Integrated Optimization of Video Server Resource and Streaming Quality Over Best-Effort Network

Hang Yu, Ee-Chien Chang, Wei Tsang Ooi, Member, IEEE, Mun Choon Chan, Member, IEEE, and Wei Cheng, Student Member, IEEE

Abstract—A video streaming server needs to adapt its source/channel encoding parameters (or configurations) to changes in network conditions and to differences in users’ connection profiles. The adaptation can be achieved by adjusting parameters such as frame rate, error protection ratio, and resolution. Ideally, the server should adapt the serving configurations with respect to the current network and user conditions to improve received video quality. However, adaptations that optimize playable frame rate require intensive computation, and storing all possible configurations requires a tremendous amount of storage. This brings forth the issues of how to obtain good video quality and reduce server resources usage at the same time. We address this issue in this paper. Our approach is based on the observation that transcoding between certain configurations can be performed very efficiently. We propose a framework to compute a set of configurations to store on the server by considering two opposing goals: (a) maximizing expected received quality of the video, and (b) minimizing server resource usage by lowering transcoding cost and expected number of switches between configurations. The second objective also reduces the number of configurations, and therefore reduces the total storage required. Our framework models the relationship among different configurations in a partial order, formulates the search of a good set of configurations as an energy minimization problem, and we use techniques in image segmentation to solve the problem. Experimental results show that our framework relieves the server load and increases the number of clients served, while only slightly reducing the expected frame rate.

Index Terms—Forward error correction (FEC) protection, graph cut, image segmentation, media server, MPEG streaming, rate adaptations.

I. INTRODUCTION

Streaming media applications need to adapt to changes in network conditions to avoid network congestion [1] and to share bandwidth fairly with other connections [15]. In addition, users of the streaming applications can connect to the servers with very different access bandwidth, from a user using a dial-up modem with 32 kbps, to a user in the local area network with 10 Mbps or more. By adapting the sending rate based on

both the bandwidth and loss characteristics, the server can meet a diverse set of service requirements.

In a best effort network, packets are transported without any notion of resource reservation or priority. As a result, network adaptation is often performed through rate adjustment in the following ways. First, the server can dynamically adapt its sending rate by transcoding the source video in real time. Some common transcoding operations include dropping frames [4], changing frame patterns [9], and tweaking quantization factors [1]. Such approaches may require substantial transcoding cost and sacrifice compression ratio or video quality to meet real-time requirements. Alternatively, the server can store multiple versions of the same video content, each with different bit rates. During transmission, the server switches among different versions to achieve the desired sending rate [8]. This approach avoids the cost of transcoding during streaming, but frequent switches among stored versions results in low reference locality and can incur additional I/O cost. Furthermore, storing multiple versions may require a lot of storage as the server could serve various users concurrently with different video content. To trade off between the cost of transcoding and switching, the server can use a hybrid solution: store only several versions of the video content, and transcode in real-time (if needed) to meet the desired sending rate.

Generally, these adaptation mechanisms can be summarized into the following three categories according to the work by Rejaie et al. [14].

1) Adaptive encoding: By adaptive encoding, the stored pre-encoded stream is requantized on the fly based on the feedback of network condition. There are two disadvantages for this method. First, the operation of re-encoding is CPU-intensive, thus making it impossible to handle a large number of clients. Second, the range of the output rate provided by the online encoder can not be large.

2) Switching among multiple pre-encoded streams: In this approach, multiple pre-encoded streams with different quality are stored in the server. When the network conditions change, the server switches to the appropriate stream accordingly.

3) Hierarchical encoding: With hierarchical encoding, multiple layered encoded versions of each stream are stored in the server. The server will add or drop layers of the stream for delivery based on network conditions. There are some differences between switching streams and hierarchical encoding. The former is source-based that the sender or the server is responsible for the quality adaptation to the available network condition while the latter is receiver-based that the receivers or clients regulate channel to
maximize the quality. The hierarchical encoding is more suitable for multicast to heterogeneous clients while the switching streams can be used in both unicast and multicast.

Most video compression schemes provide flexibility in the choice of encoding parameters, which can be used to adapt to the network condition. In particular, the frame pattern, (that is, the temporal pattern among the MPEG I-, P- and B-frames) can be chosen to suit the packet loss probability. Fig. 1 illustrates a common MPEG video frame pattern and the dependencies among different frames. Due to bit error and packets loss, not all frames can be recovered by the receiver. For example, the lost of P-frame will cause subsequent P- and B-frames to be unrecoverable, affecting the quality of the video. To increase the expected number of frames successfully reconstructed, the server can adapt to the network condition by changing the amount of redundant information sent to protect the video against different packet loss rates [3], [12]. For example, the application can increase the number of forward error correction (FEC) packets when the loss rate is high, to protect against loosing important packets (e.g., key frames). Such loss adaptation requires the server to reduce the number of bits allocated to less important data by either reducing quality, or dropping frames all together.

Given the current network condition, it is desirable to use the optimal joint source/channel encoding that maximizes the overall expected video quality received by the user. When network condition changes, the optimal encoding parameters may change, and the server may want to switch to the new optimal encoding parameters. However, switching between encoding parameters can be expensive. For instance, to switch an MPEG encoding from frame pattern IBPBP to IBBBP, the server needs to either transcode P-frames into B-frames or switch to another precomputed version of the video encoded using frame pattern IBBBP. One of our main observation is that transcoding among certain configurations can be carried out with negligible computing cost. This relationship gives a partial (not necessary linear) order for the possible set of configurations based on the transcoding relationship and suggests that a server could, instead of working on all possible configurations, consider only a small set of configurations and use these small set of configurations to generate all other configurations. There are many such choices and hence we need a formulation of a “good” set of configurations and an algorithm to search for it.

This paper offers an approach for trading off the expected video quality received by the user, and resource requirement at the server. Our goals are to: 1) achieve good expected recovered frame rate (when it is clear from the context, we call it expected frame rate hereafter) at the receiver; 2) reduce the cost for switches among versions stored on the server; and 3) reduce the transcoding cost during the transmission. To handle these opposing goals, we exploit the relationship among the configurations, and formulate the above problem as an energy minimization problem. The searching space to achieve the goals includes the combination of frame pattern and amount of FEC. We use a modification of the model by Wu [18] to determine the expected frame rate of a particular configuration at a particular network condition. To reduce the time for finding an optimal set of configurations, we employ the techniques from Boykov [2], which use graph minimum cut in each local search. Our formulation captures the tradeoff of the expected frame rate at the receiver and the resource usage at the server. As far as we know, there are no existing work that consider all these goals at the same time.

A. Related Work

Streaming of MPEG video over best-effort network has been studied extensively in the literature. These work mostly focus on techniques such as retransmission [5], rate shaping [6], [19] and prioritized transmission [11], [16]. These existing work focus on improving the quality of the MPEG stream received by the receiver. In contrast, our work aims at obtaining good quality and reducing server resources usage at the same time.

Our equations for expected frame rate are based on the work by Wu [18], Mayer-Patel [9], and Wolfinger [17]. These works have developed analytical frameworks for deriving frame loss probability given a frame pattern and the amount of FEC protection. Our formula for expected frame rates is based on the one by Wu [18]. We modified the equations slightly to take frame dropping into consideration.

There are some similarities between our optimization problem and the image segmentation model proposed by Boykov et al. [2]. Under this model, segmentation should be done in such a way to minimize: 1) the total differences of the assigned pixel values and the original values and 2) the number of pairs of neighboring pixels which lie in two different segments. These two goals are similar to our problem in the sense that, we want to assign each encoding parameters to network conditions in such a way that: 1) maximizes video quality and 2) minimizes number of switches between “neighboring” network conditions.

B. Outline

The remainder of this paper is organized as follows. We define our model in Section II. Section III provides the formulations to find the expected frame rate for a given configuration and network condition. Section IV presents our solution to find the optimal configurations. We show experimental results in Section V and conclude the paper in Section VI.

II. ASSUMPTIONS AND NOTATIONS

This section presents our assumptions and defines notations and terminologies used in our model. We first give the model for network conditions. Then, we formally define the notion of configurations, the concept of configuration reducibility and seed-set.

A. Network Conditions

The network condition is simplified to two parameters about which we care the most—the bandwidth and the probability of packet loss. The bandwidth is the maximum number of bytes we

---

**Fig. 1.** Common MPEG frame pattern. The arrows indicate the frame dependencies. For example, the first B-frame can be recovered only if the first I-frame and first P-frame are successfully recovered.
can send into the network per unit time, without causing congestion or being TCP-unfriendly. The bandwidth can be obtained using the TCP equation \[10\]. The probability of packet loss can be obtained through receiver’s feedback using RTCP. We denote the current network conditions using tuple \((b, p)\), where \(b\) denotes the TCP-friendly bandwidth we can send, and \(p\) denotes the current probability of packet loss.

B. Sending Configurations

Although our approach is generic and can be applied to different video coding, in this paper we consider MPEG video since it is a widely accepted standard. Furthermore, we are interested in adaptations by modifying frame patterns, dropping B-frames, and changing FEC ratio. While dropping frames may seem drastic and can have adverse effect on the viewing quality, we like to point out that the bits saved by not sending a B-frame can be allocated to FEC ratio to protect other more important frames.

We represent the frame pattern sent by the sender, per unit time, to be a string consisting of symbols \(\{I, P, B, x\}\). The order of the symbols in the string corresponds to the display order and each symbol represents the frame type. The symbol \(x\) denotes that the frame has been dropped by the sender. We also represent the frame pattern by three numbers \((N_I, N_P, N_B)\). Each \(N_I, N_P\) or \(N_B\) corresponds to the number of the respective I-, P-, and B-frames in the group. This simplification is useful because, given \((N_I, N_P, N_B)\), there are few reasonable choices of frame pattern. We consider a fixed GOP length, thus the length of the frame pattern is fixed. During transmission, FEC can be included to protect each frame. As different MPEG frame types have different importance, FEC ratio for I-, P-, and B-frames should be different as well.

The combination of the frame pattern and the FEC ratios defines a sending configuration or configuration for short. Each configuration is represented as the tuples \((F, r_I, r_P, r_B)\), where \(F\) is a frame pattern, and the last three tuples are the FEC ratios for I-, P-, and B-frames respectively. The FEC ratio is the ratio of the packet size for the FEC to the original packet size for the frame. An example of a configuration is \((I\times B\times P\times B\times B, 0, 0, 0)\).

C. Reducibility

We say that a configuration \(C_1\) can be reduced to another configuration \(C_2\) if any MPEG video encoded with \(C_1\) can be transcoded to \(C_2\) with negligible computational overhead. In this paper, we consider transcoding that simply discards packets as negligible computation. Hence, the relationship \(R\) will be partial order and can be represented using a Hasse diagram, which we call reducibility graph. \(R\) depends on the definition of “negligible” computation overhead. We write \(C_1 \geq_R C_2\) to denote \(C_1\) can be reduced to \(C_2\) with respect to a given \(R\) (or simply \(C_1 \geq C_2\) when it is clear from the context).

In this paper, we consider the following two ways of reduction as negligible computation:

- Reducing the FEC ratio. For example, \((I\times B\times P\times B\times L, 0.1, (0.1, 0.1)) \geq (I\times B\times P\times B\times L, 0.1, (0.1, 0.05, 0)).\)
- Dropping B-frames. For example, \((I\times B\times P\times B\times L, 0, 0, 0) \geq (I\times B\times P\times B\times L, 0, 0, 0).\)

Reducing the FEC ratio is convenient since we can simply throw away the extra FEC bits. Dropping B-frames only involves discarding the frame data itself, as no other frame depends on B-frames. Including the flexibility of dropping P- and I-frames will not introduce much more reasonable configurations and hence is omitted for simplicity. Fig. 2(a) shows a simple example of reducibility graph using the second way of reduction.

D. Seed Configurations

We now define irreducible configurations. Let \(C\) be the set of all possible configurations. A set of configurations \(\mathcal{L}\) is called irreducible set if \(\mathcal{L}\) does not contain two different configurations \(C_1\) and \(C_2\) such that \(C_1\) can be reduced to \(C_2\) (i.e., \(C_1 \geq_R C_2\)) and vice versa. Given any set of configurations \(\mathcal{M}\), it is possible to find an irreducible set of configurations \(\mathcal{L}\) such that every configuration in \(\mathcal{M}\) can be reduced from a configuration in \(\mathcal{L}\). We call a configuration in \(\mathcal{L}\) a seed for \(\mathcal{M}\) and \(\mathcal{L}\) the seed-set. Note that it is not necessary that \(\mathcal{L} \subseteq \mathcal{M}\), for example, in Fig. 2(b) \(C_1\) and \(C_2\) are not in the original \(\mathcal{M}\).

The intuitive meaning of a seed-set is as follows. Let us consider Fig. 2(b) as an example; suppose the server needs to support streaming of video from a large set of configurations \(\mathcal{M}\). The server does not need to store video streams encoded in each of the configuration in \(\mathcal{M}\). It suffices to find a seed-set \(\mathcal{L}\) of \(\mathcal{M}\) and store a set of video encoded using \(\mathcal{L}\). During transmission, if video encoded with a particular configuration \(C_4\) in \(\mathcal{M}\) is needed, the server can transcoding the video from a configuration \(C_1\) in \(\mathcal{L}\) with negligible computation overhead. A directed edge from \(C_1\) to \(C_4\) not only indicates that the expected video quality of \(C_1\) is better than \(C_1\), it also indicates that transcoding of \(C_1\) to \(C_4\) can be easily done.

E. Assigning Configuration to Network Condition

The sender needs to decide which configuration to use given a network condition. This decision is represented by an assignment. An assignment \(f\) is a mapping from network conditions to configurations. We write \(f(b, p) = \mathcal{C}\) to indicate mapping

![Fig. 2. Reducibility graph. (a) Example of reduction by dropping B-frame. (b) Reducibility. \(C_1\) and \(C_2\) are the seeds of the set \(\mathcal{M} = \{C_4, C_3, C_6, C_5, C_7\}\).](image-url)
from network condition with bandwidth \(b\) and packet loss probability \(p\) to a configuration \(C\). An assignment \(\tilde{f}\) is conforming if the bandwidth required by \(f(b,p)\) is no more than \(b\). We denote the range of an assignment \(f\) as \(L_f\). An assignment \(\tilde{f}\) is called a seed assignment if \(L_{\tilde{f}}\) is irreducible. Note that the bandwidth required by \(\tilde{f}(b,p)\) might be more than \(b\). Hence, it is not necessary that a seed assignment is conforming.

We can now express more formally how a server stores multiple versions of the same video content and adapts them with negligible computational overhead during transmission. Suppose the server uses conforming assignment \(f\) to map network conditions to a configuration, i.e., when the network condition is \((b,p)\), the video with configuration \(f(b,p)\) is sent. However, instead of precomputing and storing videos encoded with \(L_f\), it computes \(\tilde{f}\) such that \(\tilde{f}(b,p) \supseteq f(b,p)\) for all \(b\) and \(p\) and stores video encoded with \(L_{\tilde{f}}\). When the current network condition is \((b,p)\), the server retrieves video \(\tilde{f}(b,p)\) and transcodes it to \(f(b,p)\) with minimal computation overhead. Furthermore, if the network condition changes from \((b_0,p_0)\) to \((b_1,p_1)\), and \(\tilde{f}(b_0,p_0) = \tilde{f}(b_1,p_1)\), the server can switch configuration from \(f(b_0,p_0)\) to \(f(b_1,p_1)\) by changing the transcoding operation without switching to a different version on disk.

### III. Expected Recovered Frame Rate

Here, we briefly present the analytical model of Wu [18] on computing the expected recovered frame rate, that is, given a configuration and network condition \((b,p)\), we want to know the expected frame rate successfully recovered at the receiver’s end. Although we choose Wu [18] to obtain the expected frame rate, it can be replaced by another model, for example, that of Mayer-Patel [9] or a more accurate estimation through simulation. In other words, this is orthogonal to our overall framework.

Let \(K\) be the average number of packets sent in a video frame and \(r\) the FEC ratio of the frame. Thus, the average number of FEC packets for the frame is \(Kr\). Let \(N\) be the total number of packets sent for a frame. Then, we have \(N = K + Kr\). Furthermore, \(N\) must satisfy the bandwidth requirement \(b\).

Under FEC coding scheme, the original \(K\) packets can be recovered if at least \(K\) packets (out of \(N\)) are received. Let \(p\) be the probability of the packet loss during transmission. By modeling the sending process as a series of Bernoulli trials, the probability that \(K\) packets are successfully recovered is given by

\[
g(N, K, p) = \sum_{i=K}^{N} \binom{N}{i} (1-p)^{i} p^{N-i}.
\]

Thus, we can compute the probability to successfully recover a specific frame type I-, P- or B-frame as \(g_I\), \(g_P\), and \(g_B\), respectively.

Note that we assume that packet losses are independent events. In practice, packet loss events in the Internet are correlated. However, Wu et al. [18] have shown that this assumption affects the result only marginally.

Given a configuration \(C\), let \(N_i\) be the number of frames of type \(i\) and let \(r_i\) be the FEC ratio for frame of type \(i\). The expected number of I-frames recovered \(R_I\) is given by

\[
R_I = N_I g_I.
\]

Successful recovery of P-frames depends on successful recovery of all I-frames and P-frames before them. The expected number of P-frames recovered, \(R_P\), is thus given by

\[
R_P = N_I g_I \frac{g_P N_P^{1+1}}{1-g_P}.
\]

The expected number of B-frames recovered can be computed by

\[
R_B = N_{B-P} N_I g_I g_B \left( \frac{g_P N_P^{1+1}}{1-g_P} + g_B g_P N_B \right)
\]

where \(N_{B-P}\) is the number of B-frames in an interval of I-P or P-P frames.

Hence, given \(C\), \(b\), and \(p\), the expected reconstructed frame rate \(Efr(C,b,p)\) is

\[
Efr(C,b,p) = R_I + R_P + R_B.
\]

### IV. Optimal Configurations

In Section III, we showed how to find the expected recovered frame rates \(Efr(C,b,p)\) at the receiver given a configuration \(C\) and a particular network condition \((b,p)\). Here, we consider a range of network conditions, and show how to determine the seeds, that is, a set of irreducible configurations to be stored at the sender. Recall that our goals are to find such a seed-set so that both the overall expected cost of transcoding, and the size of the irreducible set are small while achieving a good recovered frame rate at the receiver. To model these opposing goals, we use an energy function to represent the effectiveness of a seed-set.

We present our model in details below.

First, we assume that the bandwidth \(b\) and packet loss probability \(p\) are sampled into discrete values. Let the sampled values be \(b_1, b_2, \ldots, b_n\) and \(p_1, p_2, \ldots, p_m\). We can imagine that every conforming assignment \(f\) on these sampled values forms a rectangular \(n \times m\) pixels image, where the value of the pixel at coordinate \((i, j)\) is the value \(Efr(f(b_i, p_j), b_i, p_j)\).

While sampling the bandwidth and packet loss into discrete values may cause loss of accuracy, we argue that the bandwidth and packet loss probability are estimated and approximated to begin with.

**Main Formulation:** Now, we express the goals and requirements in the search of a good assignment. We express the effectiveness of a given seed assignment \(\tilde{f}\) as a weighted subtraction of two opposing energy functions

\[
E(\tilde{f}) = E_t(\tilde{f}) - \beta E_n(\tilde{f}).
\] (1)

The first energy function \(E_t\) is the energy term representing the expected frame rate over some underlying probability distribution of the network condition. As we will explain in the next section, this function captures both the expected frame rate and the switching cost. The energy function \(E_n\) measures the
expected number of switches among different seeds when network condition changes. The function \( E_f (\hat{f}) \) captures both the expected frame rate at a particular network condition \((b, p)\) and how the network condition changes over time. The variable \( \beta \) is a parameter to trade off the relative emphasis between \( E_f \) and \( E_s \). We give the details of the formulation of \( E_f \) and \( E_s \) in Sections IV-A and -B.

### A. Overall Expected Frame Rate \( E_f (\hat{f}) \)

Let \( \text{Pr}(b_i, p_j) \) be the probability that the network bandwidth is \((b_i, p_j)\), let \( \text{BW}(C) \) be the size of the video encoding with configuration \( C \) in a unit time, and let \( K(C, b, p) \) be the best expected frame rate over all possible configurations \( C' \) that can be reduced from \( C \) and the bandwidth required by \( C' \) is no more than \( b \); that is,

\[
K(C, b, p) = \max_{C' \subseteq C, \text{BW}(C') \leq b} \text{Efr}(C', b, p).
\]

The expected overall frame rate can be expressed as

\[
E_f (\hat{f}) = \sum_{i,j} \text{Pr}(b_i, p_j) K \left( \hat{f}(b_i, p_j), b_i, p_j \right).
\]

We assume that the network condition is uniformly distributed, i.e., \( \text{Pr}(b_i, p_i) = c \), where \( c \) is some constant.

### B. Smoothness \( E_s (\hat{f}) \)

To describe our energy function that measures the smoothness, we first define the notion of neighborhood. Two points \((i, j)\) and \((x, y)\) are neighbors if \( ((i, j), (x, y)) \in \mathcal{N} \). A possible neighborhood (also known as 4-neighborhood) is

\[
\mathcal{N} = \{((i, j), (x, y)) : |i-x| = 1 \text{ or } |j-y| = 1, \text{ but not both} \}.
\]

We denote the cost of switching between configurations using function \( \text{Dist}(\cdot, \cdot) \), where \( \text{Dist}(C_1, C_2) = 1 \) if \( C_1 \neq C_2 \), 0 otherwise. The energy function \( E_s \) can then be defined as

\[
E_s (\hat{f}) = \sum_{(i,j), (x,y) \in \mathcal{N}} \text{Pr}(b_i, p_j) \text{Dist} \left( \hat{f}(b_i, p_j), \hat{f}(b_x, p_y) \right).
\]

We assume that it is equally probable for the network condition \((b_i, p_j)\) to change to any one of the neighbors \((b_x, p_y)\). If a more accurate model on the network is available, we can include the transition probability in (4).

By incorporating the transition probability, \( E_s \) is in fact the expected number of switches between configurations per unit time. This interpretation is useful in deciding what value to choose for \( \beta \).

### C. Finding the Optimal

Recall that the given parameters in the energy function (1) are the set of configurations \( C \) to be considered, the expected frame rate \( \text{Efr}(\cdot, \cdot, \cdot) \) for each configuration and network condition, the network model \( \text{Pr}(\cdot, \cdot) \), reducibility graph \( R \), and the weight \( \beta \). There are two modes of optimality.

#### Mode I: Finding the Optimal

Find the seed assignment \( \hat{f} \) that maximizes (1).

#### Mode II: Given an integer \( k \), find the seed assignment \( \hat{f} \) that maximizes (1) subjected to the constraint \( |C_f| \leq k \).

Recall that a seed assignment \( \hat{f} \) implicitly determines a conforming assignment \( f \). To adapt to the network condition \((b, p)\), the sender applies transcoding on video with \( \hat{f}(b, p) \) to obtain one with configuration \( f(b, p) \) and sends it over the network.

Energy functions similar to (1) appear in a number of applications, in particular image segmentation (for example, Boykov [2]) and the Potts model [13]. Optimizing these energy functions is generally difficult, and it is NP-hard even for a simple case [2]. Fortunately, extensive studies have been carried out and many efficient approximation algorithms have been proposed. We employ the method in Boykov [2] to find local optimal efficiently based on \((\alpha \leftarrow \beta)\) swap. Under this method, each network condition (or pixel in image segmentation) is first assigned a configuration (or “label” in image segmentation). Next, iteratively, optimal local move is performed. Here, a local move is a swap between a pair of configurations \( C_{\alpha}, C_{\beta} \). For example, if there is a swap of \( C_{\alpha} \) and \( C_{\beta} \), network condition which is assigned as \( C_{\alpha} \) might be changed to \( C_{\beta} \), and vice versa. Note that, for any two configurations, there are exponentially many ways to swap them. Hence, it is nontrivial to find the optimal local move. The algorithm by Boykov [2] finds the optimal local move in polynomial time by reducing the problem to graph min-cut. Note that the iteration of optimal local moves only guarantees to find the local maximum. Nevertheless, experimental studies show that it performs well, especially if there is a good initial guess. In our implementation, the initial assignment of the network condition \((b, p)\) is the \( C' \) s.t. \( \text{Efr}(C', b, p) \) is maximized.

### D. Searching Space

To maximize \( E(\hat{f}) \) using Mode I, we employ the above-mentioned \((\alpha \leftarrow \beta)\) swap method. We choose the initial frame pattern from \( C' \). Fig. 3 shows the convergence of our implementation. Details of the parameters for network condition and MPEG frame patterns used will be discussed in Section V.

In Mode II, we want to find at most \( k \) seeds that maximizes \( E(\hat{f}) \). We use the following approximation. The idea is to use
Mode I, but adjust the value of $\beta$ iteratively so that the size of the seed-set $\mathcal{C}$ (w.r.t. Mode II) is $k$.

Owing to the large number of possible frame patterns, we apply restrictions to exclude “unreasonable” patterns. The restrictions are similar to that of Mayer-Patel [9]: Let $N_I$, $N_P$, $N_B$ be the number of I-, P-, and B-frames in a frame pattern, respectively. First, for each combination of $N_I$, $N_P$, and $N_B$, only a frame pattern is considered. This is based on the intuition that different frame types should be distributed as evenly as possible. Next, the minimum ratio of anchor frames (i.e., I- and P-frames) to all frames is bounded. Specifically, $1/3 \leq (N_I + N_P)/(N_I + N_P + N_B) \leq 1$.

With the restrictions mentioned, the total number of frame patterns considered in this experiment is reduced to 2977 for our test video. However, the number of configurations is actually much more than that. This is because a frame pattern can be associated with different FEC ratio. Note that it is always better to have more FEC protection as long as the bandwidth allows. Hence, for a frame pattern and bandwidth, we only consider the configuration that includes maximum possible number of FEC bits. With this observation, in the implementation, explicit representation of all configurations in $\mathcal{C}$ can be avoided.

V. EXPERIMENTAL RESULTS

Here, we study the effectiveness of the proposed approach. In Section V-A, the optimal seed assignment is computed using the algorithm described in Section IV, and the performance is measured in terms of expected reconstructed frame rate. In Section V-B, we study the impact of the proposed approach in a video streaming setting and investigate the impact of the approach on the server resource usage and video quality.

A. Optimal Seed Configuration

1) Assumptions and Settings: Frame size: The size of each frame type is shown in Table I. Packets of all three frame types have the size of 1500 bytes. We consider a frame pattern with 30 frames, and the playback interval for each frame is 1/30 second. Hence, the best possible expected frame rate is 30 fps.

Network condition: We assume that network conditions are sampled and quantized. Bandwidth $b$ is measured by the maximum number of packets transmitted per second. Bandwidth $b$ starts from 120 to 300 with a step size of 10. With a packet size of 1500 bytes, this amounts to effective bandwidth of 1.44 to 3.6 Mbps. Packet loss probability $p$ starts from 0.01 to 0.1 with a step size of 0.01.

2) Seed Configuration: As mentioned in Section II-C, we consider two ways of achieving “negligible” transcoding cost. In the first reducibility graph $R_1$, the “negligible” transcoding is achieved by dropping FEC packets. In the second reducibility graph $R_2$, transcoding is achieved by dropping FEC packets and/or B-frames. We used the Foreman sequence as the test video. Six different seed configurations are computed, using the two reducibility graphs and four different weights ($\beta = 0, 3, 8, 30$).

3) Results and Analysis: For different $R_1$, $R_2$, and $\beta$, we perform experiments for optimization Mode I which finds the seed assignments maximizing the overall energy with no constraint on the size of the seed set. The results are summarized in Fig. 5.

In order to comprehend the results easily, Fig. 4 gives a detailed legend. In Fig. 4, each symbol, for example, “$\triangle$,” represents an assigned configuration. The number below each symbol is the corresponding expected frame rate (rounded to the nearest integer). The lines partition the network condition into regions corresponding to different configurations.

Fig. 5(a) gives the optimal configuration and best frame rate for each network condition, that is, for each network condition $(b, p)$, the frame rate is

$$\max_{C \in \mathcal{C}} \text{Efr}(C, b, p).$$  \hspace{1cm} (5)

For example, if the network condition $(b, p)$ is $(150, 0.07)$, using the configuration $(N_I, N_P, N_B) = (1, 4, 9)$ represented by “$\triangle$,” we can achieve the best frame rate of 15 fps. In Fig. 5(a), 21 different seed configurations are required to generate the highest reconstructed frame rate for the 180 (18 × 10) possible network conditions.

Fig. 5(b) illustrates the expected overall frame rate using only the standard configuration $(N_I, N_P, N_B) = (2, 8, 20)$ (no frame dropping is considered in this figure). This scheme cannot cope with “bad” network conditions when the bandwidth is low or the loss probability is high. As shown in the figure, the expected frame rates are all zeros when the bandwidth is less than 250 packets since the size of the configuration is 248 packets. The expected frame rate decreases drastically when the loss probability decreases for fixed bandwidth.

Fig. 5(c)-(f) gives the expected frame rate for two reducibility graphs $R_1$ and $R_2$ with different weight $\beta$. The advantage of using reducibility graph $R_2$ is that it provides a better approximation to the optimal configuration. With $R_2$, each seed may lead to different configurations. Compared to $R_1$ using one configuration in each seed, the overall expected frame rate is improved without incurring much switching cost among the seeds.
Adjusting weight $\beta$ can change the size of the seed-set. This can be employed to find solutions for Mode II.

Comparing Fig. 5(c)-(f) with the standard configuration in Fig. 5(b), choosing a better seed-set can achieve significant improvement in terms of expected frame rate. On the other hand, the expected frame rate is not far from performing no configuration reduction, which is illustrated in Fig. 5(a). With $R_1$, the size of the seed set decreases from 6 to 3 when $\beta$ is increased from 3 to 8. Similarly, with $R_2$, the size of the seed set decreases from 6 to 3 when $\beta$ is increased from 3 to 30.

B. Assigning the Seed to Network Conditions

Here, we present experimental results to show how the seed obtained from previous section can improve the server performance when adapting to changing network condition.
1) Video Streaming Scenarios: We consider the following video streaming task: a video server serves multiple clients with different video sequences for each client. The video server packs each video sequence into RTP packets and generates FEC protection packets according to current network condition. The packetized video sequences and appendant FEC protection data are then delivered by the server using UDP. If packets are lost, clients will repair the damaged video sequences with the help of FEC protection.

When the network condition changes, there are three ways for the server to adjust the sending rate as we mentioned in Section I. We compared three servers each employing a different streaming strategy in our experiment.

- **S_{trans}**: The server stores the source video sequence and uses a real-time transcoder (including generation of FEC packets) to transcode the source video with the optimal encoding parameters to match the available network condition for streaming.

- **S_{opt}**: The server stores multiple versions of the source video sequence plus pre-computed FEC packets and each version contains the optimal configuration corresponding to each network condition. For the test sequence, 21 different configurations are stored. The server will switch to the optimal configuration for streaming that matches the network condition so that the highest frame rate can be obtained. This approach also achieves good expected frame rate and has minimum transcoding cost. However, it requires the largest amount of storage and I/O due to switching among different configurations.

- **S_{seed}**: The server stores only several versions of the source video sequence and these versions form the seed-set obtained by our method presented in Section IV. When the network condition changes, the server will switch to a configuration that provides the best reconstructed frame rate among the available configurations in the seed set. This approach reduces storage and I/O cost significantly by decreasing the performance slightly compared to S_{opt}.

For the experiment, we measured the network condition between the National University of Singapore (NUS) and Carnegie Mellon University (CMU) over the Internet 2. The software used is *pathload*, a public-domain software available from www.pathrate.org. Pathload is a statistical tool that measures the available bandwidth (b) and loss rate (p) between 2 points and provides a single long-term estimate when some convergence conditions are met. We modified the software slightly so that it reports both bandwidth and loss characteristics more frequently. Measurements are taken over an hour in 2-min intervals. In order to facilitate our experiments, the bandwidth...
is scaled from 10 to 25 Mbps to the range 1.8–3.5 Mbps. In addition, as an approximation, the measurements are assumed to be taken 30 s apart. Fig. 6(a) shows the various network conditions measured, and Fig. 6(b) shows how the network conditions changed and how each network condition maps to a seed set with three configurations.

2) Measurements and Implementations: We compare the CPU time, I/O time during streaming, the storage space required, and the video quality at the client of $S_{\text{trans}}$, $S_{\text{opt}}$, and $S_{\text{seed}}$. We also compare the overall performance by the number of clients the server can serve simultaneously to achieve real-time performances. We will see in the next section that CPU time is the dominant factor for $S_{\text{trans}}$. For $S_{\text{opt}}$ and $S_{\text{seed}}$, the main concern is I/O time and storage space. We perform the experiments on Linux server with a 2-GHz processor and 512 M RAM. To better understand the resources usage, certain operations are simplified. Network operations like establishing connections, the actual sending of the generated packets are omitted. Specifically, the following steps are carried out during testing.

- For $S_{\text{trans}}$, the operation includes three steps. First, transcoding: we use the codec from libavcodec [7] to transcode the source video sequence with the optimal encoding parameters matching current network condition and generate a transcoded version of the source video. Second, packetizing: the server packs the transcoded versions of source video into RTP packets. Finally, FEC generation: the server generates the number of required packetized FEC protection data for each RTP data packet stored in a transcoded version.

- For $S_{\text{opt}}$ and $S_{\text{seed}}$, during pre-processing, the server performs the same three steps as $S_{\text{trans}}$ to generate RTP packets plus FEC protection and stores each version as a file. For $S_{\text{opt}}$, all versions listed in Fig. 5(a) are generated such that the optimal configurations matching all network conditions are available. For $S_{\text{seed}}$, configurations available correspond to a seed-set obtained by reduction approach. During streaming, the server switches to an appropriate stored configuration to adapt network condition. To simulate the effect of serving multiple clients, multiple processes are run simultaneously and each process is responsible for serving a 900-s video to a client. The total elapsed time is measured. If the time exceeds 900 s and the number of processes (i.e., clients) is, say 40, then it is impossible for the server to achieve real time requirement while serving 40 clients.

At the client-side, the perceived quality of the video stream can be measured by the successfully recovered frame rate. In Section III, we have described an analytical model for the expected recovered frame rate given a particular network condition. However this model may overestimate the expected frame rate since it allows the size of packet to be non-integer. A more accurate estimate can be obtained by simulating the packets loss, that is, for each packet in the stream, it is dropped with current loss probability $p$. FEC is then performed on the received packets and the recovered frame rate can be obtained. The simulation is repeated 100 times for better estimate.

3) Results and Analysis: Fig. 7(a) shows the time required by $S_{\text{trans}}$. The bottom line shows the time required to transcode 1 frame. The average time required per frame is 5.79 ms. Since the playback interval is 33.33 ms per frame, the server will not be able to support more than 6 video simultaneously. This is confirmed by the upper line in Fig. 7(a) which shows the time required when the transcoding simultaneously processes six videos. Although more efficient transcoding maybe available, the speedup probably will not be drastic, without degrading the video quality or compression ratio. The storage required by $S_{\text{trans}}$ is simply the size of the source video which is 0.31 GB. $S_{\text{opt}}$ stores the test-video in 21 versions, each corresponds to an optimal sending configuration. Table II gives the total size of these 21 versions, which is 7.22 GB. Table II also shows the storage size required by $S_{\text{seed}}$ if the size of seed-set is 1, 2, or 3.
In Fig. 7(b), we compare the disk I/O time by $S_{opt}$ and $S_{seed}$. The I/O time is measured by the elapsed wall clock time for accomplishing the execution of the multiple processes which represent multiple clients. The horizontal line at 900 s indicates the point where the server cannot achieve real-time requirement (the video duration is 900 s). Apparently, switching operation among multiple stored videos needs more time than that of among only several stored videos. This observation supports our hypothesis that frequent switching results in low reference locality and thus incurs more I/O costs.

The differences in performance of $S_{seed}$ and $S_{opt}$ get larger as the number of clients increases, which means the performance of $S_{opt}$ will deteriorate as more clients are served. Due to the limited storage space we have in our server, we are unable to conduct experiment for $S_{opt}$ when the number of clients is large (for example, to support 35 clients, 252.7 GB is required). Hence, the extended dotted line in Fig. 7(b) for $S_{opt}$ is extrapolated. From the graph, it is difficult to accurately predict the elapsed time if sufficient storage is available. However, the results clearly indicate that there are noticeable improvement made by $S_{seed}$. In addition, the required storage in $S_{seed}$ is reduced by at least 87%. The difference in elapsed time for different size of seed-set is small. This is probably due to small number of switches is made during streaming.

Fig. 8(a) compares the video quality by $S_{opt}$ and $S_{seed}$ with three seeds. The expected frame rate is obtained by the analyt-
Fig. 8(b) compares the expected frame rate with three seeds. The discrepancy can also be inferred by adjusting the number of seeds. For example, the network conditions from Fig. 6(b). In Fig. 6(b), the regions marked 0, 1, and 2 represent the network conditions assigned to the three corresponding seeds. For example, the network conditions in region 0 are assigned to the seed with the configuration \((N_T, N_F, N_B) = (1, 4, 10)\). When network condition changes to a region supported by seed 1 \(((N_T, N_F, N_B) = (1, 5, 12))\), the server needs to switch to the new seed configuration in order to obtain higher frame rate. Many network conditions change do not require configuration switching, for example, many of the changes supported by seed 2. Another way to present the switching process is shown in Fig. 6(d). In the figure, we can clearly see that while the network condition changes between 600 and 900 s, no configuration change is required. The total number of configuration change in this case is 10.

As a comparison, the switching process of \(S_{opt}\) is shown in Fig. 6(c). In the 900-s duration, \(S_{opt}\) requires 23 switches and uses up to ten different configurations. Note that, while the total number of configurations used in this scenario is only 10, all 21 configurations must be stored since the network conditions are not known in advance and may cover the entire range. The differences in switching frequency and storage requirement will impact the total number of clients supported by the server.

VI. CONCLUSION

We observe that a server’s resource is an important factor in video streaming over a best-effort network, especially in applications where the server needs to serve many clients simultaneously. Ideally, the server should always adapt to the network condition. However, adaptation requires nonnegligible resources. Our experiments demonstrate that a real-time transcoder that always switches to the optimal configuration consumes significant computing power and, hence, is not suitable to serve multiple videos. On the other hand, the experiments also demonstrate that a server that precomputes all of the required configurations needs massive storage space, and its operating system will be overwhelmed with I/O operations during streaming. To strike the right tradeoff of computing time, storage, and I/O usages, we propose a solution that uses a small number of configurations as the seeds, which can be efficiently reduced to the adapted configurations. The proposed energy model in Section IV provides a means of searching for such a good set of seeds. This model captures the fact that transcoding between certain configurations can be done very efficiently. The energy model also incorporates statistical behavior of network condition. Hence, if a good statistic of the clients, and their corresponding network condition is available, it can be used to enhanced performance. Our overall framework is general and could be applied to other methods of video encoding.

REFERENCES

Hang Yu received the B.Eng. and M.Eng. degrees in refrigeration and cryogenics engineering from Shanghai Jiao Tong University, Shanghai, China, in 1998 and 2001, respectively, and the Ph.D. degree from National University of Singapore in 2007.

His research topics were on image and media processing, volume graphics. Currently, he is a Research Fellow with the Institute for Infocomm Research, Singapore. His current research interest is image forensics.

Ee-Chien Chang received the B.Sc and M.Sc. degrees from the National University of Singapore (NUS) and the Ph.D. degree from New York University, New York, in 1998.

He was a Post-Doctoral Fellow with DIMACS and the NEC Research Institute from July 1999 to July 2000. He was with the Department of Computational Science, NUS, in 1998 and 2000. Currently, he is an Assistant Professor with the Department of Computer Science, School of Computing, NUS.

Wei Tsang Ooi (M’08) received the Ph.D. degree in computer science from Cornell University, Ithaca, NY.

He is currently an Assistant Professor with the Department of Computer Science, National University of Singapore, where he does research in multimedia systems, distributed systems, and computer networking.

Mun Choon Chan (M’97) received the B.S. degree from Purdue University, West Lafayette, IN, in 1990 and the M.S. and Ph.D. degrees from Columbia University, New York, in 1993 and 1997, respectively, all in electrical engineering.

From 1991 to 1997, he was a member of the COMET Research Group, working on ATM control and management. From 1997 to 2003, he was a Member of the Technical Staff with the Networking Research Lab. Bell Laboratories, Lucent Technologies, Holmdel, NJ. Currently, he is an Assistant Professor with the Department of Computer Science, National University of Singapore. He has published more than 40 technical papers and holds five patents. His current research interest includes heterogeneous wireless network and sensor networking.

Dr. Chan is a member of the Association for Computing Machinery.

Wei Cheng (S’08) received the B.Eng. degree in civil engineering from Tongji University, Shanghai, China, in 1996 and the M.Eng. degree in communication science and engineering from Fudan University, Shanghai, in 2005. He is currently working toward the Ph.D. degree at the National University of Singapore.

His research interests include encoding and streaming of multimedia data, distributed virtual environment, and P2P systems. Currently, he is working on streaming of 3-D progressive meshes.