Community Streaming With Interactive Visual Overlays: System and Optimization

Wai-Tian Tan, *Member, IEEE*, Gene Cheung, *Senior Member, IEEE*, Antonio Ortega, *Fellow, IEEE*, and Bo Shen, *Senior Member, IEEE*

***Abstract*—Community streaming is an enhanced form of joint content viewing where a sense of community is reinforced by the addition of interactive visual overlays, controlled in real-time by viewers, on top of a shared video stream. As a concrete example, we describe a community video system called ECHO, where personalized avatars are overlaid on top of a real-time encoded video stream of an Internet game for multicast consumption. Recognizing that only the visual overlays are generated live, we propose schemes that encode and schedule the live and non-live portions of the overlaid video separately in order to exploit the difference in delay sensitivity of the two, leading to video streams that contain two sub-streams with different delay constraints. We show that, in the known channel case, a low complexity "earliest deadline first" packet scheduling algorithm minimizes receiver buffer delay. We also analyze the case where multiple streams are multiplexed, which allows us to quantify the potential gains of allowing different delay constraints for different sub-streams. We show that a "water filling" strategy maximizes the total number of streams that can be supported. Simulation results show that the bandwidth necessary to maintain low-latency for visual overlays is reduced by about 40% when our proposed sub-stream approach is used. For multiplexing of multiple streams, our approach can increase the number of supported streams (e.g., a 30% increase when around ten streams are multiplexed).**

***Index Terms*—Video coding, video streaming.**

I. Introduction

VIDEO conferencing and buffered playback of streaming video are two important classes of networked video applications with different purposes. The goal of video conferencing is interpersonal communication, while streaming playback applications typically involve passive consumption of non-interactive video content. Individual viewing of video with no interpersonal interactivity is sometimes called *individual streaming* [1]. While communicating to associates and viewing video are distinct objectives, they are not mutually exclusive; in cases of live sporting or gaming events, one may prefer to watch the same streaming video with other viewers while communicating with them at the same time. We call such a model *community streaming*. It is similar to watching TV with family members in the living room and sharing comments on the viewed content, except the viewers may not be in the same physical location. It is unlike video conferencing in that the primary activity is content viewing.

There are many ways of creating an interactive community streaming experience, e.g., enabling voice chat for viewers of streaming video. In this paper we focus on techniques to enable *interactive visual overlays* (IVOs) of text, images, or video on top of the content being streamed, such that each overlay, controlled by a viewer and visible to the entire streaming group, is reflective of the viewer's comments, mood or presence. We believe that the general notion of supporting IVOs is of wide interest and applicable to many applications. For example, while watching a game show, viewers can share and compare their answers with other group members. Another example is a shared "whiteboard" on a background of streaming pre-encoded video, where a viewer can control the playback and visually highlight different parts of the video with an electronic pen.

While there are asynchronous means such as web-based forum to enhance a sense of community, IVOs enable real-time community-based inter-personal communication; IVOs are often at-the-moment representations of viewers' presence or thoughts, and must be delivered in a timely manner for them to be valuable. It is well known that the delay constraints are much more stringent for interactive than for non-interactive videos, and thus support of interactive video may require a more costly transport infrastructure (e.g., higher bandwidth may be needed to minimize end-to-end delay). Thus, a key technical challenge is to support a high degree of interactivity, without significant increases in system cost relative to a system that can only support non-interactive streaming.

The primary focus of this paper is to develop a video streaming system and optimization techniques to exploit the heterogeneity in transmission delay constraints for the interactive and non-interactive portions of the video content. There are many other important aspects such as user interface design that are essential to a successful community streaming application, but are outside the scope of this paper.

One approach to enable interactive visual overlays for community streaming is to employ multiple video streams: one stream for the main presentation, and one for each IVO. These streams are encoded and delivered independently, and each user is responsible for decoding, joint rendering and synchronizing the streams. Clearly, the non-interactive main presentation can

Manuscript received May 31, 2008; revised March 18, 2009. First published May 02, 2009; current version published July 17, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lexing Xie.

W.-T. Tan is with Hewlett Packard, Palo Alto, CA 94304 USA (e-mail: waitian.tan@hp.com).

G. Cheung is with Hewlett-Packard Laboratories Japan, Tokyo, Japan.

A. Ortega is with the University of Southern California, Los Angeles, CA 90089 USA.

B. Shen is with vuclip.com, Milpitas, CA 95035 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2009.2021797

be treated appropriately and differently from the interactive IVOs, since they belong to distinct streams. Clients with such capabilities already exist, e.g., SMIL-capable clients. Nonetheless, this multiple-stream solution has several drawbacks. First, it requires sophisticated clients capable of handling multiple simultaneous video streams, which is beyond the capability of many smartphones. Second, this approach may not be using bandwidth efficiently, since visual information occluded by overlays is also transmitted.

Instead, in this paper we consider a single-stream approach where the interaction among the users leads to *modifications in a single shared stream*. In so doing, community streaming can be extended to simpler but ubiquitous clients that can only handle single streams. This approach also avoids inefficiency in bandwidth usage by transmission of occluded regions. Nevertheless, treating each video frame in the shared stream with homogeneous delay requirement would require that all video data be transmitted as interactive traffic. Instead, we present a system optimized so as to accommodate different delay constraints for different regions of the *same video frame* in the shared stream. This allows us to exploit the inherent heterogeneity in transmission requirements of community streaming without resorting to the use of multiple streams.

Generally, IVOs should be allowed to move only in regions that do not significantly obscure the main presentation. In the example of Fig. 3, the IVOs can only move horizontally in the bottom of the frame, even though in general, the location can change on a per scene basis. This locality of movement is important, since it allows us to separate the video into regions with and without IVOs and treat them separately. Specifically, the spatial portion without IVOs can be prefetched or transmitted ahead of time, while the portion with IVOs is generated live and transmitted in a low-latency fashion. We need a compressed video stream where part of a frame can be prefetched while the other part may be modified later in unpredictable ways. To achieve this we propose to separately encode the two regions with and without IVOs so that there are no coding dependencies between them. This way, one of them can be modified without affecting the bit-rate and reconstruction quality of the other. We call such video stream *separable*, and we will show that this kind of "separability" can be easily achieved within standard video codecs.

We also propose and evaluate packet scheduling algorithms for streams containing two or more sub-streams with different delay constraints. We distinguish between two different scenarios. For the deterministic case, where transmission is over a dedicated channel with known bandwidth, e.g., an ISDN line, we show that a low complexity earliest deadline first (EDF) algorithm can guarantee minimum receiver buffer delay. Thus our multiple sub-stream approach does not impose significant complexity overhead on the servers. For the stochastic case, where channel bandwidth is not known, or when multiple sessions are multiplexed, we show that a "water-filling" strategy maximizes the number of streams that can be supported.

To provide a practical system example of community streaming, we describe a streaming system called *ECHO (Enabling Community of Hecklers and Observers)* that can efficiently support multiple communities of viewers with visual overlays, with the primary streaming video being live Internet games. (Here, the IVOs and their users are figuratively called heckles and hecklers.) A fully functional prototype of ECHO has been developed on top of HP's OpenCall Media Platform (OCMP) version 4.0, using a live game sequence from the popular *Counter-Strike* [2] game as the streaming video source. We have tested our prototype with standard video clients on notebook and handheld devices in real network environments.

The outline of the paper is as follows. We first discuss related work in Section II. We then give an overview of the ECHO system in Section III, which helps us highlight system architecture and coding trade-offs. In Section IV, we derive and prove the optimality of the EDF scheduling algorithm for transmission of a single-stream over a known channel. In Section V, we discuss multiplexing gain using separable streams over non-separable streams, and prove that the optimal scheduling strategy is "water-filling". Results that quantify different trade-offs and conclusions are provided in Sections VI and VII, respectively.

II. Related Work

Advances have been made to improve consumption model of streaming video. To address the heterogeneous capability of different handsets and preference of different users, the notion of "collaborative streaming" has been introduced [1], where heterogeneous devices may each receive the stream in the same playback time but in their own preferred resolution, bit-rate and language. Device discovery, session initialization, and session mobility are further discussed in [3]. Community streaming with IVOs advances a different aspect of the traditional usage model of streaming video by supporting personalized customization on a per group basis.

Two popular arenas that can benefit from community streaming are 3-D virtual worlds and network gaming. Recent years have witnessed the growth of both 3-D virtual worlds [4], [5] and network gaming [2], [6], particularly in Japan and Korea. Along the way, notable players have developed large followings, and *virtual world observation*—as opposed to active participation—has become a popular pastime for many. Both general video portals such as YouTube [7] and game-specific portals such as Half-Life TV [8] and StreamMyGame [9] provide videos captured during game playing. This has motivated our work on optimizations for graphics-to-video encoding and streaming of virtual world observation for both single-view [10], and multi-view video [11]. In the context of community streaming with IVOs, our earlier work [12] proposed the use of the separable video in the gaming context, and provided empirical results for the case of a single stream over a known channel. In this paper, we significantly extend the scope and generality of our earlier work to include: 1) formal derivation of an optimal scheduling algorithm for the single-separable-stream case; and 2) consideration of multiplexing of multiple streams with derivation of the optimal scheduling algorithm.

The advantages of variable bit rate (VBR) video encoding are well known, but come at the cost of additional end-to-end delay and/or bandwidth for delivery over networks [13]. Thus, streams with low delay requirements need to be encoded under low delay constraints (i.e., overall video quality) or require a

VBR channel for transmission (or alternatively a higher bandwidth CBR channel to accommodate rate variations without additional delay). Analysis of delay-bandwidth-quality trade-offs has been considered from a number of different perspectives, including effective bandwidth [14], smoothing [15], and rate control [16]. For the most part this past work focuses on the properties of a single media source, for which bandwidth usage, quality and end-to-end delay are evaluated. A key novelty of the present work is to consider scenarios where different end-to-end delay characteristics coexist within a given stream.

The concept of delay-cognizant video coding (DCVC) is proposed in [17] and [18], where different delay constraints are applied to different parts of an encoded video stream. DCVC however differs from our approach in two fundamental aspects. First, DCVC sub-streams are separated based on the video characteristics, e.g., high delay blocks in the video are those where video contents change slowly. Instead, our high delay segments will contain non-interactive portions that are not modified to incorporate visual overlays. Second, DCVC data for a given frame can be played back at different times (e.g., high delay packets can be used even after the frame they correspond to has been displayed). Instead, we follow a more traditional delay constraint, where packets for one frame have to be available before the frame is decoded in order to be useful. Thus, in our case, larger end-to-end delay is made possible by starting transmission of the non-interactive video portions early, but both non-interactive and interactive video portions are played back synchronously. Our approach is also compatible with common video compression standards.

Towards determination of the number of streams that can be supported at the same quality of service and bandwidth under statistical multiplexing, many interesting results have been developed in the effective bandwidth literature [19]–[21]. These works established a mathematical framework to study statistical multiplexing when the offered traffic is given. Using these results, our contribution is to propose an optimal scheduling algorithm that exploits the flexibility of transmission times associated with separable streams.

III. Community Streaming Systems Based on Separable Streams

The exact details of different community streaming applications are likely to differ significantly. Nevertheless, we believe that separable streams with multiple delay constraints can be a key ingredient in designing efficient community streaming for many applications. In this section, we first present a concrete community streaming system ECHO we have developed for live game watching, and show how separable streams can be supported based on common components in typical media streaming architectures. We then describe specific coding techniques that can be used to generate separable video.

A. ECHO System Overview

In our ECHO game watching prototype (see Fig. 1), a networked game is hosted by a *game server*, and typically involves two or more game players who control the *game clients*. Game server and clients typically communicate in proprietary protocols. To support game viewing for non-players

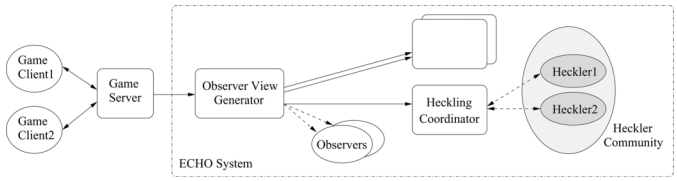


Fig. 1. ECHO system renders live game play for viewing by passive game "Observers" or active "Hecklers".

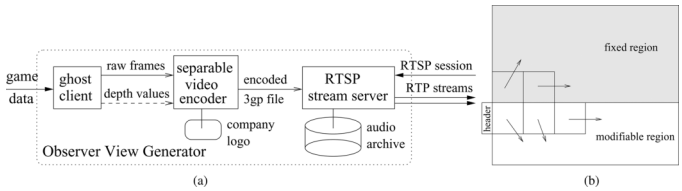


Fig. 2. Component view and video coding of Observer View Generator. (a) Component view of OVG. (b) Fixed and modifiable regions.

Fig. 3. Sample screenshots of interactive visual overlays on top of game content *Counter-Strike*.

over a wide range of devices, the game content is converted to standard-compliant video streams by the *observer view generator* (OVG) [10]. The *ghost client* inside OVG receives real-time gaming data from the game server and renders frames of the game from a chosen perspective. These frames are then encoded in real-time by a "separable video encoder" (see Section III-B). The real-time streaming protocol (RTSP) server supports game *observers* without community-style interaction by direct transmitting audio and video data to them using real-time protocol (RTP). Alternatively, a group of *hecklers* can contact a Heckling Coordinator (HC) to initiate a heckling session. The HC in turn requests the live encoded stream from OVG, composites personalized IVOs on top, as shown in Fig. 3, and transmits the enriched media to members of the newly formed community. Each user can control his/her heckles by sending commands in real-time.

The HC receives real-time input from the users and constructs a new video with overlays corresponding to the heckles. A straightforward implementation of HC would employ full video decoding, compositing of heckles in pixel domain, follow by full video encoding. Two unique characteristics of our particular video content call for a different implementation. First, IVOs tend to occupy only a small portion of the screen as the primary activity is game observation. Re-encoding parts of the video that are modified by the IVOs is wasteful in computation resources, and leads to lower visual quality in general. Instead, we employ *partial transcoding* to only process the necessary part of a video stream. Second, the game content is not dependent on user input, and can be buffered or delayed at HC without increasing perceived latency at the

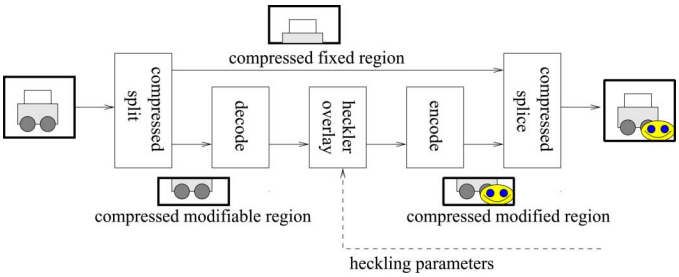


Fig. 4. *Heckling Coordinator* employs partial transcoding in which independent sub-streams are obtained from a separable video source. This allows differential treatment in transport and avoids unnecessary re-encoding.

users. The user action, on the other hand, is interactive. To exploit the difference in delay constraints for different parts of the content, separable video is produced by OVG, which allows easy separation of compressed bitstreams into independently decodable regions (see Section III-B).

The heckles are controlled by dual-tone multi-frequency (DTMF), which extends usage of the service to generic video-streaming capable cellular phones. As shown in the *heckler overlay* unit of Fig. 4, the overlay operation is performed in the pixel domain in a portion of the video frames. Pixel domain operations support a rich set of user actions such as changes in avatar location, speed, size, and opacity.

Several characteristics of the ECHO system are worth noting. First, encoding of the live game content is performed only once, even with multiple spurred heckling sessions, each coordinated by a different HC. This translates to large computation savings. Second, only a game watching application is described, but the system is generally applicable to other content such as sporting broadcast. Third, no software download and installation is necessary for client devices, as standard protocols are being used for streaming and avatar control.

B. Coding for Separable Video Streams in ECHO

The video format for ECHO is H.263, selected over more recent standards such as H.264/AVC [22] for two reasons. First, most existing video handsets support H.263 while H.264 is only supported by high-end devices. Second, encoding complexity for H.264/AVC is drastically higher than H.263, limiting the cost effectiveness of an H.264 implementation. However, the techniques described here are applicable to H.264. Our recent work [23] also suggests methods to reduce H.264/AVC encoding complexity by taking advantage of available depth information.

Since the primary goal of a user is game watching, with community interaction being a secondary objective, it is natural to assume that the IVOs will occupy a relatively small portion of the screen, as illustrated in Fig. 3. For ease of discussion, we assume IVOs are overlaid on several bottom macroblock (MB) rows of a video.[1] The video content is encoded into two independent or *separable regions*: a *fixed region* that is comprised of the top several MB rows, and a *modifiable region* consisting of the remaining MB rows. MBs in the fixed region are encoded so that their motion vectors cannot cross into the modifiable region,[2] as shown in Fig. 2(b). A Group of Block (GOB) header (part of H.263 syntax) is inserted before the first MB row in the modifiable region to facilitate identification of the region during bitstream parsing. Doing so allows a downstream HC to only search for and decode the modifiable region, *without* decoding the fixed region. The resulting stream is compliant to baseline H.263, and can be properly decoded by any standard clients without knowledge of the separable regions. It should be noted that independently decodable regions in a single compressed video stream is not a new concept. In particular, explicit syntax support is provided by independent segment decoding in Annex R of H.263 version 2, which is less commonly supported.

The HC, shown in Fig. 4, performs simple bitstream parsing (without decoding) to split the compressed bitstream into the fixed and modifiable regions. Decoding, rendering of IVOs, and subsequent encoding are only performed on the modifiable region, which reduces the computing load. Although the use of separable video generally incurs a penalty in compression efficiency, avoiding the re-encoding of the fixed region can lead to overall higher video quality at the user (more details in Section VI).

Note that there are alternative methods to realize separable streams for community streaming. One example is object-based encoding methods of MPEG-4. Another example is the *redundant slice* feature of H.264 which allows multiple representations of a MB [22]. While redundant slices are mainly employed for error resilience purposes, with matching decoder support, it is theoretically possible to arrange the redundant slices to correspond to the interactive visual overlays. The advantage of these approaches is the flexibility that visual overlays can now be spatially located anywhere in the frame. The disadvantage is high encoding and decoding complexity, as well as potentially lower compression efficiency due to the need to transmit occluded regions.

IV. Transport Scheduling for a Single Separable Video Stream

Consider a separable video stream with two sub-streams with different transmission delay requirements. Since the IVOs do not affect the non-interactive portions, the non-interactive substream can begin transmission ahead of time during an initial buffering period in high-latency mode. Even sizeable delay on the order of seconds is acceptable, with the only limitation of not introducing excessive start-up latency; we call this initial delay before session startup for data buffering the *initial buffer delay*. On the other hand, portions that are being modified in response to actions by an active user need to be conveyed to all active users quickly. We call *response lag* the delay between the time of a user's action and the time the affected visual overlay is displayed on the user's screen.

While separable video has advantages for cases where packet losses occur (e.g., retransmissions are possible for the high

[1]In general the region reserved for overlays could be anywhere and indeed could change over time. Clearly, our proposed technique will be more advantageous when the overlay region is relatively small, so that most content can be delivered under high latency conditions. This is likely to be the case in practice, since enabling overlays throughout the frame would interfere with the viewing of the main streaming content.

[2]These restrictions on the MVs can potentially reduce in coding performance. This will be examined in Section VI.

latency sub-stream), we focus here on the lossless case. As noted in Section II, better video quality can be achieved when operating the encoder in VBR mode [13]. For non-separable streams, in order to transmit VBR encoded video over a known constant bit-rate channel, one needs to enable rate averaging by either: 1) averaging the rate needs over time using receiver buffering, and/or 2) sharing the constant rate link with several VBR streams, so as to achieve statistical multiplexing. For the single-stream case, the receiver buffer required for averaging VBR video can be sizable, which would lead to a large receiver buffer delay and thus an undesirably long response lag for visual overlays. Alternatively, increased interactivity could be achieved by raising the channel bandwidth, which would increase the overall system cost as compared to a non-interactive system.

When separable streams are used, however, one can achieve better performance (lower delay and/or reduced bandwidth for the same video quality) by taking advantage of the different delay requirements of the sub-streams. Buffering can be used for smoothing the rate variations in the non-interactive stream, i.e., transmission of this stream can start early with a large initial buffering period. Conversely, delay for the interactive sub-stream can be kept low by allowing it to use a large percentage of shared bandwidth during its high bit-rate periods.

While this is intuitively possible, it is less clear whether achieving these multiplexing gains across sub-streams requires complex scheduling algorithms. In this section, we derive an optimal, low complexity scheduling algorithm called *earliest deadline first* (EDF) for two sub-streams with different delay requirements within the same separable stream over a known channel. Using EDF, we can find the minimum response lag required given an initial buffer delay, thus optimizing the tradeoff between the two. We will show in Section VI that the response lag required for the separable-stream approach using EDF is significantly smaller than the non-separable-stream approach.

A. Application Requirements

We first formally define the scheduling problem for separable streams: given initial buffering delay and response lag, how to schedule transmission of non-interactive and interactive sub-streams (assuming a feasible schedule exists) so that receiver buffer underflow is avoided? How can we determine whether a feasible schedule exists? We show that EDF provides definite answers to both questions. Thus, for a given pre-encoded stream, we can use EDF to find the optimal tradeoff between initial buffer delay and response lag.

In a general setting, we consider streaming from a sender two sub-streams with heterogenous delay requirements that must be played back synchronously at a receiver. More precisely, consider non-interactive and interactive sub-streams X and Y, with bit-rate profiles $f_X[n]$ and $f_Y[n]$, respectively; i.e., the number of bytes in frame n in sub-stream $X, n = 0, \ldots, N-1$, is $f_X[n]$. To deterministically guarantee no receiver buffer underflow for given initial buffer delay and response lag, we assume sender knows entire profiles $f_X[n]$ and $f_Y[n]$ prior to transmission. For stored content, non-interactive profile $f_X[n]$ is readily available. We also observe that interactive profile $f_Y[n]$ (which changes in real-time as users interact via IVOs) closely follows the original

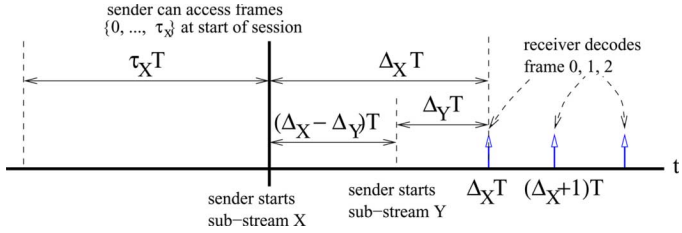


Fig. 5. Receiver time-line of multiple sub-stream system.

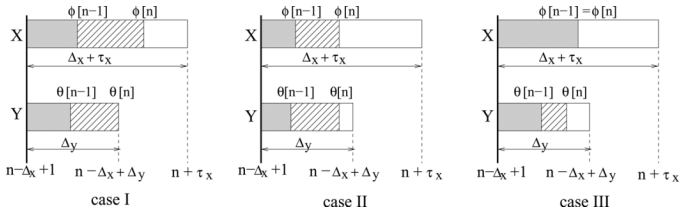


Fig. 6. Three possible cases of sender buffers under EDF, where shaded, striped, and white areas correspond to data transmitted earlier, data being transmitted in current interval $[nT, (n+1)T)$, and available data not transmitted yet, respectively.

profile $f_Y^o[n]$ of the modifiable region without IVOs when the same quantization parameters are used. Thus we can approximate $f_Y[n]$ by $f_Y^o[n]$ to determine appropriate choices of initial buffer delay and response lag. After initial buffer delay and response lag are determined *a priori*, scheduling will be based on $f_X[n]$ and actual $f_Y[n]$ during actual transmission.

To simplify discussion, in what follows we will refer to the frames being delivered in each sub-stream, even though each of these "frames" contains only part of the original video frame: two corresponding frames (one per sub-stream) will be combined to provide a single output video frame.

We develop our formulation from the perspective of the sender's clock, with the sender starting streaming video data to the receiver at time $t = 0$. We first assume video frames are captured and played back every T seconds. We assume also that the receiver starts decoding $\Delta_X T$ seconds after the sender starts sending frames. $\Delta_X T$ is the *initial buffering delay*.

Suppose now that the time at which frame n in sub-stream X is available for transmission at the sender, $n \geq 0$, is $(n - \tau_X)T$; i.e., at time nT the sender can access frames $0, \ldots, n + \tau_X$. We call τ_X the *read-ahead interval* for sub-stream X. Frames in sub-stream Y have a tighter delay constraint; i.e., they are available for transmission at sender later than the same frames in X, so that there is a shorter time interval available for delivery. More precisely, at time nT, the sender can access frames $0, \ldots, n - \Delta_X + \Delta_Y$ of sub-stream Y, where $\Delta_Y < \Delta_X$. Smaller Δ_Y means IVOs will be played back sooner. Thus, while there are other factors that affect the *response lag*, we will simply call $\Delta_Y T$ the response lag. See Fig. 5 for an illustration of the sender's time-line.

Let the *sender buffer* of sub-stream $X, B_X^s[n], n \geq 0$, be the amount of data in X that sender has available but not delivered yet at time nT. Let $r_X[n], n \geq 0$, be the amount of data in X sender decides to transmit in time interval $[nT, (n+1)T)$; the selected data $r_X[n]$ could be part of one frame, an entire frame, or data from multiple frames. Similar quantities $B_Y^s[n]$

and $r_Y[n]$ are defined for sub-stream Y. We can then write the respective sender buffer occupancies as the difference between total available video data and the amount already transmitted:

$$B_X^s[n] = \sum_{i=0}^{n+\tau_X} f_X[i] - \sum_{i=0}^{n-1} r_X[i], \quad \forall n \geq 0$$

$$B_Y^s[n] = \sum_{i=0}^{n-\Delta_X+\Delta_Y} f_Y[i] - \sum_{i=\Delta_X-\Delta_Y}^{n-1} r_Y[i] \quad \forall n \geq \Delta_X - \Delta_Y$$

where data transmitted at a given time interval cannot exceed the amount of data available in the buffers.

At receiver, playback of *both* sub-streams starts after the initial buffer period at time $t = \Delta_X T$ before which data simply accumulates in the *receiver buffer*. Assuming the transmission channel has zero delay, the receiver buffer occupancies $B_X^r[n]$ and $B_Y^r[n]$ for the two sub-streams at time nT are obtained by subtracting the total amount of data played back from the total amount of data received as shown in the equation at the bottom of the page. To avoid *receiver buffer underflow*—case where receiver experiences playback interruption due to late data arrival, receiver buffer occupancy must be nonnegative at all times:

$$B_X^r[i] \geq 0 \quad \forall i \geq 0 \tag{1}$$

$$B_Y^r[i] \geq 0 \quad \forall i \geq \Delta_X - \Delta_Y \tag{2}$$

which also guarantees synchronous playback of both sub-streams at the receiver.

B. Problem Definition

We now assume a VBR channel with average bandwidth C per time period T, and where an instantaneous bandwidth of $c[n]$ bytes over time period $[nT, (n+1)T)$ is permitted for transmission of both sub-streams. The maximum available bandwidth $c[n]$ cannot be exceeded at any frame period:

$$r_X[i] + r_Y[i] \leq c[n] \quad \forall i \geq 0. \tag{3}$$

We further assume the average bandwidth is no smaller than the average bit-rate of both sub-streams:

$$C \geq \frac{\sum_{i=0}^{N-1} f_X[i] + f_Y[i]}{N}. \tag{4}$$

$c[n] = C, \forall n$ is the special case of a CBR channel.

Given a VBR bandwidth $c[n]$, bit-rate profiles $f_X[n]$ and $f_Y[n]$, read-ahead interval τ_X and initial buffer delay $\Delta_X T$ and response lag $\Delta_Y T$, a *feasible transmission schedule* for both sub-streams X and Y is the set of scheduled transmissions $r_X[i], r_Y[i], i \geq 0$ such that no receiver buffer underflows according to (1) and (2), and that the bandwidth constraint (3) is never violated. Our goal is to compute a feasible schedule (if one exists) given the problem parameters.

C. Earliest Deadline First (EDF) Transmission Schedule

We present a transmission schedule named EDF that is guaranteed to be feasible as long as the space of feasible schedules is non-empty. In other words, if EDF is infeasible, then no feasible schedule exists for given problem parameters. We call this property of EDF *minimally feasible*. EDF operates as follows.

Earliest Deadline First Schedule (EDF):

> At each instant nT, send data of the available frame in either sub-stream X or Y with the earliest playback deadline (smallest frame index). If both sub-streams contain frames of the same smallest index, select first data of frame in sub-stream X[3]. Repeat until budget $c[n]$ has run out.

Properties of EDF: We next discuss three properties of EDF that provide insights into EDF and are needed in subsequent proof of EDF's minimal feasibility. Denote by $\phi[n]$ and $\theta[n]$ the largest indices of complete frames that have been transmitted in sub-streams X and Y, respectively, under EDF by time interval $[nT, (n+1)T)$:

$$\phi[n] = \max_{m \in \mathcal{I}} \{m\} \quad \text{s.t.} \quad \sum_{i=0}^{m} f_X[i] \leq \sum_{i=0}^{n} r_X[i]$$

$$\theta[n] = \max_{m \in \mathcal{I}} \{m\} \quad \text{s.t.} \quad \sum_{i=0}^{m} f_Y[i] \leq \sum_{i=\Delta_X-\Delta_Y}^{n} r_Y[i].$$

By definition, $\phi[0] \leq \phi[1] \leq \cdots \leq \phi[n]$ and $\theta[0] \leq \theta[1] \leq \cdots \leq \theta[n]$. If the first frame of sub-streams X and Y has not been completely delivered yet, we set, respectively, $\phi[n] = -1$ and $\theta[n] = -1$.

First, EDF has a *work-conserving* property at any *time interval* $[nT, (n+1)T)$, meaning it always transmits if there is available data. As a corollary, EDF sends the maximum amount of data possible over any *time duration* $[0, (n+1)T)$.

Second, under EDF, the largest transmitted frame index for non-interactive sub-stream X is always at least as large as that for interactive sub-stream Y:

$$\phi[n] \geq \theta[n] \quad \forall n \geq 0.$$

This follows directly from the fact that data for the same frame are available earlier in X than Y, and that the tie-breaking rule

[3]It can be shown that EDF is also minimally feasible if the tie breaking rule is in favor of sub-stream Y instead.

$$B_X^r[n] = \begin{cases} \sum_{i=0}^{n-1} r_X[i], & 1 \leq n < \Delta_X \\ \sum_{i=0}^{n-1} r_X[i] - \sum_{i=0}^{n-\Delta_X} f_X[i], & \forall n \geq \Delta_X \end{cases}$$

$$B_Y^r[n] = \begin{cases} \sum_{i=\Delta_X-\Delta_Y}^{n-1} r_Y[i], & \Delta_X - \Delta_Y + 1 \leq n < \Delta_X \\ \sum_{i=\Delta_X-\Delta_Y}^{n-1} r_Y[i] - \sum_{i=0}^{n-\Delta_X} f_Y[i], & \forall n \geq \Delta_X \end{cases}$$

favors X. More precisely, the sender can choose from $\{n - \Delta_X + 1, \ldots, n + \tau_X\}$ and $\{n - \Delta_X + 1, \ldots, n - \Delta_X + \Delta_Y\}$ for X and Y, respectively, where $\tau_X \geq 0$ and $\Delta_X > \Delta_Y$.

Third, the three cases below, as illustrated in Fig. 6, completely describe the possible states EDF is in:

1) **Case I**: $\theta[n] = n - \Delta_X + \Delta_Y$. Sender has transmitted all available frames in sub-stream Y.
2) **Case II**: $\theta[n] < n - \Delta_X + \Delta_Y$ and $\phi[n] \leq \theta[n] + 1$. Sender has available data in sub-stream Y, and largest transmitted frame index in X is within one plus largest transmitted frame index in Y.
3) **Case III**: $\theta[n] < n - \Delta_X + \Delta_Y$ and $\phi[n] > \theta[n] + 1$. Sender has available data in sub-stream Y, and largest transmitted frame index in X is strictly larger than one plus largest transmitted frame index in Y.

It is obvious that the three cases partition the space of possible states. Using these three properties, we now prove by contradiction that EDF is indeed minimally feasible.

D. Proof of EDF's Minimal Feasibility

Suppose EDF is not feasible and leads to its first receiver buffer underflow at time $(n+1)T$. Let Π' be a feasible schedule that does not lead to underflow at $(n+1)T$. First, EDF cannot be in case I at time nT. Under case I, EDF has transmitted all available frames in sub-stream Y by interval $[nT, (n+1)T)$. Given $\phi[n] \geq \theta[n]$, EDF has transmitted at least as many frames in sub-stream X, and hence receiver buffer underflows at time $(n+1)T$ (at least) in sub-stream Y. Since Π' cannot transmit more frames in Y than available, Π' cannot avoid underflow of sub-stream Y at $(n+1)T$.

Suppose EDF is in case II at time nT. Due to the work-conserving property of EDF, Π' cannot transmit strictly more total data than EDF, hence Π' cannot avoid receiver buffer overflow at time $(n+1)T$ simply by sending more data than EDF. Moreover, case II ($\phi[n] \leq \theta[n] + 1$) means that the two largest transmitted frame indices from two sub-streams, $\phi[n]$ and $\theta[n]$, are within one of each other. Due to the earliest deadline policy, presence of available data in sub-stream Y for frame $\theta[n] + 1$ means that no data corresponding to frame $\theta[n] + 2$ have been transmitted for sub-stream X. Thus, any reallocation of bandwidth from one sub-stream to the other would not increase the smaller of the two indices. That means that Π' cannot avoid reciever buffer overflow by scheduling data differently in sub-streams for the same total amount of sent data. Hence Π' will also lead to receiver buffer underflow—a contradiction.

Finally, suppose EDF is in case III at time nT. Due to the earliest deadline policy, $\phi[n] > \theta[n] + 1$ and presence of available data in sub-stream Y means that there is no transmission of data in sub-stream X during $[nT, (n+1)T)$ under EDF. Denote by $k < n$ the time-slot when sub-stream X is last transmitted. By definition, $\phi[k] = \phi[k+1] = \cdots = \phi[n]$. Thus, case III condition of $\phi[n] > \theta[n] + 1$ means

$$\phi[k] = \phi[n] > \theta[n] + 1 \geq \theta[k] + 1. \tag{5}$$

That means at instant kT, EDF had no more available frames in sub-stream Y to send before sending sub-stream X data of later deadlines. Moreover, since instant kT, EDF has used all available bandwidth to send data from sub-stream Y. So another schedule Π' cannot possibly send more data from sub-stream Y by time nT, and would necessarily underflow sub-stream Y, thereby the whole video. This is a contradiction to the assumption of a feasible Π'.

Since a contradiction is reached for all three cases, we can conclude that if EDF schedule is not feasible, then no feasible schedule exists. This completes the proof of EDF's minimal feasibility.

E. Tradeoffs of Initial Buffer Delay and Response Lag

Given EDF is minimally feasible, we can use EDF to find the *minimum* response lag $\Delta_Y T$ such that no receiver buffer underflow occurs given initial buffer delay $\Delta_X T$—the best tradeoff between Δ_Y and Δ_X—using bisection search:

1) Initialize *feasible lag* $\Delta_Y^f := \Delta_X$ and *infeasible lag* $\Delta_Y^i := 0$. Let *current lag* $\Delta_Y(0) := \Delta_X$ and apply EDF to check for feasibility given $\Delta_Y(0)$ and Δ_X. Let $n := 0$.
2) If EDF is infeasible given $\Delta_Y(0)$ and Δ_X, conclude that there is no feasible schedule for $\Delta_Y \leq \Delta_X$. Stop.
3) If EDF is feasible given $\Delta_Y(n)$ and Δ_X, update feasible lag $\Delta_Y^f := \Delta_Y(n)$. Go to step 5.
4) If EDF is infeasible given $\Delta_Y(n)$ and Δ_X, update infeasible lag $\Delta_Y^i := \Delta_Y(n)$.
5) Update current lag $\Delta_Y(n+1) := \lceil (\Delta_Y^i + \Delta_Y^f)/(2) \rceil$. If $\Delta_Y(n+1) = \Delta_Y(n)$, stop.
6) Apply EDF using $\Delta_Y(n+1)$. $n := n + 1$. Go to step 3.

Using the above procedure, we can find the optimal tradeoff between initial buffer delay and response lag. In practice, Δ_X is selected *a priori* as the maximum initial buffering delay a user can tolerate. The minimum response lag Δ_Y given Δ_X can then be found using the above procedure.

V. Transport Scheduling for Multiple Separable Video Streams

In this section, we theoretically examine the improvement in multiplexing gain of separable video over non-separable video. When non-separable video is employed for community streaming with interactive visual overlay, each compressed video frame needs to be transmitted with low latency. In other words, the demand for bandwidth is *inelastic* for non-separable video. In contrast, separable video typically contains a small interactive sub-stream that has inelastic demand for bandwidth, and a non-interactive sub-stream that can be pre-fetched. The demand for bandwidth is flexible or *elastic* for the non-interactive stream. In this section, we develop scheduling methods for separable video to exploit its inherent flexibility and quantify the gain using an effective bandwidth formulation. Specifically, under identical channel bandwidth and bound on probability of resource overload, we seek to compare the number of streams that can be supported when separable and non-separable video is respectively employed. Such network dimensioning question is of practical interest for network provisioning and call admission control purposes.

A. Effective Bandwidth Background

Of particular relevance are existing effective bandwidth results on multiclass traffic [19], [20], [24]. Specifically, the goal

is to compute effective bandwidths α_j so that a network link can carry n_j streams of class j if the following is satisfied:

$$\sum_{j=1}^{J} n_j \alpha_j \leq C \tag{6}$$

where C is the total capacity of a link, and streams of the same class are assumed to be independent and with similar statistics. A remarkable set of results have been obtained for cases where the link buffer is zero (bufferless) [19], [21] or large (infinite) [20], [24].

If the total traffic $X(t)$ into a bufferless link of capacity C is stationary and ergodic, then the fraction of time $X(t)$ exceeds C equals $P\{X(T) > C\}$ at any given time instant T. As a result, it suffices to consider X as a random variable:

$$X = \sum_{j=1}^{J} \sum_{i=1}^{n_j} X_{ij}$$

where the random variable X_{ij} is the offered traffic from a constituent stream. Then, for any $s \geq 0$, the probability of resource overflow can be bounded by [21], [19]

$$\log P\{X \geq C\} \leq \sum_{j=1}^{J} n_j \Lambda_j(s) - sC \tag{7}$$

where Λ_j is the log moment generating function for class j, and is defined as

$$\Lambda_j(s) = \log \mathbf{E}[e^{sX_{ij}}]. \tag{8}$$

Given a prescribed maximum overload probability of $e^{-\delta}$ (i.e., $\log P\{X \geq C\} \leq -\delta$), it is shown that an effective bandwidth can be chosen for class j as [19]

$$\alpha_j = \frac{1}{s^*} \Lambda_j(s^*) \tag{9}$$

where s^* minimizes the expression in (7). This choice satisfies the prescribed error bound as long as (6) is satisfied and gives an asymptotically tight error bound. The significance of (9) is that it allows computation of an effective bandwidth which can be used in conjunction with (6) for determining how many flows of each class are admissible without violating a given overload probability.

B. Scheduling of Separable Video Over Bufferless Link

Most works related to effective bandwidth assume the offered traffic is *inelastic* or fixed, and of identical importance, and study aggregate behavior under FIFO queuing. Extensions to priority queuing with inelastic offered traffic are described in [25], where it is found to be necessary to associate each traffic class with multiple effective bandwidths, one for each priority level. In contrast, in the context of separable video, differential treatment of sub-streams is based on their elasticity. In other words, the total traffic of a separable stream, X_{ij}^S, where i and j are arbitrary indices, can be considered as the sum of an inelastic sub-stream X_{ij}^I and an elastic sub-stream X_{ij}^E:

$$X_{ij}^S = X_{ij}^I + X_{ij}^E \tag{10}$$

with different choices of X_{ij}^E resulting in different traffic profiles and effective bandwidths for X_{ij}^S. In the rest of this section, we discuss how to design X_{ij}^E in order to maximize the number of streams that can be supported under (6), and compare that to the non-separable video case. When multiplexing multiple separable video streams, the aggregate traffic X^S is given by

$$\begin{aligned} X^S &= \sum_{j=1}^{J} \sum_{i=1}^{n_j} X_{ij}^S \\ &= \sum_{j=1}^{J} \sum_{i=1}^{n_j} X_{ij}^I + \sum_{j=1}^{J} \sum_{i=1}^{n_j} X_{ij}^E \\ &= X^I + X^E \end{aligned} \tag{11}$$

where X^I and X^E denote the aggregate amount of inelastic and elastic traffic, respectively. X^I is given and cannot be modified, but our ability to transmit fixed regions well ahead of delivery deadlines affords us flexibility in the design space of X^E.

1) Intra-Stream Optimization of Elastic Traffic: Consider the case when separable video streams are multiplexed but transmitted independently of each other, e.g., by different servers that do not communicate. Then we need to first determine the optimal random variables X_{ij}^{E*} that minimizes the effective bandwidth of X_{ij}^S:

$$X_{ij}^{E*} = \underset{\{X_{ij}^E \geq 0 : \mathbf{E}[X_{ij}^E] = m_j^E\}}{\arg\min} \alpha_j \tag{12}$$

where m_j^E is average bit-rate for the elastic traffic from class j. If $\sup X_{ij}^I \leq m_j^E + m_j^I$, where $m_j^I = \mathbf{E}[X_{ij}^I]$ is the average bit-rate of inelastic traffic from class j, it can be shown that

$$X_{ij}^{E*} = m_j^E + m_j^I - X_{ij}^I \tag{13}$$

which is clearly nonnegative and has mean m_j^E. This choice guarantees that $X_{ij}^S = m_j^E + m_j^I$ is a constant, which intuitively minimizes effective bandwidth. More formally, since the exponential function is convex, Jensen's inequality establishes that

$$\begin{aligned} \alpha_j &= \frac{1}{s^*} \log \mathbf{E}\left[e^{s^* X_{ij}^S}\right] \geq \frac{1}{s^*} \log e^{s^* \mathbf{E} X_{ij}^S} \\ &= m_j^E + m_j^I = \alpha_{X_{ij}^{E*}} \end{aligned} \tag{14}$$

or that X_{ij}^{E*} in (13) achieves the lowest effective bandwidth $\alpha_{X_{ij}^{E*}}$ among all random variables with the same mean. In other words, the elastic traffic should be scheduled in a "water-filling" manner to make the overall traffic constant. The formulation above imposes no constraint that elastic traffic must be transmitted before its deadline. In practice, for streaming of long streams, this can be enforced by initially transmitting at a higher rate to build up the buffer, which would have negligible effect on overall behavior.

When $\sup X_{ij}^I > m_j^E + m_j^I$, X_{ij}^S cannot be rendered constant with any choice of X_{ij}^E, and the optimal strategy is given by the Kuhn–Tucker conditions $X_{ij}^E \geq 0$ and $\mathbf{E}[X_{ij}^E] = m_j^E$:

$$X_{ij}^{E*} = \left(M_j - X_{ij}^I\right)^+ \tag{15}$$

where M_j is given by

$$\mathbf{E}\left[M_j - X_{ij}^I\right]^+ = m_j^E \tag{16}$$

where X_{ij}^{E*} is again performing a "water-filling" strategy to provide as constant a data rate as possible for X_{ij}^S. It should be noted that (15) and (16) is the general solution that subsumes (13) as a special case.

With the optimal strategy of elastic traffic determined, the total traffic profile X^S can then be computed, and standard effective bandwidth analysis discussed earlier in the section can be applied to compute the maximum number of sessions that can be supported. Numerical comparison will be given later in Section VI-C.

2) Cross-Stream Optimization of Elastic Traffic: Generally, when multiple separable streams are originating from a single server, or if the servers coordinate, it is possible to further reduce effective bandwidth by jointly optimizing the elastic traffic across multiple streams. Specifically, instead of (15) and (16) that consider each stream in isolation, we consider the joint assignment of aggregate elastic traffic X^E. Using similar methods as before, the optimal assignment of elastic traffic, X^{E*} is given by

$$X^{E*} = (M - X^I)^+ \tag{17}$$

where M is given by

$$E[M - X^I]^+ = m^E \tag{18}$$

where m^E is the average rate for aggregate elastic traffic. Intuitively, intra-stream optimization is trying to rearrange elastic traffic to make individual stream as constant in bit-rate as possible. In contrast, cross-stream optimization treats the aggregate traffic as a single stream, and attempts to rearrange aggregate elastic traffic to make the overall traffic as constant as possible. Clearly cross-stream optimization performs better due to further flexibility to shift resources across streams. The performance improvement is illustrated in the example in Section VI-C.

Note that the "water-filling" scheduler is different from the EDF scheduler not only in strategy but in resulting traffic profile as well. This is not a contradiction, since the two schedulers are optimal in their respective goal and under their respective assumption. Specifically, the "water-filling" scheduler addresses the provisioning problem for *statistical* multiplexing of independent streams with the assumption that the overall traffic is *stationary*. In contrast, EDF is a server strategy to *deterministically* exploit channel bandwidth by a single stream and generally produces *non-stationary* traffic profile. In addition, EDF assumes that a single stream has complete control of the channel, while during multiplexing of multiple independent streams, such assumption no longer holds, and the work-conserving property of EDF cannot be maintained.

VI. Results

In this section, we first examine the effect of using separable video on compression efficiency. This complements the discussion in Section III on the computation savings associated with separable video. We next compare the interactive delay necessary to guarantee smooth video playback for separable and non-separable video for different bandwidth capacities and initial buffering delay in Section VI-B. Finally, we examine the statistical gains of the "water-filling" scheduling in Section VI-C. All test sequences are QCIF (176×144) at 10 fps, unless otherwise noted.

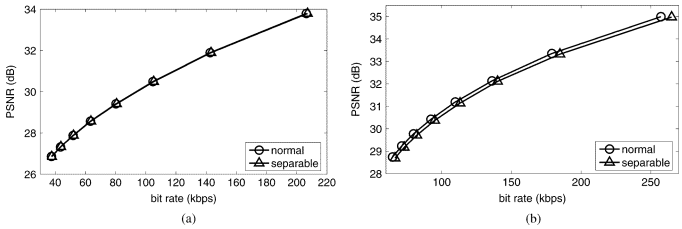


Fig. 7. Loss in coding efficiency in PSNR when using separable video. (a) *Counter-Strike 1*. (b) *Foreman*.

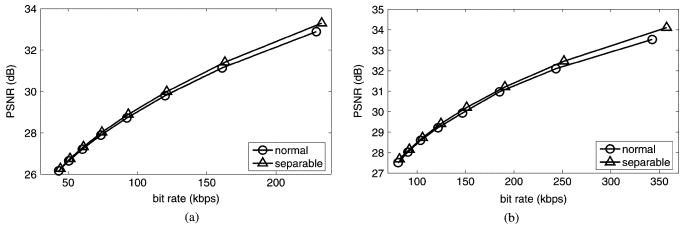


Fig. 8. Gain in coding efficiency in PSNR when separable video are subsequently re-encoded. (a) *Counter-Strike 1*. (b) *Foreman*.

A. Coding Efficiency of Separable Video

Generally, restricting motion vectors to create separable video as discussed in Section III-B incurs a loss in coding efficiency. Fig. 7(a) characterizes such loss for a video sequence from game *Counter-Strike*. Sequences *Mother and Daughter* and *Mobile Calendar* at 30 fps show similar characteristics (not shown) with the rate-distortion curves of normal and separable cases being virtually identical. For the *Foreman* sequence (30 fps), shown in Fig. 7(b), the PSNR loss is more pronounced at about 0.2 dB.

One advantage of using separable video is the ability to selectively process only the modifiable region of the video. For video coded using normal methods, it is necessary to re-encode the entire frame if any part of the frame is modified. This requires additional computation and introduces *regeneration loss* compared to the use of separable video, where the fixed region is not re-encoded. Fig. 8 shows the PSNR gain when separable video is employed rather than normal video encoding. We notice that there is generally a 0.1 to 0.2 dB improvement at usable qualities around 32 dB. Notice that for the *Foreman* sequence, the PSNR gain is achieved over an initial loss in PSNR of Fig. 7(b). We thus conclude that using separable video in our ECHO system incurs no significant loss in compression efficiency.

The bit-rate profiles for two different scenes of *Counter-Strike* produced by HC are shown in Fig. 9, with separate lines for non-separable video, the modifiable region of separable video, and total rate of separable video. First, we observe that the bit-rate profiles of the non-separable video and the separable video are very similar. This confirms our earlier observation that the coding cost of using separable video instead of non-separable video is small. Second, we see that all three

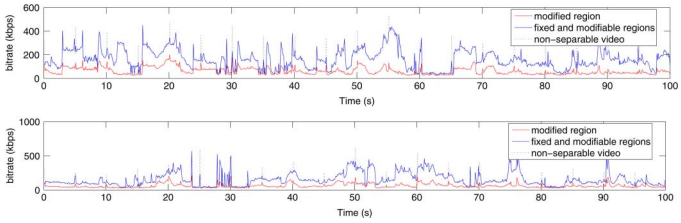


Fig. 9. Two scenes of *Counter-Strike* show large bit-rate variation over time.

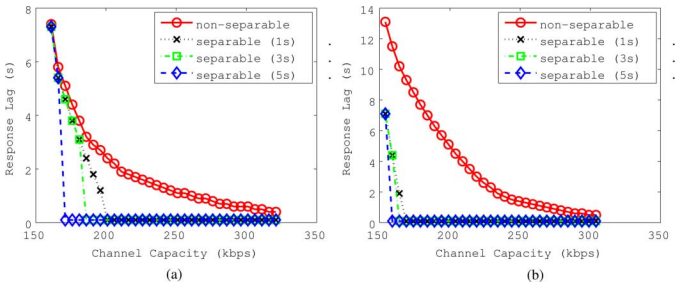


Fig. 10. Response lag versus channel capacity for *Counter-Strike* sequences. (a) *Counter-Strike 1*. (b) *Counter-Strike 2*.

bit-rate profiles of the same sequence have large variations as a function of time. The largely VBR nature of the encoded video has important consequences in terms of multiplexing, which we will discuss shortly.

We find that the shape of the profile in Fig. 9 does not change significantly when H.264 is employed in place of H.263. When natural video is used instead of synthetic game content, we generally observe similar fluctuation in bit-rate profile, though at a slower pace. This is probably due to the faster motion and more frequent scene change in action games such as *Counter-Strike* than natural video. The rate control scheme employed in all comparison is constant quantization parameter.

B. Scheduling Single Video Stream Over Known Channel

In this section, we show experimentally the advantages of separable video over non-separable video when scheduling a single video stream over an idealized delivery channel with constant bit-rate (CBR). We assume zero transmission delay and zero transmission loss in the channel. For non-separable stream, for a given channel capacity, we calculate the receiver buffer delay required to avoid receiver buffer underflow during playback. This is equivalent to the response lag for the interactive visual overlay for the non-separable stream. For separable stream, for a given channel capacity and initial buffering delay, we calculate the response lag required to avoid interactive sub-stream playback underflow. We accomplish that using the procedure in Section IV-E that utilizes EDF to find the best tradeoff between initial buffer delay and response lag. The results are shown in Fig. 10. First, we observe that, as expected, the response lag is inversely proportional to the channel capacity for all performance curves. Second, we see that separable streaming outperforms non-separable stream for the full range of channel capacity, and for all values of initial buffer delay. In particular, separable streaming can reduce the response lag over non-separable streaming by up to 5 and 11 s for *Counter-Strike 1* and *Counter-Strike 2*, respectively. This validates experimentally our claim that the separable-stream approach has significant streaming benefits over non-separable streaming. Finally, we see that as the amount of initial buffer delay increases, the performance of separable streaming improves. Intuitively, larger buffering yields better VBR rate averaging over a CBR channel, as discussed in Section IV.

C. Statistical Multiplexing of Traffic With Poisson Distribution

We next consider the multiplexing of traffic streams with instantaneous data rate X_i that are independent and identically distributed according to the Poisson distribution of parameter λ, or $P\{X_i = k\} = (\lambda^k e^{-\lambda})/(k!)$. This distribution is chosen for its nonnegative support that is suitable for representing data rate, the existence of closed form expressions for various quantities of interest, and having a shape that is a reasonable approximation to compressed video frame sizes. Then, the effective bandwidth is given by (9):

$$\alpha = \frac{C - n\lambda}{n \log \frac{C}{n\lambda}}.$$

We assume the inelastic and elastic components of the corresponding separable stream are, respectively, given by $X_{ij}^I = \beta X_{ij}, 0 \leq \beta \leq 1$, and $\mathbf{E}[X_{ij}^E] = (1-\beta)\mathbf{E}[X_{ij}] = (1-\beta)\lambda$. For average per-stream data rate of $\lambda = 10$, and error rate bound of $e^{-\delta} = 0.001$, the effective bandwidth and bandwidth utilization for separable and non-separable video are given in Fig. 11(a) and (b), respectively. Curves labeled "Separable" are obtained using only intra-stream optimization, while curves labeled "Separable-Cross" are obtained using cross-stream optimization. In Fig. 11(a), we observe generally that effective bandwidth decreases as the fraction of inelastic traffic β decreases due to more flexibility in traffic shaping. When $C = 100$, where an order of $C/\lambda = 10$ streams are multiplexed, we see that the effective bandwidth of non-separable video is 30% above the average rate of 10. For separable video with $\beta = 0.9$ (10% elastic traffic), the effective bandwidth needed to guarantee the same error bound drops to 11.7 at the same multiplexing level. Further increase in the ratio of elastic traffic further decreases effective bandwidth, and with $\beta = 0.5$ or half elastic traffic, the effective bandwidth becomes virtually 10, which is the ideal value achieved by constant rate source. We also observe that effective bandwidth generally decreases as C increases due to higher level of statistical multiplexing. In particular, we see that the effective bandwidth of non-separable video drops from 13 when $C = 100$ (≈ 10 stream) to 10.3 when $C = 4000$ (≈ 400 streams). In contrast, lower effective bandwidth can be achieved at $C = 200$ with 25% elastic traffic and at $C = 100$ with 50% elastic traffic. This indicates that the use of elastic traffic is an attractive alternative to increasing the level of multiplexing to realizing multiplexing gain. The maximum number of streams supported n^* at different values of C are shown in Fig. 11(b) with a normalization factor of C/λ, the ideal maximum. The quantity $n^*/(C/\lambda) = n^*\lambda/C$ is also the bandwidth utilization. We see that non-separable video achieves utilization of 60% (six streams) to 94% (376 streams) for C equals 100 and 4000, respectively. In contrast, separable

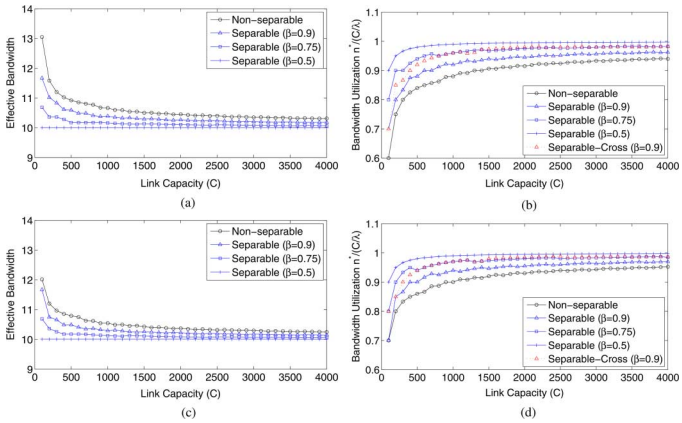


Fig. 11. Comparison of multiplexing of separable and non-separable videos with Poisson distributed data rate of $\lambda = 10$, and different resource overflow probability $e^{-\delta}$. Only curve labeled "Separable-Cross" employs cross stream optimization. Only intra-stream optimization is employed for other curves labeled "Separable". (a) Eff. bw ($e^{-\delta} = 0.001$). (b) Utilization ($e^{-\delta} = 0.001$). (c) Eff. bw ($e^{-\delta} = 0.01$). (d) Utilization ($e^{-\delta} = 0.01$).

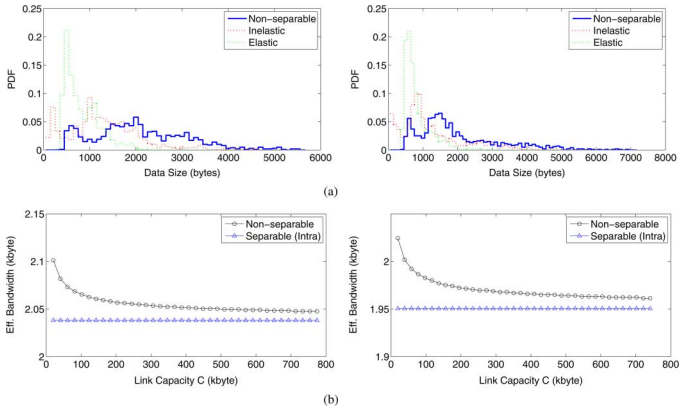


Fig. 12. Comparison of streaming using separable and non-separable video using compressed video trace. (a) Byte size distribution for two *Counter Strike* sequences. (b) Effective bandwidth for *Counter Strike* sequences.

video with 50% elastic traffic achieves significantly higher utilization of 90% (nine streams) to virtually 100% (399 streams) for C equals 100 and 4000, respectively. Corresponding results for cross-stream optimization are shown in the "inelastic" curve in Fig. 11(b). As expected, cross-stream optimization further improves utilization compared to intra-streaming optimization with the same fraction of elastic traffic.

Corresponding results for $e^{-\delta} = 0.01$ are given in Fig. 11(c) and (d). Generally, higher tolerance of error corresponds to lower effective bandwidth, but similar trends are observed as in Fig. 11(a) and (b).

D. Statistical Multiplexing of Trace Traffic

We next consider the statistical multiplexing of empirical traffic given in Fig. 9. The distribution of frame byte sizes are given in Fig. 12(a) for the two *Counter Strike* sequences. Unlike the Poisson distribution considered earlier, the empirical distributions in Fig. 12(a) are each derived from only 100 s of video and do not exhibit a long tail. With a peak-to-average ratio of about 3, the streams multiplex very well, achieving an effective bandwidth within 4% of the average bit-rate when about ten streams are multiplexed [leftmost point in Fig. 12(b)]. Nevertheless, the multiplexing advantage of using separable video is still clearly demonstrated, as the limiting effective bandwidth is readily achieved when order of ten streams are multiplexed. In contrast, the same limiting effective bandwidth is not achieved with close to 400 streams multiplexed [rightmost point in Fig. 12(b)] when non-separable video is employed. The average elastic traffic for both sequences is about 40%.

VII. Conclusion

In this paper, we have presented a networked video system that supports shared viewing of video with a sense of community that is reinforced by user controlled interactive visual overlays. We believe this form of community streaming is applicable to many applications and show specific ways to optimize such system. We argue that it is advantageous to employ separable video streams due to lower computation and transport requirements, as well as comparable coding efficiency. We also show that the EDF scheduling algorithm is optimal when streaming a single stream over a known channel, and that a "water-filling" scheduling is optimal for multiplexing of many streams.

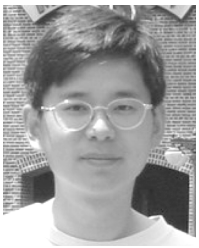
Acknowledgment

The authors would like to thank T. Sakamoto of HP Labs Japan for software support related to the *Counter-Strike* client.

References

[1] V. Kahmann, J. Brandt, and L. Wolf, "Collaborative streaming and dynamic scenarios," *Commun. ACM*, vol. 49, no. 11, pp. 58–63, Nov. 2006.
[2] *Counter-Strike*. [Online]. Available: http://www.counter-strike.net.
[3] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer, "Ubiquitous device personalization and use: The next generation of IP multimedia communications," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 3, no. 2, May 2007, Article no. 12.
[4] *Second Life: Official Site of the 3D Online Virtual World*. [Online]. Available: http://secondlife.com.
[5] *Meet-Me*. [Online]. Available: http://www.meet-me.jp.
[6] *Sauerbraten*. [Online]. Available: http://sauerbraten.org.
[7] YouTube. [Online]. Available: http://www.youtube.com.
[8] *Half-Life TV*. [Online]. Available: http://www.hltv.org.
[9] *StreamMyGame*. [Online]. Available: http://streammygame.com.
[10] G. Cheung, T. Sakamoto, and W. t. Tan, "Graphics-to-video encoding for 3G mobile game viewer multicast using depth values," in *Proc. IEEE Int. Conf. Image Processing*, Singapore, Oct. 2004.
[11] G. Cheung, A. Ortega, and T. Sakamoto, "Coding structure optimization for interactive multiview streaming in virtual world observation," in *Proc. IEEE Int. Workshop Multimedia Signal Processing*, Cairns, Australia, Oct. 2008.
[12] G. Cheung, W.-T. Tan, B. Shen, and A. Ortega, "ECHO: A community video streaming system with interactive visual overlays," in *Proc. IS&T/SPIE 15th Annu. Multimedia Computing and Networking (MMCN'08)*, San Jose, CA, Jan. 2008.
[13] T. V. Lakhsman, A. Ortega, and A. R. Reibman, "VBR video: Tradeoffs and potentials," *Proc. IEEE*, vol. 86, no. 5, pp. 952–973, May 1998.
[14] D. Tse, R. Gallager, and J. Tsitsiklis, "Statistical multiplexing of multiple time-scale Markov streams," *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1028–1038, Aug. 1995.
[15] J. D. Salehi, S.-L. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 397–410, Aug. 1998.
[16] C.-Y. Hsu, A. Ortega, and A. Reibman, "Joint selection of source and channel rate for VBR video transmission under ATM policing constraints," *IEEE J. Select. Areas Commun.*, vol. 15, no. 6, pp. 1016–1028, Aug. 1997.

[17] Y.-C. Chang and D. G. Messerschmitt, "Segmentation and compression of video for delay-flow multimedia networks," in *Proc. ICASSP 1998*, Seattle, WA, Apr. 1998.
[18] Y.-C. Chang and D. G. Messerschmitt, "Improving network video quality with delay cognizant video coding," in *Proc. ICIP 1998*, Chicago, IL, Sep. 1998, vol. 3, pp. 27–31.
[19] F. Kelly, *Stochastic Networks: Theory and Applications*. Oxford, U.K.: Oxford Univ. Press, 1996, ch. Notes on Effective Bandwidths, pp. 141–168.
[20] G. Veciana and J. Walrand, "Effective bandwidths: Call admission, traffic policing and filtering for ATM networks," *Queueing Syst.*, vol. 20, no. 1–2, pp. 37–59, Mar. 1995.
[21] J. Hui, "Resource allocation for broadband networks," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1598–1608, Dec. 1988.
[22] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[23] G. Cheung, A. Ortega, and T. Sakamoto, "Fast H.264 mode selection using depth information for distributed game viewing," in *Proc. IS&T/SPIE Visual Communications and Image Processing (VCIP'08)*, San Jose, CA, Jan. 2008.
[24] G. Kesidis, J. Walrand, and C. Chang, "Effective bandwidths for multiclass Markov fluids and other ATM sources," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 424–428, Aug. 1993.
[25] A. Berger and W. Whitt, "Effective bandwidths with priorities," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 447–460, Aug. 1998.



Wai-Tian Tan (M'00) received the B.S. degree from Brown University, Providence, RI, in 1992, the M.S.E.E. degree from Stanford University, Stanford, CA, in 1993, and the Ph.D. degree from University of California, Berkeley, in 2000.

He was a Software Developer for Oracle Corporation from 1993 to 1995 and joined Hewlett Packard Laboratories, Palo Alto, CA, in 2000, where he is now a Senior Researcher in the Media Communications and Networking Laboratory. His research interests are video compression structures for error resilience and real-time adaptive streaming systems.

Gene Cheung (SM'07) received the B.S. degree in electrical engineering from Cornell University, Ithaca, NY in 1995 and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 1998 and 2000, respectively.

In August 2000, he joined Hewlett-Packard Laboratories Japan, Tokyo, where he is currently a Senior Researcher. His research interests include media transport over wireless networks, joint source/network coding for video streaming, and community-based media interaction.

Dr. Cheung is currently an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA and a voting member of the IEEE Multimedia Communications Technical Committee.

Antonio Ortega (F'07) received the Telecommunications Engineering degree from the Universidad Politecnica de Madrid, Madrid, Spain, in 1989 and the Ph.D. in electrical engineering from Columbia University, New York, in 1994, where he was supported by a Fulbright scholarship.

In 1994, he joined the Electrical Engineering-Systems Department at the University of Southern California (USC), Los Angeles, where he is currently a Professor. He has served as Associate Chair of EE-Systems and as Director of the Signal and Image Processing Institute at USC. His research interests are in the areas of multimedia compression, communications, and signal analysis. His recent work is focusing on distributed compression, multiview coding, error tolerant compression, wavelet-based signal analysis, and information representation in wireless sensor networks. His work at USC has been or is being funded by agencies such as NSF, NASA, DOE, as well as a number of companies. Over 20 Ph.D. students have completed their thesis work under his supervision at USC and his work has led to over 200 publications in international conferences and journals.

Dr. Ortega is a member of ACM. He has been Chair of the Image and Multidimensional Signal Processing (IMDSP) technical committee and a member of the Board of Governors of the IEEE Signal Processing Society (2002). He was technical program co-chair of ICIP 2008 and was previously program co-chair of MMSP 1998 and ICME 2002. He is currently an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and in addition to a previous stint as an Associate Editor for he IEEE TRANSACTIONS ON IMAGE PROCESSING he has also served as an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS and for the *EURASIP Journal on Advances in Signal Processing*. He received the NSF CAREER award, the 1997 IEEE Communications Society Leonard G. Abraham Prize Paper Award, the IEEE Signal Processing Society 1999 Magazine Award, and the 2006 *EURASIP Journal of Advances in Signal Processing* Best Paper Award.

Bo Shen (SM'04) received the B.S. degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, China, and the Ph.D. degree in computer science from Wayne State University, Detroit, MI.

He is the Chief Video System Architect for a Silicon Valley startup company vuclip.com, a pioneer in providing video services to mobile phones. He was with Hewlett-Packard Laboratories as a Senior Research Scientist from 1997 to 2007. His research interests include multimedia signal processing, multimedia networking, and content distribution systems. He has published more than 60 papers in prestigious technical journals and conferences.

Dr. Shen has been on the Program Committee for a number of technical conferences including ACM SIGMM. He has served as an Associate Editor of the IEEE TRANSACTION ON MULTIMEDIA and also served as its Lead Guest Editor on a special section on mobile multimedia applications. He holds seven U.S. patents, with many pending.