Scalable Joint Source-Network Error Control Coding for Overlay Video Streaming

Yufeng Shan, *Student Member, IEEE,* Ivan V. Bajic, *Member, IEEE,* John W. Woods, *Fellow, IEEE,* and Shivkumar Kalyanaraman, *Member, IEEE*

***Abstract*— In this paper, we investigate a joint source-network coding scheme for adapting a scalable video bitstream. Both embedded bitstream and error-control codes are easily and precisely adapted in a multidimensional way at intermediate overlay nodes to satisfy multiple heterogeneous users without complex transcoding. These distributed nodes form an overlay service network, and adapt both the video bitstream and error-control codes based on both user video request and network conditions. Video coding functions are distributed across the video source and the network. A novel fine granular adaptive FEC scheme, a generalization of MD-FEC, is proposed for error recovery during video transmission to heterogeneous users. Encoding once, the new method can satisfy multiple heterogeneous users simultaneously without decoding/recoding FEC at intermediate network nodes.**

***Index Terms*— scalable video, FEC, overlay network, adaptation, video streaming**

I. Introduction

Simultaneously streaming video to heterogeneous devices, such as powerful PCs, laptops, and handset devices, is a challenging problem, since different users may have different video frame-rate, resolution, and quality preferences, or computational and connection-link capabilities. In order to serve heterogeneous users, conventional 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 separate copies of the same bitstream to different users, which is clearly not efficient in terms of bandwidth utilization. IP multicast is an efficient way for simultaneous bulk data delivery. The most serious problem faced by multicast today is the deficiency of its deployment in the wide-area network infrastructure. As an alternative, application-layer multicast [1] was proposed. In this approach, end systems, instead of routers, are organized into an overlay network to relay data to each other in a peer-to-peer fashion. Recently, service overlay networks (SON) [2] [3] [4] [5] are gaining attention, in which user-defined application-level functionalities are provided at overlay nodes, more than simple forwarding of packets.

This paper explores the feasibility of using a service-overlay network to address the problem of streaming video to heterogeneous users simultaneously. The challenge is to encode a video to facilitate efficient and precise adaptation of the encoded bitstream (adapting both the video bitstream and error control codes) to satisfy multiple users without complex transcoding at intermediate overlay nodes. Here, by 'adapt' we mean reducing one or more of the three scalability dimensions (frame-rate, resolution, and quality) of a video bitstream along with the corresponding error-control codewords to satisfy heterogeneous users and respond to network congestion.

MPEG-21 [6] aims to enable the use of multimedia resources across a wide range of networks and devices. The proposal for the MPEG-21 Part 7 standard is digital item adaptation (DIA), which raises the possibility of in-network video adaptation [6][7] and fits very well into an overlay infrastructure. To adapt a multimedia bitstream for multiple users, Mukherjee *et al* [8][9] developed a metadata-based method called structure scalable meta-format (SSM) [10]. Their work focuses on a framework for modeling and adaptating arbitrary scalable multimedia bitstreams in a manner that is fully format agnostic, but no specific error control methods are considered to match the bitstream adaptation. Priority encoding transmission (PET) by Albanese *et al* [11] is a packetization scheme that combines layered source coding with unequal erasure protection. The authors applied the PET scheme to the *I*, *P*, and *B* layers of MPEG video, but did not optimize the code rate to minimize the end-to-end distortion for a given overall transmission rate. Several algorithms have been proposed for optimal forward error-correction code (FEC) assignment for progressive (embedded) data. Puri *et al* [12] solved the problem using a Lagrange multiplier-based algorithm. Mohr *et al* [13] described how to achieve an approximately optimal assignment of FEC to progressive data using a local search algorithm that is essentially a Lagrangian optimization. Stankovic *et al* [14] presented an efficient algorithm for greedy search from a near optimal initial condition. The above papers mainly focus on developing end-to-end optimization schemes to protect a progressive bitstream, without any adaptation being considered for diverse users. Chou *et al* [15] presented and evaluated constructions for two-layer multiple description codes using FEC to satisfy various user preferences. Stankovic *et al* [16] modified the method of [15] and defined an optimal layered multiple-description code as one that minimizes the largest performance loss experienced by any client. Both of the papers did not consider multiple adaptations for several users simultaneously.

The main contribution of this paper is a joint source-network coding (JSNC) scheme which enables multidimensional, arbi-

Yufeng Shan, Shivkumar Kalyanaraman and John W. Woods are with Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA.

Ivan V. Bajic is with School of Engineering Science, Simon Fraser University, Vancouver, BC, Canada.

This work was partly supported by the ARO grants DAAD19-00-1-0559 and W911NF-04-1-0300, part of this work was presented at ICIP 2005, Geneva, Switzerland

trary, and efficient, yet near optimal adaptation of both the encoded video bitstream and the error-control code to serve multiple diverse users. JSNC includes the following two novel concepts:

1) integrated video coding (IVC): Video coding functions (i.e. bitstream truncation and packetization) are distributed across the source and the network to facilitate simple and precise adaptation of the bitstream for heterogeneous users.
2) novel fine granular adaptive FEC (FGA-FEC): In cooperation with IVC, the proposed FGA-FEC scheme can adapt the FEC coded bitstream to satisfy multiple heterogeneous users without FEC decoding/recoding at intermediate overlay nodes. This extends scalability to channel coding.

Our work is different from proxy-based streaming and multicast layered streaming. Proxy streaming systems cache video content at a local proxy disk and transcode (decode/recode) the video bitstream for different users. In multicast layered video streaming, a video server sends different layers to different multicast groups. Receivers adapt to network conditions by joining and leaving these multicast groups, which however, results in a large amount of signaling traffic in a dynamic network. Further, the adaptation is limited to available layers. Meanwhile, our JSNC only sends one bitstream but can arbitrarily adapt video frame rate, quality, and spatial resolution without transcoding. Moreover, JSNC can provide an efficient error-control mechanism in cooperation with scalable video to satisfy heterogeneous users simultaneously.

The rest of our paper is organized as follows. In Section II, we describe the details of our JSNC scheme. Simulated and experimental results are given in Section III. Conclusions and prospective future work are listed in Section IV.

II. Joint Source Network Coding

A. System Overview

JSNC uses an overlay infrastructure to assist video streaming to multiple users by providing light weight support at intermediate overlay nodes. These overlay nodes with certain service functions, such as bitstream adaptation, FEC computation and so on, more than just store-and-forward, are called data service nodes (DSN). Diverse end users may have different network connection, computational capacity, and video display size, hence, they probably have different subjective ideal video and adaptation order preferences. Here, *ideal video* is defined as the type of bitstream the user initially requests from the system. Additionally, *adaptation order* is the user's chosen adaptation order in terms of quality, frame rate and resolution. The ideal video and adaptation order are input to system from the user-node console at the beginning of streaming. Since DSNs are often placed within a high-speed network, in this paper, we assume that there is *no congestion between DSNs*. We also assume that a profile (description file [17]) of the video bitstream is sent before the streaming session to both end users and DSNs to facilitate adaptation.

We outline our idea with a simplified example. In Fig. 1, DSNs construct an overlay network to serve several users. Users "A" to "G" have different video requests (shown as "frame-rate/resolution/bitrate"). Here C and Q represent the common CIF and QCIF formats, respectively, and p_a to p_g are the average packet-loss rates of different overlay virtual links.

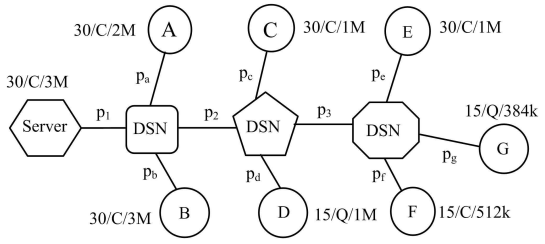


Fig. 1. Intermediate adaptation of the video bitstream according to user video requests and network conditions

While streaming, the user nodes determine their *video request* based on available bandwidth, user ideal video, and user adaptation order using Algorithm I below. These video requests are collected and aggregated from end users to the server by the DSNs. The server then encodes the scalable video using JSNC based on the highest video request (in the Fig.1, 30/C/3M) and the current packet-loss rates. JSNC divides each network packet into small blocks and packs the FEC coded bitstream in such a way that if any original data packets are adapted (dropped or shortened), the corresponding parity bits are also completely removed. At intermediate DSNs, the adaptation is conducted by removing some blocks from each packet and/or dropping whole packets to satisfy the end users (described in detail at Section II-D). Since there is no FEC decoding/recoding, JSNC is very efficient in terms of computation. Furthermore, the data manipulation is at block level, which is precise in terms of adaptation, given a sufficiently small block size.

Three important questions need to be answered.

- How should the video be encoded by JSNC to facilitate multidimensional adaptation?
- How should the bitstream be adapted efficiently, given the user's ideal video, adaptation order, and network conditions?
- How should FEC be designed to accommodate heterogeneous users, and yet be easily adaptable for highly scalable bitstreams at intermediate data service nodes?

B. Integrated Video Coding (IVC)

The main goal of IVC is to encode and represent a scalable video to facilitate simple and precise adaptation of the bitstream to the available bit budget, both at the source and inside the network.

In general, a piece of scalable bitstream which contains N nested tiers of scalability with the ith tier containing L_i layers, $i = 0, 1, 2, ..., N-1$, can be represented as a N-dimensional hypercube. The total number of *atoms* (elements) of the cube is $\prod_{i=0}^{N-1} L_i$. A specific atom is denoted $A(l_0, l_1, ..., l_{N-1})$, where $l_i \in \{0, 1, ..., L_i - 1\}$. For instance, Fig. 2 shows a three-dimensional (3-D) cube of atoms in dimensions {frame

rate ,resolution ,quality} [10]. There are $L_0 = 5$ frame-rate layers, $L_1 = 3$ resolution layers, and $L_2 = 5$ quality layers. Each atom corresponds to a piece of subband bitstream with size in bits:

$$S(l_0, l_1, ..., l_{N-1}) = \|A(l_0, l_1, ..., l_{N-1})\|. \quad (1)$$

Adaptation of the scalable bitstream is equivalent to selecting a subset of atoms for transmission.

We use the fully scalable MC-EZBC video coder [18] to show the method of adaptation. We limit the tiers of scalability to three: temporal (frame-rate) scalability, spatial (resolution) scalability and SNR (quality) scalability. In the sequel, we use $\{L_t, L_s, L_q\}$ instead of the more general $\{L_0, L_1, L_2\}$.

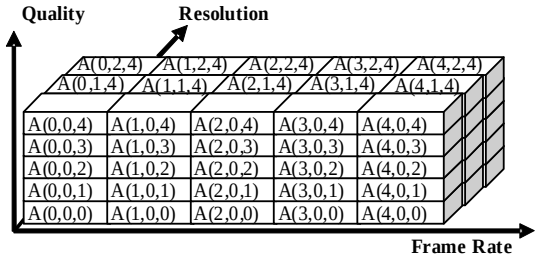


Fig. 2. 3-D video scalability in the form of atoms of a GOP, $A(i, j, k)$ represents an atom of {framerate, resolution, quality}

Given the representation in Fig. 2, we can adapt the atoms to the desired frame-rate, resolution and quality, according to user's ideal video and adaptation order, as well as the available bit budget. Given a bit budget Ω, a subset of atoms can be chosen to satisfy:

$$\sum_{l_t=0}^{T} \sum_{l_s=0}^{S} \sum_{l_q=0}^{Q(l_t,l_s)} S(l_t, l_s, l_q) \leq \Omega, \quad (2)$$

where T and S are the number of temporal and spatial layers, respectively, $T \leq L_t$ and $S \leq L_s$. Here $Q(l_t, l_s)$ is the number of quality layers at that temporal and spatial layer. There may be several different $T, S, Q(l_t, l_s)$ values which satisfy (2), so the particular set is chosen based on user's ideal video and adaptation order. Since we have only three forms of adaptation {frame rate, resolution, quality}, the total number of adaptation orders is $3 \times 2 \times 1 = 6$. Table I lists all the possible adaptation orders, where {*t, s, q*} represents {frame rate, resolution, quality} adaptations, respectively.

ID	Adaptation order	ID	Adaptation order
1	$q \rightarrow t \rightarrow s$	4	$t \rightarrow s \rightarrow q$
2	$q \rightarrow s \rightarrow t$	5	$s \rightarrow q \rightarrow t$
3	$t \rightarrow q \rightarrow s$	6	$s \rightarrow t \rightarrow q$

TABLE I
TOTAL NUMBER OF ADAPTATION ORDERS

The user can choose to adapt SNR, frame rate, resolution or any combination of the three. For example, the user ideal video may be to view a video with PSNR no less than γ dB with full frame rate and spatial resolution, and the predefined adaptation order is $q \rightarrow t \rightarrow s$. If the desired PSNR cannot be met, we reduce the frame rate down one temporal level and, if necessary, further reduce resolution down one spatial level, and do this iteratively, until to the minimum tolerable bitstream with ξ temporal levels and ζ spatial levels, or the minimum PSNR (quality) requirement is met.

Table II summarizes the terms used in the sequel.

Terms	Definitions
L_t,L_s,L_q	the number of layers in user's ideal video
ID	adaptation order identification number
γ, ξ, ζ	adaption order parameters: PSNR $\geq \gamma$ dB, temporal layers $\geq \xi$, and spatial layers $\geq \zeta$
maxPSNR	best achievable video quality at a certain spatial and temporal layer, given the available bit budget
B	available bandwidth of a link
p	packet-loss rate of a link
$E[D(R)]$	mean video distortion

TABLE II
TERMS USED IN THE ALGORITHM DESCRIPTIONS

To respond to the available bit budget, DSNs or user nodes adapt the bitstream based on user specified adaptation orders. At each adaptation step, we determine the best achievable video quality (maxPSNR), and iterate until satisfying max-PSNR $\geq \gamma\ dB$ according to the specified adaptation order:

$$\begin{aligned} At\ each\ step: \quad & T = i; \quad i \in \{L_t, L_t - 1, \cdots, \xi\} \\ & S = j; \quad j \in \{L_s, L_s - 1, \cdots, \zeta\} \\ Find: \quad & \\ & maxPSNR(T, S) \\ Subject\ to: \quad & \sum_{l_t=0}^{T} \sum_{l_s=0}^{S} \sum_{l_q=0}^{Q(l_t,l_s)} S(l_t, l_s, l_q) \leq \Omega \end{aligned} \quad (3)$$

A unique solution of $T, S, Q(l_t, l_s)$ can be found by use of Algorithm 1 below, given a specified adaptation order. The result of solving (3) is the target adaptation data set that constitutes this user node's video request. If all adaptations still cannot meet the user requirement, there is no bitstream sent to this user. Therefore, the user node's video request could be the same as user ideal video, given enough available bit budget, or shrink down to zero if minimal requirements cannot be met due to very low available bit budget.

Now we show how to map this adaptation method into an embedded MC-EZBC coded bitstream shown at the top in Fig. 3. The server encodes the bitstream in such a way that the subsets corresponding to lower frame-rate/resolution/quality of the video are embedded in bitstreams corresponding to higher frame-rate/resolution/quality. Different sub-bitstreams can be extracted by intermediate DSNs in a simple manner without transcoding, to readily accommodate a variety of users considering their computing power, connection bandwidth, and so on. We use the same notation as [19]. Each GOP coding unit consists of independently decodable bitstreams $\{Q^{MV}, Q^{YUV}\}$ as shown in Fig. 3. Let $l_t \in \{1, 2, ..., L_t\}$ denote the temporal scale. The motion vector (MV) bitstream, Q^{MV}, can be divided into temporal scales and consists of Q_{lt}^{MV} for $2 \leq l_t \leq L_t$. Let $l_s \in \{1, 2, ..., L_s\}$ denote the spatial scale. The subband coefficient bitstream, Q^{YUV}, is also divided into temporal scales and further divided into spatial scales as $\{Q_{lt,ls}^{YUV}\}$, for $1 \leq l_t \leq L_t$ and $1 \leq l_s \leq L_s$. Thus,

Algorithm 1: Pseudo code to find T, S, Q

Input : Ω, γ, ξ, ζ, L_t, L_s, L_q
Output: T, S, Q

$T = L_t$; $S = L_s$;
$minT = \xi$; $minS = \zeta$;
while *($T \geq minT$* **or** *$S \geq minS$)* **do**
  solve (3) ;
  if *(maxPSNR $< \gamma$)* **then**
    $T = T - 1$ if $T > minT$;
    solve (3) ;
    if *(maxPSNR $< \gamma$)* **then**
      $S = S - 1$ if $S > minS$;
    else
      solution found, **stop**
    end
  else
    solution found, **stop**
  end
end

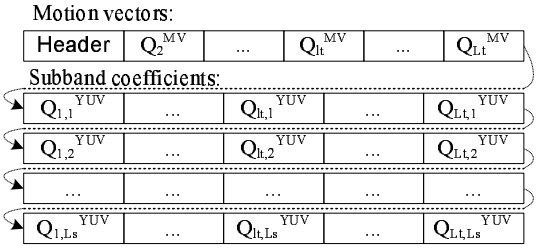


Fig. 3. Hierarchy of MC-EZBC bitstream to facilitate 3-D adaptation

the video at $(1/4)^m$ spatial resolution and $(1/2)^n$ frame rate is obtained from the full bitstream as:

$$\begin{aligned} Q_{m,n} &= \{Q_{lt,ls}^{YUV} : 1 \leq l_s \leq L_s - m; 1 \leq l_t \leq L_t - n\} \\ &\bigcup \{Q_{lt}^{MV} : 2 \leq l_t \leq L_t - n\} \end{aligned} \quad (4)$$

In every sub-bitstream $Q_{lt,ls}^{YUV}$, subbands from Y, U and V are coded in an embedded manner from the most significant bit (MSB) to the least significant bit (LSB). Scaling in terms of quality is obtained by stopping the decoding process at any point in bitstream $Q_{m,n}$, given the available bit budget.

Since the adaptation can be implemented as simple dropping of corresponding atoms, DSNs do not need to decode and recode the bitstream, thus being very efficient. Further, the adaptation is done based on atoms in a bitstream, which is almost as precise as pure source coding if the size of the atom is chosen small enough.

C. Fine Granular Adaptive FEC

Automatic retransmission request (ARQ) and FEC coding are two widely used methods to protect packets from channel losses. Due to the feedback flood problem in a multicast environment, we choose to study FEC as our protection method. How to design an FEC mechanism for heterogenous users and how to incorporate it with the IVC scheme is our main design goal.

When parts of the video bitstream are actively dropped, the DSNs need to update the FEC codes. This update has the same basic requirements as the in-network video coding - efficiency (low computational 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 and PET [11]. Arbitrary adaptation of RS codewords is difficult. For example, $RS(n,k)$ codeword cannot be adapted to $RS(n-l,k-l)$ by simply dropping l symbols. One way to adapt an $RS(n,k)$ is to decode first and then recode $RS(n-l,k-l)$, which is not computationally efficient for multiple adaptations along the transmission path or for multiple heterogeneous users. FGA-FEC solves the problem by adapting the FEC in a "fine granular" manner to satisfy multiple diverse users, as discussed below. The FGA-FEC

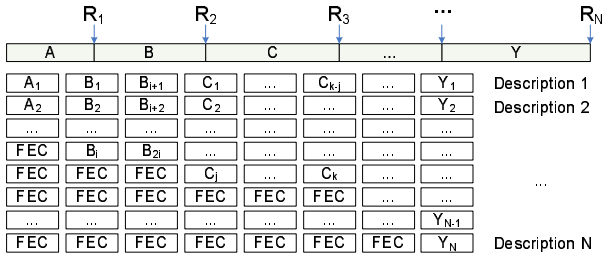


Fig. 4. FGA-FEC encoding of one GOP, FEC is added vertically at block level and each horizontal line is packetized into one network packet. The amount of FEC added and the bit allocation are obtained through optimization.

encoding method is quite similar to PET [11] and MD-FEC [12]. Given a piece of coded video bitstream, shown at the top in Fig. 4, divided into chunks as $A, B, C, \cdots, Y$, in FGA-FEC, we further divide each chunk of bitstream into *blocks*. A smaller block size means finer granularity and hence better adaptation precision. In Fig. 4, the bitstream is divided into blocks as $(A_1, \cdots; B_1, \cdots, B_{2i}; C_1, \cdots, C_k; \cdots; Y_1, \cdots, Y_N)$, where $i \leq j \leq \cdots \leq N$ and $i, j, k, ..., N$ are determined based on network conditions and the block size. RS encoding is applied vertically across these data blocks to generate parity blocks. Each vertical column represents a data chunk divided into blocks, followed by the generated parity blocks. More FEC protection is added to the more important parts of the bitstream and less FEC is allocated to data with lower priority. The optimal allocation of FEC to different chunks of data is described in [12][13][14] and [20], as well as later in this paper. After FEC encoding, each horizontal row of blocks is one description, and in this paper, one description is equivalent to one network packet.

Similar to MD-FEC [12], FGA-FEC transforms the priority ordered bitstream from an embedded video coder into non-priority descriptions. The granularity of FGA-FEC adaptation is at block level. For instance, suppose that a DSN needs to adapt the video bitstream by dropping one piece of bitstream, say $\{C_{k-j}, \cdots C_k\}$ in Fig. 4. This can be achieved by removing the original data and FEC blocks related to $\{C_{k-j}, \cdots C_k\}$ from each network packet. Fig. 5 shows the adaptation of one FGA-FEC encoded GOP, where two blocks need to be

removed from each description of the GOP. Hence, all packets

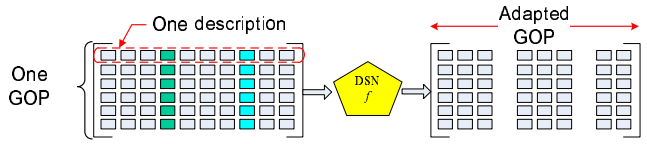


Fig. 5. Adaptation of an FGA-FEC encoded GOP, two dark blocks are removed from each description, including both original data and parity bits

(both data and parity) are shortened, and this is the only processing that needs to be done - no FEC transcoding is necessary. Further, the removed parity bits correspond precisely to the data bits that are dropped. To facilitate intermediate adaptation, an information packet is sent ahead of each GOP to tell the DSNs about the block size, FEC code, and bitstream organization, with the consumption of a very small amount of bandwidth [17].

Next, we find the optimal FEC assignment for a given scalable video bitstream. The bitstream has three types of adaptation: SNR adaptation, frame-rate adaptation, and resolution adaptation. Frame-rate and resolution adaptation can only be performed in discrete layers. For instance, a CIF video can only be adapted to QCIF video in terms of resolution adaptation. There is no continuous resolution type between CIF and QCIF. The CIF-related part of the bitstream is directly removed from the original bitstream (Fig 3), and no optimization is needed. Since the bitstream is progressively encoded, the SNR adaptation is fine granular. Then we can find the optimal solution for SNR scalability. Furthermore, whenever frame-rate adaptation or resolution adaptation is performed, protecting the adapted bitstream remains the problem of finding optimal protection in terms of quality.

Suppose we want to create N packets per GOP. Following [20] and [12], let q_i be the probability that any i out of N packets are successfully delivered. The goal is to find the optimal bitrate partition $R = \{R_1, R_2, ..., R_N\}$ in Fig. 4, which minimizes the end-to-end mean distortion $E[D(R)]$,

$$E[D(R)] = \sum_{i=0}^{N} q_i D(R_i), \tag{5}$$

subject to:

$$\begin{cases} 0 \le R_1 \le R_2 \le ... \le R_N; \\ R_{total} \le R_{max}; \\ R_i - R_{i-1} = k_i * i; \qquad k_i \ge 0 \ and \ i = 1, ..., N \end{cases}$$

where R_{max} is the available bandwidth for the channel, and $R_0 = 0$. Given a packet-loss probability p and assuming independent losses, q_i can be calculated as:

$$q_i = \begin{pmatrix} N \\ i \end{pmatrix} (1-p)^i p^{N-i}. \tag{6}$$

R_{total} is the total bandwidth (bitrate) available for both FEC and video data and can be calculated as:

$$\begin{aligned} R_{total} &= \frac{R_1}{1}N + \frac{R_2 - R_1}{2}N + \cdots + \frac{R_N - R_{N-1}}{N}N \\ &= \sum_{i=1}^{N} \frac{N}{i(i+1)} R_i = \sum_{i=1}^{N} \alpha_i R_i, \end{aligned} \tag{7}$$

where $\alpha_i = \frac{N}{i(i+1)}$ for $i = 1, 2, \cdots, N-1$; and $\alpha_N = 1$.

Finding the optimal rate break points $\{R_n, n \in [1, N]\}$ is effectively a bit allocation problem addressed in [12], [13], and [14]. For simplicity, we use a generalized BFOS algorithm [21] (Algorithm 2 below) which finds the optimal bit allocation solution by a simple search. As shown in [12], if $\alpha_i/q_i \le \alpha_{i+1}/q_{i+1}$ for some i, then in the optimal solution we will have $R_i = R_{i+1}$. Hence, R_{i+1} need not be computed - it is sufficient to optimize R_i and then set $R_{i+1} = R_i$ at the end. We therefore remove from the list $R_1, R_2, \cdots, R_N$ any such R_{i+1}, remembering its indices, and re-label the remaining variables into a new list $R_1, R_2, \cdots, R_{N'}$, where $N' \le N$.

Algorithm 2: BFOS algorithm

1) For $i = 1, 2, ..., N'$, set $R_i = R_{max}$.
2) For $i = 1, 2, ..., N'$, calculate the change in distortion.

$$\begin{aligned} (\frac{\triangle D}{\triangle R})_i &= -\frac{q_{i+1}}{\alpha_{i+1}}[D(R_{i+1}) - D(R_{i+1}+1)] \\ &\quad -\frac{q_i}{\alpha_i}[D(R_i) - D(R_i - 1)] \end{aligned} \tag{8}$$

   Let l be the index i for which $(\frac{\triangle D}{\triangle R})_i$ is minimum.
3) Set $R_l = R_l - 1$.
4) Calculate the total rate R_{total}.
5) If $R_{total} \le R_{max}$, stop. Otherwise go back to step 2
6) For $i = 1, 2, ..., N$, round down R_i to the nearest multiple of i.
7) For JSNC, we add the additional step of rounding the resulting R_i to a multiple of the block size.

More detail regarding the BFOS algorithm can be found in [21] and [22]. Now that we have the optimal bit-allocation result, the break points of the bitstream are known, and we can encode the bitstream using FGA-FEC as illustrated in Fig. 4.

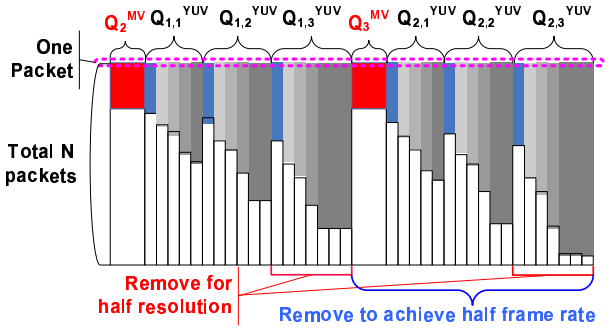


Fig. 6. FEC assignment and adaptation to different frame rate and resolution. Adaptation of SNR can be easily achieved by removing related vertical blocks from each packet. Blank colored blocks contains FEC.

To facilitate adaptation, the JSNC encoded bitstream can be rearranged, for example as shown in Fig. 6, with $L_t = 3$ and $L_s = 2$. DSNs can adapt the JSNC encoded bitstream according to the user node's video request and network conditions. For temporal adaptation and spatial adaptation, DSNs can directly remove the related part from the encoded bitstream in Fig. 6. For SNR adaptation, the DSN needs to calculate which sub-bitstreams need to be removed, and adapt each

packet as illustrated in Fig. 5. Since we shorten each packet in the same GOP by removing related blocks, both FEC and data blocks are actively removed.

D. Joint Adaptation

In this section, we state how to adapt the JSNC encoded bitstream according to user video request and network conditions. The adaptation unit is one GOP.

As mentioned already, each user node determines its video request based on user ideal video, adaptation order and network conditions. While streaming, each end user periodically estimates the available bandwidth B and packet-loss rate p to its uplink, where p can be measured based on observed packet losses, and B can be estimated using the TCP friendly equation [23],

$$B = \frac{S}{T_{rtt}\sqrt{\frac{2p}{3}} + T_{rto}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}, \tag{9}$$

where S is the packet size in bytes, T_{rtt} is the estimated round-trip time between user and its DSN in seconds, T_{rto} is the TCP timeout on this link.

User video request and adaptation order are collected and fed back to the server by the DSNs. After the user node determines its video request using Algorithm 1 (Section II-B), a 7-tuple of parameters is sent to the DSN as following:

$$\{T, S, Q, ID, \gamma, \xi, \zeta\},$$

where T, S, Q denote the user video request in temporal, spatial, and quality layers, respectively; and ID and $\gamma,\ \xi,\ \zeta$ indicate the adaptation order and user minimal requirements, cf. Table II. Each DSN aggregates the video requests of its downstream users into a $L_s \times L_t$ matrix, Θ, whose (i,j)th element ($\Theta(i,j)$) denotes the requested highest quality layer at spatial resolution i and frame rate j among all its downstream users. The Θs are aggregated along the DSNs to server. Suppose that a given DSN has two child DSNs that send their video requests Θ_1 and Θ_2, this parent DSN then produces aggregated video requests as follows:

$$\Theta(i,j) = max(\Theta_1(i,j), \Theta_2(i,j)).$$

Effectively we are streaming through the overlay network to satisfy the maximum of the user video requests. The server can then encode the video bitstream according to the aggregated user video request and loss rate (estimated using the method in [24]). Since we assume that there is no congestion between DSNs, R_{max} is chosen large enough so that R_N can accommodate the aggregated video requests.

Similar to an end user, each DSN maintains a QoS parameter vector for both available bandwidth and packet-loss rate $\{\mathbf{B}, \mathbf{p}\}$ of its direct links, where

$$\mathbf{B} = \{B_i : i \in \{0, 1, 2, ..., N_{down}\}; \tag{10}$$

$$\mathbf{p} = \{p_i : i \in \{0, 1, 2, ..., N_{down}\}\}, \tag{11}$$

and $i = 0$ is for uplink, N_{down} is the total number of its direct downlinks. These parameters are used for adaptation of the JSNC coded bitstream for each user.

While streaming, DSNs adapt the JSNC coded bitstream for their direct downlinks based on user video request, adaptation order and network conditions. Adaptation to user video request is straight forward, DSNs can directly remove the bitstream and FEC codes as shown in Fig. 5 and 6. To adapt to a lower bandwidth ($B < R_{max}$), DSNs need to do further adaptation of the bitstream by dropping descriptions (starting from description N and then $N-1, \cdots$, until $R_{total} \le B$) or shortening each packet (starting from rate break points $R_N, R_{N-1}, \cdots$, until $R_{total} \le B$) at Fig. 4 or by doing a combination of the two methods. Suppose after adaptation, there are only M ($\le N$) descriptions left with the available rate break-points $R_1, R_2, \cdots, R_K$, where $K \le M$. Then the expected distortion after adaptation is:

$$E[D(R)] = \sum_{k=0}^{K} q_k D(R_k), \tag{12}$$

subject to:

$$R_{total} \le B$$

where the probability of receiving j out of M packets is

$$q_i = \binom{M}{j} (1-p)^j p^{M-j}. \tag{13}$$

Algorithm 3: Algorithm to adapt to lower bandwidth

Input : B, N, T, S, Q, ID, γ, ξ, ζ
Output: $R_1, \cdots, R_K$, $N - i$

Adapt based on user video request, T, S, Q;
Update $R_1, R_2, \cdots, R_N$;
Loop:
Find largest M', such that $\sum_{j=1}^{M'} \alpha_j R_j \le B$, where $\alpha_j = \frac{M'}{j(j+1)}$ for $j = 1, \cdots, M'-1$, $\alpha_{M'} = 1$;
$M'' = N - M'$;
for *($i = 0$; $i \le M''$; $i++$)* **do**
  Remove descriptions N to $N - i$ in Fig. 4;
  Find R_K such that $\sum_{j=1}^{K} \alpha_j R_j = B$, where $\alpha_j = \frac{N-i}{j(j+1)}$ for $j = 1, \cdots, N-i-1$; $\alpha_{N-i} = 1$;
  Round R_K as Algorithm 2 step 7;
  Calculate q_j as (13) with $M = N - i$;
  $E[D(R)]_i = \sum_{j=0}^{K} q_j D(R_j)$;
end
$minE[D(R)] = min\{E[D(R)]_i, i = 0, \cdots, M''\}$;
if *($minE[D(R)] \Leftrightarrow maxPSNR \ge \gamma$)* **then**
  solution found, **stop**;
else
  Move down one adaptation level;
  Adapt bitstream based on this adaptation level;
  Update $R_1, R_2, \cdots, R_N$;
  goto **Loop**;
end

Given the available bandwidth B of a particular user, there might be many pairs of M and $\{R_K\}$ that satisfy the bit budget. The task of the DSN is to find a target adaptation bitstream that has the minimum $E[D(R)]$ using Algorithm 3, i.e. the DSN needs to find the best combination of dropping

descriptions and shortening packets by a search algorithm, based on user adaptation order. In detail, a DSN first adapts the bitstream to the user video request, since some subband bitstreams might be removed (as shown in Figs. 5 and 6), the rate break points $R_1, R_2, \cdots, R_N$ should be updated. If still $R_{total} > B$, the DSN needs to further adapt the bitstream to satisfy the available bandwidth, again considering the user adaptation order. At each adaptation step, the DSN first finds the search range, by iteratively removing description N and rate break point R_N, description $N-1$ and R_{N-1}, $\cdots$, until there are M' descriptions and M' rate break points left, which satisfy $R_{total} \leq B$. Thus, the maximum number of descriptions which can be dropped to satisfy the bit budget is $M'' = N - M'$. Within the search range $(0 \rightarrow M'')$, we start from $i = 0$ (no description is dropped), we search for a rate break point R_K by iteratively dropping the right-most blocks of Fig. 4 (R_K can be an original break point or a new break point between two original points), which satisfies $R_{total} = B$, to calculate $E[D(R)]_i$. Then we move to $i = 1$ (drop one description), search again for a rate break point R_K, calculate $E[D(R)]_i$,$\cdots$, until $i = M''$. Then we find the minimum distortion (min$E[D(R)]$) of this step. The process is repeated along the adaptation order, until we meet the user minimal quality requirement $maxPSNR \geq \gamma dB$. After calculation, the DSN only needs to send out $N - i$ descriptions with maximum video bitrate R_K to this user, where i corresponds to the step with min$E[D(R)]$.

Algorithm 3 can be simplified to a coarse, computationally efficient method, we call *direct truncation*, wherein the DSN adapts the JSNC coded bitstream by directly shortening each description to satisfy the available bandwidth, with no descriptions dropped ($M'' = 0$). Direct truncation could be used at DSNs that lack computational power, such as battery powered mobile nodes.

Summarizing, we have presented several building blocks of the proposed JSNC scheme, with the overall procedure shown in Table III. Steps 1 and 2 are inputs to the streaming system, while steps 3-6 are repeated at every GOP.

Overall procedure	
1.	Users decide adaptation-order, the output is ID, γ, ξ, ζ;
2.	Users decide ideal video, the output is L_t, L_s, L_q, which are inputs to Algorithm 1;
3.	User nodes determine video requests using Algorithm 1, the result is T, S, Q;
4.	DSNs collect/aggregate video requests to server;
5.	Server runs Algorithm 2 to allocate FEC and then encodes video using JSNC;
6.	DSNs adapt JSNC coded bitstream to serve users using Algorithm 3;

TABLE III
OVERALL PROCEDURE

III. SIMULATIONS AND EXPERIMENTS

JSNC distributes the video coding functions across the server and the network and can adapt both the video bitstream and error-control coding to satisfy multiple diverse users by simply adjusting the packet size and/or dropping related packets at intermediate nodes, without decoding/recoding the FEC. Several questions need to be answered about the new technique:

1) Can the in-network block-based adaptation of embedded bitstreams achieve almost the same quality as source coding?
2) Since JSNC is a generalization of MD-FEC, how does it perform compared with MD-FEC as block size varies?
3) JSNC serves multiple heterogeneous users by adapting both source- and channel-coded bitstream. Is it optimal or near optimal?
4) How does JSNC perform compared with conventional unicast streaming?

We performed simulations and experiments to show the effectiveness of our JSNC approach. We used 300 frames of the *Foreman* CIF test sequence at 30 fps, 16 frames/GOP, scalable source coder MC-EZBC, and Reed-Solomon codes used for FGA-FEC channel coding. Adaptations are done at intermediate overlay nodes using the JSNC encoded MC-EZBC bitstream. Each simulation is run at least ten times, and we present only averages for statistically meaningful results.

A. JSNC vs. Source Coding

At an intermediate DSN node, suppose we need to adapt the bitstream to match the available bandwidth of a certain downlink. The scalable source coder can generate a bitstream that exactly matches the bandwidth. But JSNC adapts the bitstream at block level, so the adaptation will not be as precise as that at the source coder. So here we focus on comparing the coding efficiency of JSNC vs. source coding, and we assume there is no FEC added.

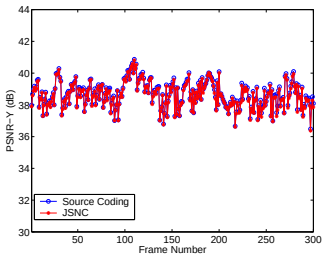


Fig. 7. MC-EZBC source coding vs JSNC for block size 8 bytes and 992 Kbps available bandwidth

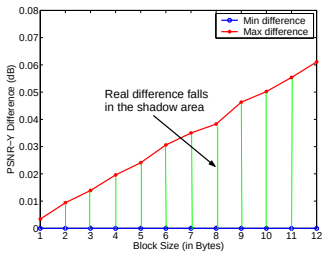


Fig. 8. Effect of block size, smaller block size means finer granularity of adaptation

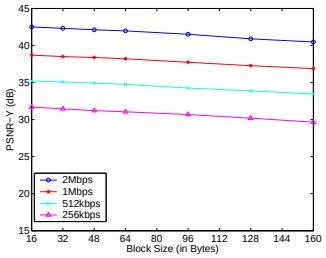


Fig. 9. Effect of larger block size at different rate

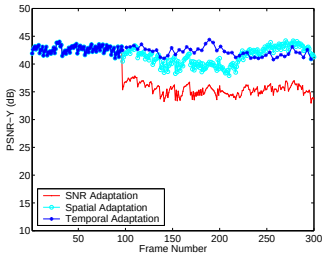


Fig. 10. Video quality of 3-D adaptation to match the available bandwidth from 2 Mbps down to 512 Kbps

Given a video bitstream at 1 Mbps, JSNC can packetize the bitstream into 128 packets per GOP in the order of SNR

scalability, with block size of 8 bytes, resulting in a network packet size of about 520 bytes. In a scenario where the last mile available bandwidth of a user is 991 Kbps, to match this bandwidth, JSNC can only adapt the 1 Mbps bitstream to 976 Kbps by removing the last 3 blocks from each packet. Fig. 7 shows PSNR-*Y* of the source coded video at 991 Kbps versus JSNC adaptation to network condition at 976 Kbps. The overall PSNR of JSNC is 0.04 dB lower than source coding in this case.

Obviously, the block size of JSNC can affect adaptation precision. In Fig. 8, we plot granularity of adaptation versus block size. Here, the last block is removed from each network packet (equivalent to removing the last column in Fig. 4). Clearly, smaller block size means finer granularity. The two curves illustrate two extreme cases. The "min difference" happens in the case where both JSNC and source coder remove the same amount of bitstream to satisfy a bit budget. The "max difference" happens at the case where JSNC removes the whole "Y" column in Fig. 4, while the source coder only needs to remove the "Y_N" block from the bitstream. In other cases, the PSNR difference falls in the lined area between these two extreme curves. Fig. 9 further shows the adaptation granularity at larger block sizes and different bitrates by removing the last block of each packet. The granularity becomes coarser as block size becomes larger.

Each user has an adaptation order to respond to dynamic network conditions. In Fig. 10 we show the corresponding video quality when the available bandwidth drops. Originally, the user is receiving a 2 Mbps, *Foreman*, CIF, 30 fps bitstream. Starting with frame 100, however, the user has only 512 Kbps available bandwidth. Here, we list three possible choices for the user: (a) SNR adaptation to 512 Kbps; (b) Temporal adaptation to 7.5 fps; (c) Spatial adaptation to QCIF. Both choices (b) and (c) need additional SNR adaptation to fit in 512 Kbps. The user can choose an adaptation order based on profiles like Fig. 10 and its own display and computing capabilities.

Results in this subsection show that JSNC can adapt the bitstream almost as precisely as can the source coder. The maximum difference is less than 0.003 dB for a block size of 1 byte in Fig. 8.

B. JSNC vs. MD-FEC

JSNC extends MD-FEC by providing additional multidimensional adaptation capabilities with both source data and FEC data, to facilitate in-network processing. JSNC coding is at block level, the difference between JSNC and MD-FEC occurs at Steps 6)-7) of Algorithm II (BFOS). For MD-FEC, R_j is rounded down to a multiple of j in bit level, but for JSNC, rounding is at block level, and usually the block size is one byte.

We compare JSNC and MD-FEC by encoding the first GOP of *Foreman* using JSNC and MD-FEC respectively. The block size of JSNC is set to 1 byte. R_{max} is 1 Mbps and the average loss rate is set to 20% for both schemes. The total number of encoded layers is 35 in both cases. Here, we refer to the piece of bitstream between two rate break points, shown at top of Fig. 4, as one layer.

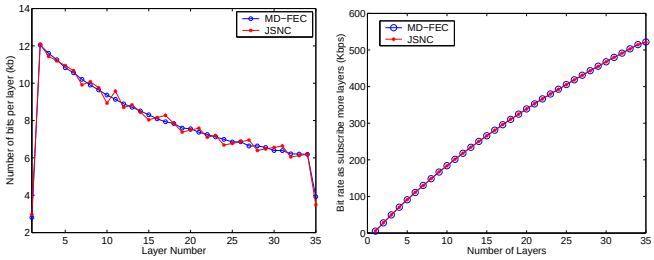
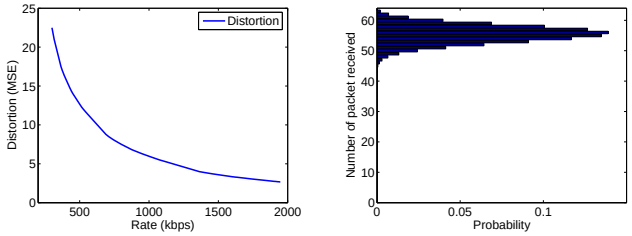


Fig. 11. Number of bits in each layer Fig. 12. Bit rate as layer adds up

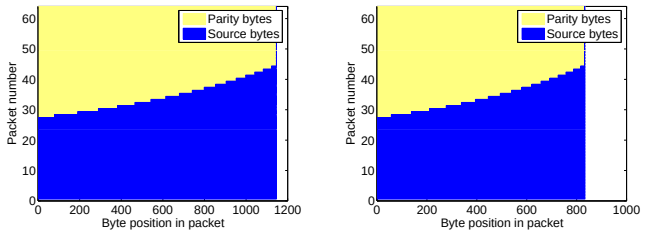
Fig. 11 compares number of bits in each layer and we found that the two schemes can generate a very similar layer size. The bit difference between JSNC and MD-FEC is due to the rounding at different precision levels. Fig. 12 shows that the two schemes generate almost the same bitrate as we move through the layers from base layer upwards.

Results in this section show that JSNC performs almost the same as MD-FEC in terms of protection and bit allocation.

C. JSNC Adaptation vs. Optimal Bit Allocation



(a) RDcurve of the seventh GOP of foreman (b) The probability of number of received packets



(c) Bit allocation 1100 Kbps, the source data and FEC data byte position in each 1145-byte packet (d) Bit allocation 800 Kbps, the source data and FEC data byte position in each 833-byte packet

Fig. 13. The operational rate distortion curve, binomial (64,0.15) distribution and matched bit-allocation result of the seventh GOP *Foreman* CIF sequence

At intermediate nodes, JSNC adapts the bitstream and FEC by shortening packets and/or simply dropping packets to serve multiple heterogeneous users. A better solution would be to optimize the bitstreams on the links for each individual user and adapt the bitstream by decoding/recoding the FEC codewords based on user-node video request, user adaptation order, user minimal requirements, and network conditions. We illustrate the better solution using the seventh GOP of the *Foreman* sequence in Fig. 13. The MC-EZBC encoded video bitstream is optimized by MD-FEC for two users with maximum available bandwidth of $R_{max} = 1100$ Kbps and 800 Kbps, respectively. The packet-loss probability p is set

to 0.15 for both users. The number of descriptions per GOP is 64. Given the rate-distortion curve of the seventh GOP of the *Foreman* sequence (Fig. 13(a)) and the distribution of the number of packets being received (Fig. 13(b)), the optimal resulting bit allocation (Fig. 13(c) and 13(d)) shows the byte position in each packet (reference to Fig. 4 for packetization). The JSNC scheme optimizes the protection based on the highest requirement user (1100 Kbps) and adapts the encoded bitstream of this GOP from 1100 Kbps to 800 Kbps by using Algorithm 3. Fig. 14 shows the performance of our JSNC scheme vs. the optimal bit allocation in terms of video PSNR for the 800 Kbps user. In this case, we have only a 0.02 dB deficit over the optimal decode/recode solution.

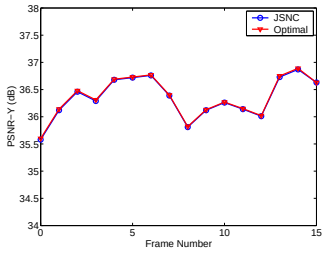
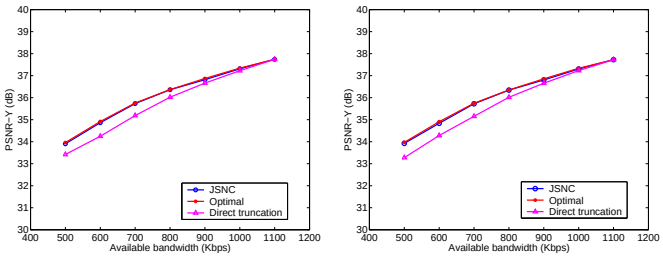


Fig. 14. Comparison of JSNC and optimal decode/encode solution to satisfy the 800 Kbps user



(a) (b)

Fig. 15. Comparison of JSNC, optimal decode/recode solution, and direct truncation at different available bandwidths; (a) against the theoretical mean distortion by calculation (b) against the video quality by simulation

We further compare JSNC, optimal decode/recode solution, and the direct truncation method at different available bandwidth, where the JSNC coded bitstream is adapted from 1100 Kbps to different rates as shown in Fig. 15. Fig. 15(a) compares the theoretical calculation results of mean PSNR of JSNC using Algorithm 3, the optimal decode/recode solution, and direct truncation. Algorithm 3 can both drop descriptions and shorten packets to achieve the best adaptation, which results in a near optimal video quality. The direct truncation method has a coarse, but still acceptable adaptation result. Fig.15(b) compares the resulting video quality of the three schemes by simulation, where the adapted bitstreams are transmitted through a channel with a 15% packet-loss rate.

The JSNC adaptation is only actively removing blocks within each packet instead of performing a complex FEC computation, so the computational burden is very low. We tested the FEC encoding/decoding time at a Pentium 4, 1.6 GHz machine running Linux 8.2. The task is to encode 115 packets with a size of 512 bytes each. To generate an RS(120,115) code, it took approximately 4 ms. We also tested the JSNC adaptation burden on the same computer, the adaptation time to process an equal number of packets was about 1×10^{-3} ms.

Results in this section show that JSNC has near optimal performance in terms of protection, but has much lower computational burden than the optimal decode/recode solution.

D. JSNC Network Performance

Conventionally, when network congestion occurs, data packets are randomly dropped at the router to avoid congestion. On the other hand, JSNC adapts the packets at the intermediate nodes to reduce the bandwidth requirement, by dropping the least important part of the bitstream. Given a 1.5 Mbps bitstream and available bandwidth of 1455 Kbps, in Fig. 16 we compare PSNR-*Y* of JSNC versus a random drop scheme with a 3% packet-drop ratio. There is no FEC added in either scheme. Observe that the proposed scheme significantly outperforms random dropping by about 10 dB. The reason for the large degradation of the random drop PSNR is the high dependency of the scalably coded video bitstream. If one packet is dropped, further packets in the same GOP become useless. Thus, the effective packet-loss rate is much higher than 3%.

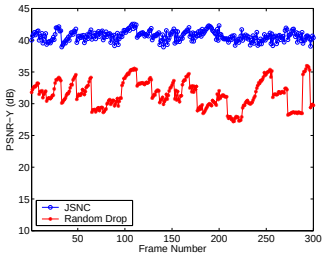


Fig. 16. JSNC vs. random drop

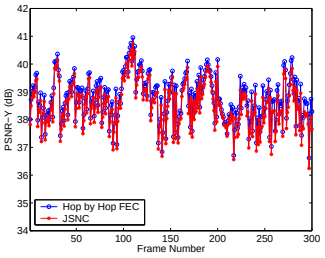


Fig. 17. JSNC vs Hop-by-Hop FEC

Next, we compare the performance of JSNC with a hop-by-hop FEC scheme. Both these schemes do adaptation at intermediate nodes. Consider the server streaming video to user "E" in Fig. 1, the available bandwidth of "E" is 1 Mbps, the packet-loss rate is $p_1 = p_2 = p_3 = p_e = 1.5\%$, and we assume that there is no bandwidth constraint between DSNs. In order to fully recover the losses, JSNC adds FEC based on the end-to-end loss rate which is approximately 6%. Meanwhile, hop-by-hop FEC needs only to protect against the 1.5% loss rate on each virtual link, so more bandwidth is allocated to video. Thus, the received video quality using hop-by-hop FEC is 0.22 dB better than using JSNC, as shown in Fig.17. But this video quality gain is achieved by much greater computational cost, since the intermediate DSN nodes need to decode/recode the FEC. As an aside, if using our OM-FEC [24] algorithm, JSNC can be engineered to have the same protection performance with hop-by-hop FEC scheme.

We further compare JSNC versus unicast in terms of video quality using the network simulator ns-2 [25] for the architecture of Fig. 18, wherein twelve users are sharing a bottleneck between nodes 3 and 4. Users 0 to 9 are requesting a scalable video from the server with the *ideal video rates* shown in

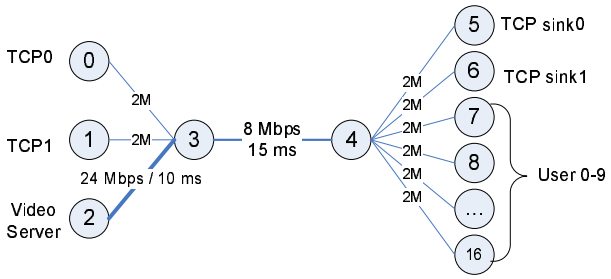


Fig. 18. ns-2 topology for comparison of JSNC with conventional unicast

Table IV, and the supporting network protocol is TFRC [23]. We have assigned FEC based on a max bandwidth of 2 Mbps and 2% of packet-loss rate.

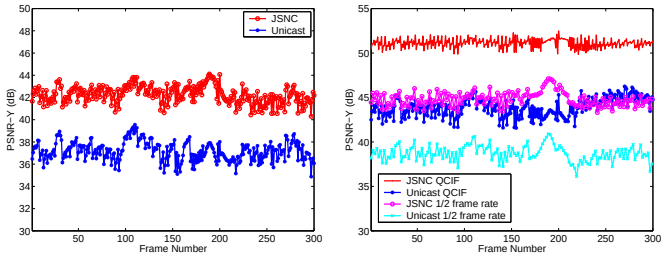
Users		Actual available rate (Kbps)	
User ID	Required rate (Kbps)	Unicast	JSNC
0	1000	750	1000
1	1100	752	1100
2	1200	753	1200
3	1300	754	1300
4	1400	754	1400
5	1500	747	1500
6	1600	752	1600
7	1700	746	1700
8	1800	748	1800
9	1900	750	1900

TABLE IV
NETWORK PERFORMANCE OF USING JSNC VS UNICAST

Due to congestion at the bottleneck in unicast, each user fairly shares the bandwidth with others, including the two TCP users. Thus, the server can only stream video to each user according to its available bandwidth (not their ideal video), shown as *actual available rate* in Table IV. Packets are actively dropped by the server according to their relative importance. If node 4 becomes a DSN node, it can adapt the bitstream to support the different users. The required bandwidth from server to node 4 is 2 Mbps in this case. The total traffic at the bottleneck is at maximum 6 Mbps (2TCPs + 1TFRC from node 4), so there is no congestion in the JSNC case. In Fig. 19, we show the captured video frames (93rd frame of *Foreman* sequence) of the 9th user in Table IV. The effective throughput is 1900 Kbps for JSNC and 750 Kbps for unicast. In this case, JSNC is objectively 5.09 dB better than unicast.

(a) JSNC (b) Unicast

Fig. 19. Sample video (93rd frame) of the 9th user in Table IV, given the available bandwidth, using JSNC (a) and unicast (b).



(a) JSNC vs Unicast (b) JSNC vs Unicast

Fig. 20. Comparison of PSNR of the 9th user: (a) JSNC vs, unicast at full frame-rate and full resolution; (b) JSNC vs. unicast at half frame-rate and quarter resolution.

Fig. 20(a) compares PSNR of both JSNC and unicast of the 9th user at full frame rate and full resolution based on the actual available bitrate listed in Table IV. Since there is not enough bandwidth for the 9th user with unicast transmission, the server can adapt the video bitstream to half frame rate or quarter resolution based on the available bandwidth and the user adaptation order to provide higher quality (PSNR) service. A considerably higher PNSR can be had as shown in Fig. 20(b), but with lower frame rate or spatial resolution.

IV. CONCLUSIONS

In this paper, we presented a joint source-network coding (JSNC) approach for scalable video streaming. JSNC encodes a scalable embedded video bitstream in such a way that both the video bitstream and the error control codewords can be easily and precisely adapted in a multidimensional way at intermediate overlay nodes to satisfy a diversity of users without complex transcoding. The adaptation at the intermediate overlay nodes is fine granular at block level. Adaptation quality is almost the same as that of 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/recoding at the intermediate nodes. Simulations and experiments show that the proposed JSNC can efficiently and precisely stream scalable video to multiple heterogeneous users. Future work will focus on cooperative adaptation between DSNs and extension to the wireless case with bit errors in the packets.

REFERENCES

[1] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *Proc. ACM SIGMETRICS*, Montery, CA, 2000, pp. 1–12.
[2] B. Raman, S. Agarwal, and et al, "The SAHARA model for service composition across multiple providers," in *Proc. of International Conference on Pervasive Computing*.
[3] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti, "CANS: Composable, adaptive network services infrastructure," in *Proc. of USENIX Symposium on Internet Technologies and Systems*, 2001.
[4] X. Gu and K. Nahrstedt, "Qos-assured service composition in managed service overlay networks," in *Proc. of International Conference on Distributed Computing Systems*, RI, May 2003, pp. 194–201.
[5] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Distributing streaming media content using cooperative networking," Microsoft Corporation, WA, Tech. Rep. MSR-TR-02-37, 2002.
[6] MPEG-21. [Online]. Available: http://www.chiariglione.org/mpeg/

[7] A. Vetro and C. Timmerer, "Digital item adaptation: overview of standardization and research activities," *IEEE Trans. Multimedia (special MPEG-21 issue)*, April 2005.

[8] D. Mukherjee, A. Said, and S. Liu, "A framework for fully format-independent adaptation of scalable bit streams," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1280–1290, Oct. 2005.

[9] D. Mukherjee, E. Delfosse, J. Kim, and Y. Wang, "Optimal adaptation decision-taking for terminal and network quality of service," *(Invited paper) IEEE Trans. on Multimedia, Special Issue on MPEG-21*, pp. 454–62, June. 2005.

[10] HP SSM. [Online]. Available: http://www.hpl.hp.com/research/ssm

[11] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Transactions on Information Theory*, vol. 42, pp. 1737–1744, Nov. 1996.

[12] R. Puri and K. Ramchandran, "Multiple description coding using forward error correction codes," in *Proc. 33rd Asilomar Conference Signals and Systems*, Pacific Grove, CA, Oct. 1999, pp. 342–346.

[13] A. E. Mohr, R. E. Ladner, and E. A. Riskin, "Approximately optimal assignment for unequal loss protection," in *Proc. ICIP*, Vancouver, BC, September 2000.

[14] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Packet loss protection of embedded data with fast local search," in *Proc. ICIP*, Rochester, NY, September 2002.

[15] P. A. Chou, H. J. Wang, and V. N. Padmanabhan, "Layered multiple description coding," in *Proc. Packet Video Workshop*, Nantes, France, April 2003.

[16] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Robust layered multiple description coding of scalable media data for multicast," *IEEE Signal Processing Letters*, vol. 12, no. 2, pp. 154–157, Feb. 2005.

[17] J. W. Woods, P. Chen, Y. Wu, and S.-T. Hsiang, *Interframe Subband/Wavelet Scalable Video Coding*, in Handbook of Image and Video Processing. Burlington, MA: Elsevier Academic Press, 2005.

[18] S.-T. Hsiang and J. W. Woods, "Embeded video coding using invertible motion compensated 3-d subband/wavelet filter bank," *Signal Processing: Image Commun.*, vol. 16, pp. 705–724, May 2001.

[19] P. Chen, S.-T. Hsiang, J. Woods, D. Mukherjee, G. Kuo, and A. Said, "Fully scalable mc-ezbc in the structured scalable meta-formats (ssm) framework," in *ISO/IEC JTC1/SC29/WG11 MPEG2002/M9290*, Awaji, Japan, Dec. 2002.

[20] I. V. Bajic and J. W. Woods, "EZBC video streaming with channel coding and error concealment," in *SPIE VCIP*, July 2003, pp. 512–522.

[21] E. A. Riskin, "Optimal bit allocation via the generalized BFOS algorithm," *IEEE Transactions on Information Theory*, vol. 37, no. 2, pp. 400–402, March 1991.

[22] I. V. Bajic, "Robust subband/wavelet coding and transmission of images and video," Ph.D. dissertation, ECSE Dept., Rensselaer Polytechnic Institute, Troy, NY, 2003.

[23] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," ICSI, Tech. Rep. TR-00-03, March 2000.

[24] Y. Shan, I. V. Bajic, S. Kalyanaraman, and J. W. Woods, "Overlay multi-hop FEC scheme for video streaming," *Signal Processing: Image Commun*, vol. 20, no. 8, pp. 710–727, 2005.

[25] ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/