-FEC scheme is proposed for error recovery during video transmission to heterogeneous users. Encoding once, the proposed FGA-FEC scheme can adapt FEC codes by only adjusting the packet size instead of FEC decoding/re-encoding in the intermediate nodes. Simulations show that the proposed JSNC can efficiently and precisely stream scalable video to heterogeneous users simultaneously.

5. REFERENCES

[1] I. V. Bajic and J. W. Woods, "EZBC video streaming with channel coding and error concealment," in *Proc. SPIE VCIP* 2003, Lugano, Switzerland, July 2003

[2] R. Puri and K. Ramchandran, "Multiple description source coding using forward error correction codes," Proc. 33rd ACSSC, Pacific Groove, CA, October 1999.

[3] Y. Chu, S. Rao, and H. Zhang "A case for end system multicast," *Proc. ACM SIGMETRICS*, pp. 1-12, CA, June 2000

[4] V. Padmanabhan, H. Wang, et al, "Distributing streaming media content using cooperative networking," MSR-TR-2002-37

[5] Y. Cui and K. Nahrstedt "Layered Peer-to-Peer Streaming," *Proc. NOSSDAV*, Monterey, CA, June 2003.

[6] S.-T. Hsiang and J. W. Woods, "Embedded video coding using invertible motion compensated 3-D subband filter bank," *Signal Processing: Image Commun.*, pp. 705-724, May 2001.

[7] http://www.hpl.hp.com/research/ssm

[8] E. Amir, S. McCanne, and R. Katz, "An active service framework and its application to real-time multimedia transcoding", *Proc. ACM SIGCOMM*, pp. 178-189, Vancouver 1998.